To Edwin T. Jaynes
Foreword

At the dawn of the new millennium, robotics is undergoing a major transformation in scope and dimension. From a largely dominant industrial focus, robotics is rapidly expanding into the challenges of unstructured environments. Interacting with, assisting, serving, and exploring with humans, the emerging robots will increasingly touch people and their lives.

The goal of the new series of Springer Tracts in Advanced Robotics (STAR) is to bring, in a timely fashion, the latest advances and developments in robotics on the basis of their significance and quality. It is our hope that the greater dissemination of research developments will stimulate more exchanges and collaborations among the research community and contribute to further advancement of this rapidly growing field.

Probabilistic Reasoning and Decision Making in Sensory-Motor Systems by Pierre Bessièrè, Christian Laugier and Roland Siegwart provides a unique collection of a sizable segment of the cognitive systems research community in Europe. It reports on contributions from leading academic institutions brought together within the European projects Bayesian Inspired Brain and Artifact (BIBA) and Bayesian Approach to Cognitive Systems (BACS). This fourteen-chapter volume covers important research along two main lines: new probabilistic models and algorithms for perception and action, new probabilistic methodology and techniques for artefact conception and development. The work addresses key issues concerned with Bayesian programming, navigation, filtering, modelling and mapping, with applications in a number of different contexts.

The thorough discussion, extensive treatment, and wide span of the work unfolding in these areas reveal the significant advances in the methodologies and technologies. The two projects culminate with this important reference to the robotics and cognitive systems community on the current developments
and new directions in the area of probabilistic reasoning. A fine addition to the STAR series!

Naples, Italy
February 2008

Bruno Siciliano
STAR Editor
Over the next decades, probabilistic reasoning will provide a new paradigm for understanding neural mechanisms and the strategies of animal behaviour at a theoretical level. This will raise the performance of engineering artefacts to a point where they will no longer be easily outperformed by the biological examples they are imitating.

The coordinated works presented in this book are motivated by this conviction and aim to advance in this direction.

The twin common scientific objectives are:

- to reconsider in the light of Bayesian probabilistic reasoning our methodology, models, algorithms and techniques for building artefacts for the “real world”. We gain inspiration from the way living beings have evolved and adapted to the properties of their natural environments, and constructing robots that use these principles;
- to provide a firm Bayesian basis for understanding how biological systems may use probabilistic logic to exploit the statistical properties of their environments, both at the level of neural mechanisms and at the level of strategies, and to use robots to test the validity of these ideas.

To reach these objectives, three axes of research and development have been addressed:

- **Neural basis of probabilistic inference**: The objective of this first axis of research is to identify how the nervous system does (or at least may) implement probabilistic inference. This hypothesize to propose well-defined models of how probabilities are represented and manipulated, and to test predictions with psychophysical performance measures and studies of regional brain activation. The goal is to improve our understanding of neural mechanisms and derived new ideas for the implementation of probabilistic inference in engineering systems.
- **New probabilistic models and algorithms for perception and action**: The main goal of this second axis is to illustrate how probabilistic computation
Preface

may account for global behaviours of organisms in interaction with their environment. In this book, the focus is on specific questions concerning multisensory perception and motion control. New probabilistic models that explain the observed behaviours in humans and animals are proposed and some are implemented on autonomous robots.

- New probabilistic methodology and techniques for artefact design and development: The third axis’ aim is to explore how the Bayesian inference and learning paradigm may be used to develop robots that acquire repertoires of reactive probabilistic behaviours (synergies) and build combinations, hierarchies and temporal sequences of these behaviours (strategies).

Using a common formalism and modeling methodology called Bayesian Programming, this book describes the main outcomes of the last two axes of research. The first axis, concerning the neural basis of Bayesian inference, is another story and is not addressed here.

This book is an outcome of two successive European projects: BIBA\textsuperscript{1} (Bayesian Inspired Brain and Artifact) and BACS\textsuperscript{2} (Bayesian Approach to Cognitive Systems).

The scientific results presented in this book mainly relies on PhD theses initiated during the BIBA project. They are also the result of strong collaborations with academic partners from both information sciences and life sciences: The Autonomous System Laboratory (ASL) from the École Polytechnique Fédérale de Lausanne (EPFL) in Lausanne, Switzerland; the Gatsby Neuroscience Unit and the Department of Physiology of the University College of London (UCL) in London, United Kingdom; the E-Motion group, affiliated with the Centre National de la Recherche Scientifique (CNRS) and the Institut National de Recherche en Informatique et Automatique (INRIA) in Grenoble, France; the Nonlinear System Laboratory (NSL), affiliated with the Massachusetts Institute of Technology (MIT) in Boston, USA; the Laboratoire de Physiologie de la Perception et de l’Action (LPPA) from the Collège de France in Paris, France, the Department of Physiology of Cambridge University in the United Kingdom and The Institut de la Communication Parlé (ICP), affiliated with CNRS in Grenoble, France.

Our first thanks go to the authors of the different papers appearing in this book and especially to the PhD students who did the essential work with much enthusiasm and cheerfulness: Miriam Amavizca, Francis Colas, Christophe Coué, Carla Cavalcante Koike, Julien Diard, Jean Laurens, Olivier Lebeltel, Ronan Le Hy, Kamel Mekhnacha, Cédric Pradalier, Guy Ramel, Jihene Serkhane, Adriana Tapus, Christopher Tay and Manuel Yguel.

\textsuperscript{1} IST-2001-32115
\textsuperscript{2} FP6-IST-027140
We also would like to give a special thanks also to the scientific leaders of the different research groups who contributed through scientific discussions but who do not appear as authors of chapters of this book: Horace Barlow, Alain Berthoz, Peter Dayan, Tony Gardner-Medwin and Jean-Jacques Slotine. They were the instigators and inspirers of a large part of this work.

Finally, we would like to thank Carole Bienvenue, Olivier Malrait and Jean-marc Bollon who took great care of all the administrative financial and technical edition questions related to the work presented in the book.

Pierre Bessière
Christian Laugier
Roland Siegwart

Grenoble, France
February 2008
# Contents

## Part I Introduction

Probability as an alternative to logic for rational sensory–motor reasoning and decision  
*Pierre Bessière* .......................................................... 3

Basic Concepts of Bayesian Programming  
*Pierre Bessière, Olivier Lebeltel* ................................. 21

## Part II Robotics

The *CyCab*: Bayesian navigation on sensory–motor trajectories  
*Cédric Pradalier, Pierre Bessière* .......................... 53

The Bayesian occupation filter  
*M.K Tay, K. Mekhnacha, M. Yguel, C. Coué, C. Pradalier, C. Laugier  
Th. Fraichard, P. Bessière* .......................... 79

Topological SLAM  
*Adriana Tapus, Roland Siegwart* .......................... 103

Probabilistic contextual situation analysis  
*Guy Ramel, Roland Siegwart* .......................... 135

Bayesian Maps: probabilistic and hierarchical models for mobile robot navigation  
*Julien Diard, Pierre Bessière* .......................... 161

Bayesian approach to action selection and attention focusing  
*Carla Cavalcante Koike, Pierre Bessière, Emmanuel Mazer* .......................... 185
Part III Industrial applications

BCAD: a Bayesian CAD system for geometric problems specification and resolution
Kamel Mekhnacha, Pierre Bessière ........................................ 215

3D human hip volume reconstruction with incomplete multimodal medical images
Miriam Amavizca, Christian Laugier, Emmanuel Mazer, Juan Manuel Ahuactzin, Francois Leitner ........................................ 243

Playing to train your video game avatar
Ronan Le Hy, Pierre Bessière ........................................ 275

Part IV Cognitive Modelling

Bayesian modelling of visuo-vestibular interactions
Jean Laurens, Jacques Droulez ........................................ 293

Bayesian modelling of perception of structure from motion
Francis Colas, Pierre Bessière, Jacques Droulez, Mark Wexler ......... 317

Building a Talking Baby Robot: A contribution to the study of speech acquisition and evolution
Jihène E. Serkhane, Jean-Luc Schwartz, Pierre Bessière .............. 347

Index ................................................................. 379

Authors ................................................................. 387
List of Contributors

Ahuactzin Juan Manuel
Probayes
345, rue Lavoisier, Inovallée
38330 St Ismier Cedex,
FRANCE
Juan-Manuel.Ahuactzin
@probayes.com

Amavizca Miriam
INRIA Rhône-Alpes
655 avenue de l'Europe
38334 Montbonnot Saint Ismier,
FRANCE
miriam.amavizca@hotmail.com

Bessière Pierre
CNRS - Grenoble University
Laboratoire LIG
655 avenue de l'Europe
38334 Montbonnot,
FRANCE
Pierre.Bessiere@imag.fr

Colas Francis
LPPA - Collège de France
11, place Marcelin Berthelot
75231 Paris Cedex 05,
FRANCE
francis.colas
@college-de-france.fr

Coué Christophe
ALMA
15 rue Georges Pérec
38400 Saint-Martin d’Hères,
FRANCE
christophe.coue@alma.fr

Diard Julien
Laboratoire de Psychologie et
NeuroCognition CNRS UMR 5105
Université Pierre Mendès France,
BSHM, BP 47
38040 Grenoble Cedex 9,
FRANCE
Julien.Diard@upmf-grenoble.fr

Droulez Jacques
Collège de France - LPPA
11, Place Marcelin Berthelot
75231 Paris Cedex 05
Jacques.Droulez
@college-de-france.fr

Fraichard Thierry
INRIA Rhône-Alpes
655 avenue de l'Europe
38330 Montbonnot
FRANCE
thierry.fraichard@inria.fr
**List of Contributors**

**Koike Carla Cavalcante**  
Brasília University  
Computer Science Department  
Campus Universitário Darcy Ribeiro  
Brasília, 70910-900,  
BRASIL  
ckoike@unb.br

**Laugier Christian**  
INRIA Rhône-Alpes  
655 avenue de l’Europe  
38334 Montbonnot Saint Ismier,  
FRANCE  
Christian.Laugier@inrialpes.fr

**Laurens Jean**  
Vestibulo-Ocular Laboratory  
Department of Neurology, HAL U 13  
Frauenklinikstr. 26  
CH-8091 Zurich,  
SWITZERLAND  
jean.laurens@gmail.com

**Lebeltel Olivier**  
CNRS - Laboratoire LRI  
Bâtiment 490, Université Paris-Sud  
91405 Orsay,  
FRANCE  
Olivier.Lebeltel@lri.fr

**Le Hy Ronan**  
ProBayes  
345 rue Lavoisier, Innovaîlé  
38330 St Ismier,  
FRANCE  
Ronan.LeHy@probayes.com

**Leitner François**  
Aesculap SAS  
24, rue Lamartine  
38320 Eybens,  
FRANCE  
francois.leitner@bbraun.com

**Mazer Emmanuel**  
Probayes  
Innovaîlé 345 rue Lavoisier  
38330 St Ismier,  
FRANCE  
emazer@probayes.com

**Mekhnacha Kamel**  
Probayes  
Innovaîlé 345 rue Lavoisier  
38330 St Ismier,  
FRANCE  
Mekhnacha.Kamel@probayes.com

**Pradalier Cedric**  
CSIRO ICT Centre, Autonomous  
System Laboratory  
PO Box 883  
Kenmore, QLD 4068,  
AUSTRALIA  
cedric.pradalier@ieee.org

**Guy Ramel**  
SYNOVA S.A.  
Ch. de la Dent d’Oche  
1024 Ecublens,  
SWITZERLAND  
guyramel@hotmail.com

**Schwartz Jean-Luc**  
Institut de la Communication Parlée,  
Dept Parole & Cognition de  
GIPSA-lab, UMR 5216, CNRS,  
INPG, UJF, Univ. Stendhal  
INPG, 46 Av. Felix-Viallet F38031  
Grenoble Cedex 1,  
FRANCE  
schwartz@icp.inpg.fr

**Serkhane Jihene E.**  
Cold Spring Harbor Laboratory  
1 Bungtown road  
Cold Spring Harbor 11724 New York,  
USA  
jihene@ecshl.edu
Roland Siegwart  
ETH Zürich, Inst.f. Robotik u. Intelligente Systeme  
Tannenstrasse 3  
8092 Zürich,  
SWITZERLAND  
siegwart@ethz.ch

Tapus Adriana  
University of Southern California - Interaction Lab  
Henry Salvatori, Mailcode 0781  
941 West 37th Place  
Los Angeles, CA 90089-0781, USA  
adriana.tapus@ieee.org

Tay meng keat Christopher  
INRIA Rhône-Alpes  
655 avenue de l’Europe  
38334 Montbonnot Saint Ismier, FRANCE  
Christopher.Tay_meng_keat@inrialpes.fr

Wexler Mark  
Laboratoire de Psychologie de la Perception  
CNRS/Université Paris V  
45, rue des Saints-Pères  
75006 Paris, FRANCE  
mark.wexler@gmail.com

Yguel Manuel  
Institut National Polytechnique de Grenoble  
655, avenue de l’Europe  
38334 Montbonnot Saint-Ismier Cedex, FRANCE  
manuel.yguel@inrialpes.fr
Part I

Introduction
Probability as an alternative to logic for rational sensory–motor reasoning and decision

Pierre Bessière
CNRS – Grenoble Université

1 Incompleteness and uncertainty: a major challenge for sensory–motor systems

We assume that both living creatures and robots must face the same fundamental difficulty: incompleteness (and its direct consequence uncertainty).

Any model of a real phenomenon is incomplete: there are always some hidden variables, not taken into account in the model, that influence the phenomenon. The effect of these hidden variables is that the model and the phenomenon never behave exactly alike. Both living organisms and robotic systems must face this central difficulty: how to use an incomplete model of their environment to perceive, infer, decide and act efficiently.

These difficulties may be clearly explained by taking the robotics field as an example. The dominant paradigm in robotics may be illustrated by Fig. 1.

The programmer of the robot has an abstract conception of its environment. He or she can describe the environment in geometrical terms because the shapes of objects and the map of the world can be specified. The environment may be described in analytical terms because the laws of physics that govern this world are known. The environment may also be described in symbolic terms because both the objects and their characteristics can be named.

The programmer uses this abstract representation to program the robot. The programs use these geometric, analytic and symbolic notions. In a way, the programmer imposes on the robot his or her own abstract conception of the environment.

The difficulties of this approach appear when the robot must link these abstract concepts with the raw signals that it obtains from its sensors and the outputs that it sends to its actuators.

The central origin of these difficulties is the irreducible incompleteness of the models. Hidden variables, which influence the sensory inputs or bias the motor outputs but are not taken into account by the program, prevent
the robot from relating the abstract concepts and the raw sensory–motor data reliably. This problem has been identified for many years in artificial intelligence and robotics under many different names, one of the most well known being the “symbol grounding problem” (see Harnad [1989] and Harnad [1990]).

As the model cannot predict exactly the readings of its sensors and the effects of its motor decisions, the sensory–motor data are then said to be “noisy” or even “aberrant”. An odd reversal of causality occurs that seems to consider that the mathematical model is exact and that the physical world has some unknown flaws.

Controlling the environment is the usual answer to these difficulties. The programmer of the robot looks for the causes of “noise” and modifies either the robot, the environment or both to suppress these “flaws”. The environment is modified until it corresponds to its mathematical model. This approach is both legitimate and efficient from an engineering point of view, even if it seems strange from a scientific point of view. Precise control of both the environment and the tasks ensures that such industrial robots work properly.

However, controlling the environment may not be possible when the robot must act in an environment not specifically designed for it (autonomous robotics). In that case, completely different solutions must be devised. Uncertainty must then be recognized for what it is: a consequence of a lack of knowledge, a direct subproduct of incompleteness. There is no noise out there, in the physical world, but there is a lot of ignorance in here, in our brains and in our models. This ignorance makes sensory–motor reasoning and decision making difficult.
Living beings must face sensory–motor decisions with the exact same handicap as robots. They do not have at hand all the required knowledge about their environment to decide the right thing to do. However, the ones that we can observe survive every day as individuals and have survived millions of years as species. This single fact proves that making not optimal but adequate sensory–motor decision with incompleteness and uncertainty can be done.

2 Probability as an alternative to logic

Rational reasoning with incomplete information is quite a challenge for artificial systems.

The purpose of the subjectivist approach to probabilistic inference and learning is precisely to tackle this problem with a well-established formal theory. During recent years, much progress has been made in this field, from both the theoretical and applied points of view.

As clearly stated by the late Edwin T. Jaynes in his masterpiece book titled “Probability Theory: the Logic of Science” [Jaynes, 2003], probability may be seen as an alternative to, and an extension of logic for rational reasoning in the presence of incompleteness and uncertainty.

This approach may be introduced by taking, once more, the robotics field as an example (see Fig. 2).

![Fig. 2. The subjectivist probabilistic approach to robotics](image-url)
In the symbolic approach, the programmer imposes his or her conception of the environment on the robot, whereas in the probabilistic approach, a common model is built conjointly by the programmer and the robot.

The programmer brings some preliminary knowledge, while the robot provides its own view of the environment by experimenting and observing to collect experimental data.

The programmer’s preliminary knowledge gives the robot some hints about what it may expect to observe. The preliminary knowledge is not a fixed and rigid model purporting completeness as did the program in the symbolic approach. Rather, it is a gauge, with free parameters, waiting to be moulded by the experimental data.

The experimental data obtained from the physical interaction of the robot with its environment, as measured by its sensors and driven by its actuators, reflect all the complexity of this interaction, including the effect of the hidden variables not taken into account by the preliminary knowledge.

Learning sets the values of the free parameters of the preliminary knowledge using the experimental data. This way, the influence of the hidden variables is taken into account and quantified.

In practice, this probabilistic approach to robotics proceeds in two steps as presented in Fig. 3:

- The first step transforms the irreducible incompleteness into uncertainty. Starting from the preliminary knowledge and the experimental data, learning builds probability distributions. The more accurate and pertinent the
preliminary knowledge is, the less uncertain and the more informational
the learned distributions are. The more important the effects of the hidden
variables, the more “noisy” the data and the more uncertain the resulting
probability distributions will be.

• The second step consists of reasoning with the probability distributions
obtained in the first step. To do so, we only require the two basic rules
of Bayesian inference (see next chapter). These two rules are to Bayesian
inference what the resolution principle is to logical reasoning (see Robinson
[1965] and Robinson [1979]). These inferences may be as complex and
subtle as those achieved with logical inference tools.

3 Organization of this book

This book presents 12 different implementations of this approach in very dif-
ferent sensory–motor systems, either by programming robots or by modelling
living systems. Each of these 12 works correspond to a PhD thesis defended
in various European institutions.

As a start, the next chapter introduces Bayesian Programming. This is a
mathematical formalism that defines in simple mathematical terms the way
that probability can be used as an alternative to logic for rational reasoning in
the presence of uncertainty. It is the direct implementation of the principles
presented in the previous section. Bayesian Programming is also a program-
ning and modelling methodology because, to respect the mathematical for-
malism, the programmer should always follow the same steps when building a
model. Finally, Bayesian Programming is a common language for understand-
ing and comparing the different models. This language is used throughout
this book by all the authors and ensures the global coherence of these 12 very
different examples.

The remainder of the book is then organized in three parts: in the first,
we present six applications of Bayesian Programming in robotics and in the
automotive industry; in the second, three industrial applications in Concep-
tion Aided Design (CAD) and video games industry; and finally, in the third
part, three models of living sensory–motor systems.

These 12 chapters address the following subjects.

• Bayesian navigation on sensory–motor trajectories: This chapter describes
the implementation of a trajectory following behavior for an autonomous
mobile robot. This implies solving the following problems: localization with
respect to a map of beacons, localization along a trajectory, self-confidence
estimation, motion control and obstacle avoidance.

The focus of the chapter is mostly on the localization part, especially on the
incremental development of a predictive observation model, at the core of
most Bayesian localization algorithm. The application we describe defines
a trajectory as a sequence of sensory observations and motor commands.
From this representation and an initial position of the robot in a neighborhood of the trajectory, the system first estimate what is the point of the trajectory closest to the current location, then estimates the resulting tracking error and generate commands to correct this tracking error and follow the trajectory.

Finally, these commands are fused with obstacle avoidance constraints. While the vehicle follows the trajectory, the systems monitors the consistency of the effective observations with what can be expected from the sensory-motor trajectory and uses this information to build a self confidence index.

The specificity of this article is that it shows that a complex robotics application can be built using only Bayesian inferences, both for state estimation and control.

- **Bayesian occupation filter**: The automotive industry is particularly interested in adaptive cruise control, where the challenge is to reduce road accidents by implementing better collision detection systems. The major requirement of such a system is a robust tracking system. Most of the existing target tracking algorithms use an object-based representation of the environment. However, these existing techniques must explicitly take into account the difficult problems of data association and occlusion. In view of these problems, a grid-based framework, the Bayesian occupancy filter (BOF), is proposed and studied. Its principle is to combine occupancy grids with a Bayesian filter approach to take care of the persistence of objects in time.

- **Topological SLAM**: This chapter aims at developing a cognitive navigation system based on a compact representation of the environment, obtained using a methodology for encoding information from it—the fingerprint of a place.

  Experimental results in structured indoor and outdoor environments with a mobile robot equipped with a multisensor system composed of two 180° laser range finders and an omnidirectional camera are also reported.

- **Probabilistic contextual situation analysis**: This work proposes a new approach to object recognition that incorporates visual and range information with information about the spatial arrangement between objects (context information).

  It is based on Bayesian networks enabling the fusion and inference of information from different data. The proposed framework first extracts potential objects from the scene image using simple features and characteristics such as colours or the relation between height and width. This basic information is easy to extract but often results in ambiguous situations between similar objects. To eliminate
ambiguity between the detected objects, the relative spatial arrangement (context information) of the objects is used in a second step. The proposed approach is verified through different real-world experiences.

- **Bayesian maps**: This chapter deals with the problem of space modelling in the context of navigation tasks. We first review part of the relevant literature and argue that a marriage between probabilistic methods in robotics and conceptual models of animal navigation in biology might be fruitful, as they appear complementary in their approach. Indeed, we then introduce a method for probabilistic hierarchical modelling of space, called the *Bayesian Map* formalism. This formalism allows incremental building of hierarchies of models, by the use of the abstraction operator, which we define.

In the resulting hierarchies, localization in the high-level model is based on probabilistic competition and recognition in the lower-level models. An abstract location therefore may not be contiguous in physical space, and its recognition is triggered symmetrically by action and perception. Experimental results illustrate the concept and hint at its usefulness for large-scale scenarios.

- **Bayesian approach to action selection and attention focusing**: In this work, we assume that an autonomous sensory–motor system is continually answering the ultimate question: “What can I do next, knowing what I have seen and what I have done so far?”.

This question can be stated mathematically as: $P(M^t \mid z^{0:t} \land m^{0:t-1} \land \pi)$. It gives a probability distribution over the values of the motor control variables $M$ at time instant $t$, knowing the values of the observed variables $Z$ from time instant $0$ to time instant $t$, as well as the values of all motor controls exerted from time instant $0$ to time instant $t - 1$.

Answering this question in this form is intractable. A series of simplifying hypotheses are presented in this chapter to enable solving this problem in real time. Two of these hypotheses, corresponding respectively to “action selection” and “attention focusing ”, are described in detail. Experiments with a middle-sized mobile robot are described to prove the practical validity of the approach.

- **Bayesian CAD system for geometric problem specification and resolution**: We present BCAD, a Bayesian CAD modeller for geometric problem definition and resolution.

This modeller provides tools for (i) modelling geometric uncertainties and constraints and (ii) solving inverse geometric problems while taking into account the propagation of these uncertainties. The proposed method may be seen as a generalization of constraint-based approaches in which we explicitly model geometric uncertainties. Using our methodology, a geometric constraint is expressed as a probability distribution on the system parameters and the sensor measurements, instead of a simple equality or inequality.
To solve geometric problems in this framework, we propose the MCSEM (Monte Carlo Simultaneous Estimation and Maximization) algorithm as a resolution technique able to adapt to problem complexity.

Using three examples, we show how to apply the BCAD system.

- **3D human hip volume reconstruction with incomplete multimodal medical images: an application of computer-assisted surgery to total hip replacement (THR):** The goal of this work is to construct a 3D hip model of a patient from partial data.

  The standard THR planning process simply relies on X-ray images, whereas THR computer-assisted systems use a 3D hip volume reconstructed from medical imaging techniques such as magnetic resonance imaging (MRI), computerized tomography (CT) or scanner. Nevertheless, 3D volume reconstruction has some problems: (i) there is a long waiting time for imaging studies, (ii) because of their cost, machines are not available in all clinics, (iii) a complete automatic segmentation process for 3D volume constructions is not yet available, and (iv) some patients cannot be exposed to some imaging techniques such as MRI, CT or scanner studies.

  Considering these problems, the main contribution of this work is a method for 3D hip volume reconstruction using minimally invasive imaging techniques: a single radiographic image (2D data) and a few echographic images (3D data).

  The proposed method consists of three main stages: (i) data acquisition of the radiographic and echographic images of the patient’s hip, (ii) inference of the patient’s hip atlas, and (iii) 3D hip volume reconstruction by a mesh deformation that adapts to the inferred atlas. These stages pose different problems related to the representation of the generic atlas, to the inference process, and to the radiographic and echographic data processing. To solve this problem, we use Bayesian techniques.

- **Playing to train your video game avatar:** The goal of this chapter is to demonstrate how, by using the Bayesian inverse programming technique, a player of a video game can teach its avatar how to play.

  However, we first show how inverse programming is also very useful for simplifying the programming burden of a video game development team. Bayesian inverse programming consists of expressing independently the conditional probabilities of the conditions, knowing the action. Although atypical, this modelling method appears to be convenient and generic and to lead to very simple learning schemes.

  Examples of programming and training avatars in the first-person shooting game called *Unreal Tournament* are presented.

- **Bayesian modelling of visual–vestibular interactions:** In addition to the five senses usually described, vertebrate species possess a sensory organ that detects motion of the head. This organ is the vestibular system, located in the inner ear. Motion information collected by the vestibular system is crucial for equilibrium. It also contributes to stabilizing the gaze in space during head movements.
In this chapter, we present a Bayesian model of vestibular information processing. This model also incorporates self-motion information available in the visual signal. This model is able to simulate visual–vestibular interactions during various motion stimuli. Furthermore, we use it to reproduce the consequences of lesions of the vestibular organs.

One example, the Off Vertical Axis Rotation (OVAR) is presented in detail.

- **Bayesian modelling of perception of structure from motion**: One of the cues that allows for the perception of the three-dimensional structure of the environment is motion parallax. The displacement of the image projected on the retina, called optical flow, depends on the distance of the object from the eye. Therefore, optical flow can help reconstruct the geometry of the scene. That is called *structure from motion*. However, optical flow is an ambiguous stimulus, as multiple scenes can produce the same flow. In this chapter, we propose a Bayesian model, based on a few simple assumptions that can account for a large range of experimental findings.

- **Building a talking baby robot**: This final chapter describes the elements of a virtual baby robot, which consists of an articulatory model that integrates the non-uniform growth of the vocal tract, a set of sensors, and a learning model. This “robot” is used to study the early stages of speech acquisition by infants.

  The articulatory model delivers sagittal contour, lip shape and acoustic formants from seven input parameters that characterize the configurations of the jaw, the tongue, the lips and the larynx. To simulate the growth of the vocal tract from birth to adulthood, a process modifies the longitudinal dimension of the vocal tract shape as a function of age.

  The auditory system of the robot comprises a phasic system for event detection over time, and a tonic system to track formants. The model of visual perception specifies the basic lip characteristics of a speaker: height, width, area and protrusion.

  The orosensorial channel, which provides the tactile sensation on the lips, the tongue and the palate, is elaborated as a model for the prediction of tongue–palatal contacts from articulatory commands.

  Two studies were performed with this system, focusing on one of the two basic mechanisms that ought to be at work in the initial periods of speech acquisition, namely vocal *exploration* and vocal *imitation*.

  The first study attempted to assess infants’ motor skills before and at the beginning of canonical babbling. It used the model to infer the acoustic regions, the articulatory degrees of freedom and the vocal tract shapes that are the most likely to be explored by actual infants according to their vocalizations.

  The second study simulated data reported in the literature on early vocal imitation to test whether and how the robot was able to reproduce the data and to gain some insights into the actual cognitive representations that might be involved in this behaviour.
4 Essential pointers to related work

Precise bibliographical pointers will be given in each chapter. However, we propose here some essential references that everyone interested in Bayesian subjectivist probabilities should know about. Most of them are reference books or landmark papers. Very different people are using Bayesian approaches in very different fields to deal with incompleteness and uncertainty. Consequently, there are different scientific communities that are often not as well connected as they ought to be for efficient propagation of ideas.

In Physics since the precursory works of Bayes [Bayes, 1763] and Laplace ([Laplace, 1774] and [Laplace, 1814] in French, [Laplace, 1996] in English), numerous results have been obtained using Bayesian inference techniques (to take uncertainty into account) and the maximum entropy principle (to take incompleteness into account). The late Edwin T. Jaynes proposed a rigorous and synthetic formalization of subjectivist probabilistic reasoning with his “Probability as Logic” theory [Jaynes, 2003]. A historical review of this approach was offered by Jaynes [1979], and an epistemological analysis by Matalon ([Matalon, 1967], in French). Theoretical justifications of probabilistic inference and maximum entropy are numerous. The entropy concentration theorems (Jaynes [1982] and Robert [1990]) are among the more rigorous, while Cox’s theorem [Cox, 1961] is the best known, although it has been partially disputed recently by Halpern (Halpern [1999a] and Halpern [1999b]) himself contradicted by Arnborg and Sjödin [2000]. Numerous applications and mathematical tools have been developed by this community, the MaxEnt workshops and associated books being the reference (Smith and Grandy [1985], Tarantola [1986], Bretthorst [1988], Erickson and Smith [1988], Fougère [1990], Smith et al. [1992], Mohammad-Djafari and Demoment [1993], Skilling [1994], Hanson and Silver [1996], Heidbreder [1996], Erickson et al. [1998], Lindén et al. [1999], Rychert et al. [2001], Mohammad-Djafari [2001], Fry [2002], Williams [2003], Erickson and Zhai [2004] and Knuth et al. [2005]).

In cognitive science and neuroscience there are several very active communities using extensively Bayesian modelling. Perception has been clearly stated as an inverse problem by Poggio [1984], consequently, probability is a very interesting tool to model it (Yuille and Bülthoff [1996] and Pizlo [2001]). Perception is also most of the time an ill-posed problem as demonstrated by several examples in the book titled "Perception as Bayesian Inference" by Knill and Richards [1996]. As such, perception is subject to a lot of ambiguities and illusions which can be explained quantitatively with Bayesian models (Geisler and Kersten [2002] and Kersten et al. [2004]). Perception is also, most of the time, the result of the fusion of information coming from a lot of different sensors either from the same sensory modality (object motion: Jacobs [1999], Weiss et al. [2002] and Hillis et al. [2004]; curvature: Drewing and Ernst [2006]) or from different modalities (visuo-auditory: Anastasio et al. [2000]; visuo-vestibular: Zupan et al. [2002]; visuo-haptic: Gepshtein and Banks [2003]; visuo-proprioceptive: Körding and Wolpert [2004]). These
Probability as an alternative to logic

different sources of information can be incoherent provoking conflicts which, again, can be modelled in Bayesian terms (visuo-haptic: Ernst and Banks [2002]; visuo-auditory: Battaglia et al. [2003]; ventriloquism: Alais and Burr [2004] and Banks [2004]). More complex bayesian models can be build as hierarchies of simpler ones as did, for instance, Gopnik and Schulz [2004] to explain the acquisition of causal dependencies by children, Nefian and Hayes [1999] for face recognition or Neal et al. [2003] for tracking of human body motion. Temporal models have also been proposed using Bayesian filtering (body orientation: Gusev and Semenov [1992]; stance control and posture: Van der Kooij et al. [1999] and Kiemel et al. [2002]).

In artificial intelligence, the importance of reasoning with uncertain knowledge has been recognized for a long time. However, the Bayesian approach has clearly emerged as one of the principle trends only since the proposal of Bayesian nets by Pearl [1988] and graphical models by Lauritzen and Spiegelhalter (Lauritzen and Spiegelhalter [1988], Lauritzen [1996], Frey [1998] and Jordan [1999]). Bayesian inference has been proved to be an NP-hard problem [Cooper, 1990]. However, very important technical progress has been achieved recently that permits approximate computation in reasonable time (Saul et al. [1996], Zhang and Poole [1996], Darwiche and Provan [1997]; Koller and Pfeffer [1997], Ruiz et al. [1998], Saul et al. [1996], Jaakkola and Jordan [1999], Jordan et al. [1999] and Aji and McEliece [2000]). The epicentres of this community are essentially the Journal of Artificial Intelligence Research (JAIR, online free journal) and the Uncertainty in Artificial Intelligence conference (UAI).

Traditional robotics programming architectures (Borrelly et al. [1998], Schneider et al. [1998], Dekhil and Henderson [1998] and Mazer et al. [1998]) were in general not concerned with the problem of uncertainty. In robotics, the uncertainty topic was either related to calibration [Bernhardt and Albright, 1993] or to planning problems [Brafman et al., 1997]. In the latter case, some authors have considered modelling the uncertainty of robot motions when planning assembly operations (Lozano-Pérez et al. [1984] and Donald [1988]) or modelling the uncertainty related to the position of a robot in a scene. Everything changed more recently, and Bayesian techniques are now flourishing in robotics. They have been largely used in Partially Observable Markov Decision Processes (POMDP) to plan complex paths in partially known environments (Cassandra et al. [1996], Kaelbling et al. [1998] and Koenig and Simmons [1998]) or for action selection [Rosenblatt, 2000]. Hidden Markov Models (HMMs) are also used to plan complex tasks and recognize situations in complex environments (Aycard [1998] and Thrun [1998]). Finally, much work has been done on probabilistic localization and navigation [Shatkay, 1998], using probabilistic occupancy grids [Konolige, 1997], Markov localization (Thrun et al. [1998], Gutmann et al. [1998], Murphy [1999] and Fox et al. [2000]), correlation-based Markov localization [Konolige and Chou, 1999], Particle filters [Fox et al., 2001] and Kalman filters (Roumeliotis and Bekey [2000a], Roumeliotis and Bekey [2000b]). We have cited here mainly the precursors for
historical reasons; a nice overview of this emerging field is presented in the book by Thrun, Burgard and Fox titled “Probabilistic Robotics” [Thrun et al., 2005]. Despite all this flourishing activity, to the best of our knowledge, the design of a robot programming system and architecture based solely on Bayesian inference had never been investigated before the PhD thesis of Olivier Lebeltel (Lebeltel [1999], Lebeltel et al. [2004]), where Bayesian Programming was first proposed. A paper by Thrun [2000] explored this same direction but with less generality. Bayesian Programming is a simple and generic framework for robot programming in the presence of incompleteness and uncertainty, as we will show in this book. It may be used as a unique formalism to restate and compare numerous classical probabilistic models such as, for instance, Bayesian Networks (BNs), Dynamic Bayesian Networks (DBNs), Bayesian Filters, Hidden Markov Models (HMMs), Kalman Filters, Particle Filters, Mixture Models, and Maximum Entropy Models. This is detailed in a survey by Bessière [2003]. A presentation of the epistemological foundations of Bayesian Programming may be found in two articles by Bessière et al. ([Bessière et al., 1998a] and [Bessière et al., 1998b], in French).

References


The purpose of this chapter is to introduce gently the basic concepts of Bayesian programming.

After a short formal introduction to Bayesian programming, we present these concepts using three simple experiments with the mobile mini-robot Khepera. These three instances have been selected from the numerous experiments we have conducted with this robot for their simplicity and didactic qualities. A more extensive description of the work done with Khepera may be found in a paper in *Advanced Robotics* [Lebeltel et al., 2004] or, in even greater detail, in the PhD thesis of Olivier Lebeltel ([Lebeltel 1999] in French).

We also present the technical issues related to Bayesian programming: inference principles and algorithms and programming language. Although these questions are very interesting, we have kept this part very short, as these technical questions are not central to this book.

1 Basic concepts and notation

1.1 Definitions

Proposition

The first concept we use is the usual notion of a logical proposition that can be either *true* or *false*. Propositions are denoted by lower-case names.

Propositions may be composed to obtain new propositions using the usual logical operators: $a \land b$ denoting the conjunction of propositions $a$ and $b$, $a \lor b$ their disjunction and $\neg a$ the negation of proposition $a$.

Variable

The notion of a discrete variable is the second concept we require. Variables are denoted by names starting with one upper-case letter.
By definition, a discrete variable \( X \) is a set of logical propositions \( x_i \), which stands for “variable \( X \) takes its \( i \)th value”. These logical propositions are mutually exclusive (\( \forall i, j \) with \( i \neq j, x_i \land x_j \) is false) and exhaustive (at least one of the propositions \( x_i \) is true). \( \langle X \rangle \) denotes the cardinality of the set \( X \) (the number of propositions \( x_i \) or equivalently the number of values the variable \( X \) can take).

The conjunction of two variables \( X \) and \( Y \), denoted \( X \land Y \), is defined as the set of \( \langle X \rangle \times \langle Y \rangle \) propositions \( x_i \land y_j \). \( X \land Y \) is a set of mutually exclusive and exhaustive logical propositions, and as such, it is a new variable. Of course, the conjunction of \( n \) variables is also a variable that may be renamed at any time and considered as a unique variable in the sequel.

**Probability**

To be able to deal with uncertainty, we attach probabilities to propositions.

We consider that to assign a probability to a proposition \( a \), it is necessary to have at least some preliminary knowledge, expressed in a proposition \( \pi \). Consequently, the probability of a proposition \( a \) is always conditioned at least by \( \pi \). For each different \( \pi \), \( P(\cdot \mid \pi) \) is an application assigning to each proposition \( a \) a unique real value \( P(a \mid \pi) \) in the interval \([0, 1]\).

Of course, we are interested in reasoning on the probabilities of the conjunctions, disjunctions and negations of propositions, denoted respectively by \( P(a \land b \mid \pi) \), \( P(a \lor b \mid \pi) \) and \( P(\neg a \mid \pi) \).

We are also interested in the probability of proposition \( a \) conditioned by both the preliminary knowledge \( \pi \) and some other proposition \( b \). This will be denoted \( P(a \mid b \land \pi) \).

### 1.2 Inference postulates and rules

**Conjunctions and normalization postulates**

Probabilistic reasoning requires only two basic rules.

1. The **conjunction rule**, which gives the probability of a conjunction of propositions.

   \[
   P(a \land b \mid \pi) = P(a \mid \pi) \times P(b \mid a \land \pi) = P(b \mid \pi) \times P(a \mid b \land \pi)
   \]

2. The **normalization rule**, which states that the sum of the probabilities of \( a \) and \( \neg a \) is one.

   \[
   P(a \mid \pi) + P(\neg a \mid \pi) = 1
   \]

---

3 By contrast, the disjunction of two variables, defined as the set of propositions \( x_i \lor y_j \), is not a variable. These propositions are not mutually exclusive.
In this book, we take these two rules as postulates. As in logic, where the resolution principle (Robinson [1965], Robinson [1979]) is sufficient to solve any inference problem, in discrete probabilities, these two rules (1, 2) are sufficient for any computation. Indeed, we may derive all the other necessary inference rules from these two, especially the rules concerning variables.

1. Conjunction rule for variables.

\[ P(X \land Y \mid \pi) = P(X \mid \pi) \times P(Y \mid X \land \pi) \]

\[ = P(Y \mid \pi) \times P(X \mid Y \land \pi) \]  

(3)

2. Normalization rule for variables.

\[ \sum_X P(X \mid \pi) = 1 \]  

(4)

3. Marginalization rule for variables.

\[ \sum_X P(X \land Y \mid \pi) = P(Y \mid \pi) \]  

(5)

1.3 Bayesian programs

A Bayesian program (BP) is a generic formalism for building probabilistic models and for solving decision and inference problems on these models. Bayesian programs are used throughout this book by all the authors to program their robots or to build their models.

This single formalism for all experiments and applications is a unique tool for comparing the different solutions with one another. It is also the basic specification of the ProBT® API used by most of the authors to automate their probabilistic computations.

A Bayesian program has two parts (see Fig. 1):

- a description that is the probabilistic model of the studied phenomenon or the programmed behaviour; and
- a question that specifies an inference problem to solve using this model.

A description itself contains two parts:

- a specification part that formalizes the knowledge of the programmer; and
- an identification part in which the free parameters are learned from experimental data.

Finally, the specification is constructed from three parts:

- the selection of relevant variables to model the phenomenon;
- a decomposition, whereby the joint distribution on the relevant variables is expressed as a product of simpler distributions; and
- the parametric forms in which either a given mathematical function or a question to another BP is associated with each of the distributions appearing in the decomposition.
### Fig. 1. Generic Bayesian Program

**Description**

The description is the probabilistic model of the studied phenomenon. All the knowledge available about this phenomenon is encoded in the joint probability distribution on the relevant variables.

\[ P(X_1 \land X_2 \land \ldots \land X_N | \pi) \]  

Unfortunately, this joint distribution is usually too complex. The first purpose of the description is to give an effective method of computing the joint distribution in a tractable manner. The second purpose is to specify the learning methods for identifying values of the free parameters from the observed data.

**Specification**

The programmer’s knowledge is specified in a sequence of three steps.

1. **Define the set of relevant variables** \( \{X_1, X_2, \ldots, X_N\} \) on which the joint distribution is defined.
2. **Decompose the joint distribution** to obtain a tractable way to compute it.
   
   The only rule that must be obeyed to obtain a valid probabilistic expression is that each variable shall appear once and only once on the left side of the conditioning bar. This is formally expressed as follows.

   Given a partition of \( \{X_1, X_2, \ldots, X_N\} \) into \( K \) subsets, we define \( K \) variables \( L_0, \ldots, L_K \) each corresponding to one of these subsets.

   Each variable \( L_i \) is obtained as the conjunction of the variables \( X_{i_1} \land X_{i_2} \land \ldots \land X_{i_r} \) belonging to the subset \( i \).

   - Relevant Variables: \( X_1, X_2, \ldots, X_N \)
   - Decomposition:
     \[ P(X_1 \land X_2 \land \ldots \land X_N) = P(L_0) \times P(L_1 | R_1) \times P(L_2 | R_2) \times \ldots \times P(L_K | R_K) \]
   - Parametric Forms:
     - \( P(L_0) \): type of distribution or question to another BP;
     - \( P(L_1 | R_1) \): type of distribution or question to another BP;
     - \( P(L_2 | R_2) \): type of distribution or question to another BP;
     - \( \ldots \)
     - \( P(L_K | R_K) \): type of distribution or question to another BP.
   - Identification:
     Learning the free parameters of the parametric forms
   - Question:
     \( P(\text{Search} | \text{known}) \)
The recursive application of the conjunction rule (3) then leads to the following.

\[
P(X_1 \land X_2 \land \ldots \land X_N | \pi) = P(L_0 | \pi) \times P(L_1 | L_0 \land \pi) \times \ldots \times P(L_K | L_{K-1} \land L_{K-2} \land \ldots \land L_0 \land \pi) \tag{7}
\]

This is an exact mathematical expression.

Conditional independence hypotheses then allow further simplifications. A conditional independence hypothesis for variable \(L_i\) is defined by picking some variables \(X_j\) among the variables appearing in conjunction \(L_{i-1} \land \ldots \land L_0\), calling \(R_i\) the conjunction of these chosen variables and setting:

\[
P(L_i | L_{i-1} \land \ldots \land L_0 \land \pi) = P(L_i | R_i \land \pi). \tag{8}
\]

We then finally obtain the following decomposition.

\[
P(X_1 \land X_2 \land \ldots \land X_N | \pi) = P(L_0 | \pi) \times P(L_1 | R_1 \land \pi) \times \ldots \times P(L_K | R_K \land \pi) \tag{9}
\]

3. Define the parametric forms that give an explicit mean to compute each distribution \(P(L_i | R_i \land \pi)\) appearing in the decomposition. This is achieved by associating with \(P(L_i | R_i \land \pi)\) either a function \(f_{\mu}(L_i)\) or a question to another Bayesian program.

In general, for \(f_{\mu}(L_i)\), \(\mu\) is a vector of parameters that may depend either on \(R_i\), or on the experimental data, or on both. Learning takes place when some of these parameters are computed using the experimental data.

\(P(L_i | R_i \land \pi)\) can be also defined as a question to another Bayesian program \(P(L_i | R_i \land \pi')\). The concept of question will be precisely defined soon, but here it is important to note that this mechanism is a valuable way of building hierarchies of Bayesian programs to obtain complex models. It may be seen as analogous to calling subroutines in classical programming.

**Identification**

In the specification phase discussed above, we can define a parametric form as a function \(f_{\mu}(L_i)\), and we can choose to leave some of its parameters \(\mu\) without given values (or as free parameters). The role of the identification phase is to attach values to these free parameters, usually by means of learning techniques. The goal is to choose these values so that the distributions \(P(L_i | R_i \land \pi)\) are as near as possible to the observed data employed in this learning phase.

Clearly, the distributions \(P(L_i | R_i \land \pi)\) depend on the data \(\delta\) used by the learning process, and, to be rigorous, this dependence should appear in the notations of the distribution: \(P(L_i | R_i \land \delta \land \pi)\). If the experimental data are changed, \(\delta\) changes and the resulting distribution changes accordingly.
Question

Given a description \( P(X_1 \wedge X_2 \wedge \ldots \wedge X_N | \delta \wedge \pi) \), a question is obtained by partitioning the set of relevant variables \( \{X_1, X_2, \ldots, X_N\} \) into three sets: the searched variables, the known variables and the free variables.

We define the variables \( \text{Search} \), \( \text{Known} \) and \( \text{Free} \) as the conjunctions of the variables belonging to these sets.

For a given value \( \text{known} \) of the variable \( \text{Known} \), a question is then defined as the following distribution.

\[
P(\text{Search} | \text{known} \wedge \pi)
\]

The only purpose of Bayesian inference is to compute such distributions efficiently. The presentation of the principles of Bayesian inference is postponed to section 3, as we would like first to clarify and illustrate the above concepts and definitions with some examples of Bayesian programs.

2 Basic examples

2.1 Experimental platform

\textit{Khepera robot}

Khepera is a two-wheeled mobile mini-robot.

![Fig. 2. The Khepera robot.](image-url)
It consists of a base on which turrets with different sensory or motor capabilities may be piled up.

The configuration presented in Fig. 2, for instance, shows a Khepera equipped with a linear camera turret and a micro turbine turret to blow on an object in front of the robot.

The base is 57 mm in diameter and 29 mm high, with a total weight of 80 g.

Khepera was designed at EPFL and is commercialized by K-Team.

The robot is equipped with eight light sensors (six in front and two behind), taking values between 0 and 511 in inverse relation to the light intensity, stored in variables $L_1, \ldots, L_8$ (see Fig. 3). These eight sensors can also be used as infrared proximeters, taking values between 0 and 1023 in inverse relation to the distance from the obstacle, stored in variables $P_1, \ldots, P_8$.

The robot is controlled by the rotation speeds of its left and right wheels, stored in variables $M_l$ and $M_r$, respectively.

From these 18 basic sensory and motor variables, we derived two new sensory variables ($\text{Dir}$, $\text{Prox}$) and one new motor variable ($\text{Vrot}$). They are described below.

- $\text{Dir}$ approximately represents the bearing of the closest obstacle (see Fig. 3). It takes values between $-10$ (obstacle to the left of the robot) and $+10$ (obstacle to the right of the robot).

\[\text{Fig. 3.} \text{Diagram of the Khepera robot, seen from above. The eight small circles are the light sensors. They may also be used as infrared proximeters.}\]
(obstacle to the right of the robot), and is defined as follows.

\[
\text{Dir} = \text{Floor}\left(\frac{90 \times (P_6 - P_1) + 45 \times (P_5 - P_2) + 5 \times (P_4 - P_3)}{9 \times (1 + P_1 + P_2 + P_3 + P_4 + P_5 + P_6)}\right)
\] (11)

- Prox approximately represents the proximity of the closest obstacle (see Fig. 3). It takes values between zero (obstacle very far from the robot) and 15 (obstacle very close to the robot), and is defined as follows.

\[
\text{Prox} = \text{Floor}\left(\frac{\max(P_1, P_2, P_3, P_4, P_5, P_6)}{64}\right)
\] (12)

- The robot is piloted solely by its rotation speed (the translation speed is fixed). It receives motor commands from the \( V_{rot} \) variable, calculated from the difference between the rotation speeds of the left and right wheels. \( V_{rot} \) takes on values between +10 (fastest to the right) and −10 (fastest to the left).

\[
V_{trans} = \frac{M_r + M_l}{2}
\] (13)

\[
V_{rot} = \frac{M_r - M_l}{2}
\] (14)

\[
M_r = V_{trans} + V_{rot}
\] (15)

\[
M_l = V_{trans} - V_{rot}
\] (16)

Environment

For all experiments described in this paper, the Khepera evolves in a square-shaped environment of side 1 m. Textured walls are placed around the environment, and Lego® bricks are employed inside this square as obstacles to the robot movement. In one of the corners, a recess made of higher Lego® bricks and identified by a small light is used as a base for the robot. Figure 4 shows the arena and its features.

2.2 Reactive behaviours

The goal of these first experiments is to teach the robot some reactive behaviours, such as object pushing, obstacle avoidance and contour following.

Goal and experimental protocol

First, in a learning phase, we drive the robot with a joystick to perform one of these behaviours, for instance, pushing objects. During that phase, the robot collects, every tenth of a second, both the values of its sensory variables and the values of its motor variables (determined by the joystick position). This data set is then used to learn the respective behaviours.

Then, in a reprise phase, the robot reproduces the behaviour it has just learned. Every tenth of a second, it decides the values of its motor variables, knowing the values of its sensory variables and the internal representation of the task.
Fig. 4. The environment of the Khepera robot. The walls are textured by a quotation from John W. Tukey: “Far better an approximate answer to the right question, which is often vague, than an exact answer to the wrong question, which can always be made precise”.

Model construction

Variables: A modeller or programmer’s first task is to select the information that is relevant to dealing with the problem, and to name variables that can encode this information.

To push objects, we must have an idea of the position of the objects relative to the robot. The front proximeters provide this information. However, we choose to summarize the information from these six proximeters in the two variables $\text{Dir}$ and $\text{Prox}$.

We also choose to set the translation speed to a constant and to operate the robot by its rotation speed $\text{Vrot}$.

These three variables are all we require to push obstacles. Their definitions are summarized as follows.

$$
\begin{align*}
\text{Dir} &\in \{-10, \ldots, 10\} \quad \langle \text{Dir} \rangle = 21 \\
\text{Prox} &\in \{0, \ldots, 15\} \quad \langle \text{Prox} \rangle = 16 \\
\text{Vrot} &\in \{-10, \ldots, 10\} \quad \langle \text{Vrot} \rangle = 21
\end{align*}
$$

Decomposition: The second step for the modeller in building a Bayesian program is to specify a means of computing the joint distribution on the relevant variables just selected. This second step is called decomposition and consists of expressing the joint distribution as a product of simpler terms. To do this, we first apply the conjunction rule (3) recursively to the joint distribution to obtain the following exact mathematical expression.
\[ P(\text{Dir} \land \text{Prox} \land \text{Vrot} \mid \delta_{\text{push}} \land \pi_{\text{reactive}}) = P(\text{Dir} \mid \delta_{\text{push}} \land \pi_{\text{reactive}}) \times P(\text{Prox} \mid \text{Dir} \land \delta_{\text{push}} \land \pi_{\text{reactive}}) \times P(\text{Vrot} \mid \text{Dir} \land \text{Prox} \land \delta_{\text{push}} \land \pi_{\text{reactive}}) \]  

(18)

We then add some extra knowledge to the model by stating some independence or conditional independence conditions between variables. In this example, this extra knowledge is very simple: we state that knowing the direction of the object relative to the robot does not tell us anything about the distance of this object, i.e. \text{Prox} is independent of \text{Dir}.

\[ P(\text{Prox} \mid \text{Dir} \land \delta_{\text{push}} \land \pi_{\text{reactive}}) = P(\text{Prox} \mid \delta_{\text{push}} \land \pi_{\text{reactive}}) \]  

(19)

With this added knowledge we obtain the final decomposition.

\[ P(\text{Dir} \land \text{Prox} \land \text{Vrot} \mid \delta_{\text{push}} \land \pi_{\text{reactive}}) = P(\text{Dir} \mid \pi_{\text{reactive}}) \times P(\text{Prox} \mid \delta_{\text{push}} \land \pi_{\text{reactive}}) \times P(\text{Vrot} \mid \text{Prox} \land \text{Dir} \land \delta_{\text{push}} \land \pi_{\text{reactive}}) \]  

(20)

\text{Parametric forms:} To be able to compute the joint distribution, we must finally assign parametric forms to each of the terms appearing in the decomposition.

\[ P(\text{Dir} \mid \delta_{\text{push}} \land \pi_{\text{reactive}}) = P(\text{Dir} \mid \pi_{\text{reactive}}) \equiv \text{Uniform} \]

\[ P(\text{Prox} \mid \delta_{\text{push}} \land \pi_{\text{reactive}}) = P(\text{Prox} \mid \pi_{\text{reactive}}) \equiv \text{Uniform} \]  

(21)

\[ P(\text{Vrot} \mid \text{Prox} \land \text{Dir} \land \delta_{\text{push}} \land \pi_{\text{reactive}}) \equiv G(\mu(\text{Prox}, \text{Dir}), \sigma(\text{Prox}, \text{Dir})) \]

We have no \textit{a priori} information about the direction and the distance of the obstacles. Hence, \( P(\text{Dir} \mid \delta_{\text{push}} \land \pi_{\text{reactive}}) \) and \( P(\text{Prox} \mid \delta_{\text{push}} \land \pi_{\text{reactive}}) \) are uniform distributions and do not depend on the data \( \delta_{\text{push}} \); all directions and proximities have the same probability.

For each sensory situation, we believe that there is one and only one rotation speed that should be preferred. The distributions \( P(\text{Vrot} \mid \text{Prox} \land \text{Dir} \land \delta_{\text{push}} \land \pi_{\text{reactive}}) \) are unimodal. However, depending on the situation, the decision to be made for \( \text{Vrot} \) may be more or less certain. This is expressed by assigning a Gaussian\(^6\) parametric form to \( P(\text{Vrot} \mid \text{Prox} \land \text{Dir} \land \delta_{\text{push}} \land \pi_{\text{reactive}}) \).

Indeed, \( P(\text{Vrot} \mid \text{Prox} \land \text{Dir} \land \delta_{\text{push}} \land \pi_{\text{reactive}}) \) stands for a family of Gaussian distributions, one for each possible position of the object relative to

\(^6\) To be exact, these distributions may not be Gaussians as \( \text{Vrot} \) is a discrete variable. These distributions are “bell shaped” distributions, the ProBT\(\text{R}^{\text{©}}\) discrete approximation for Gaussians.
the robot (i.e. one for each of the $21 \times 16$ possible values of \textit{Dir} and \textit{Prox}). Consequently there are $21 \times 16$ means and $21 \times 16$ standard deviations that have not been specified and that depend on the data $\delta_{\text{push}}$.

\textit{Model identification}

The goal of the model identification phase is to learn the values of these $21 \times 16 \times 2$ free parameters.

To obtain data to learn these parameters, we drive the robot with a joystick\textsuperscript{7}.

Every tenth of a second, we measure the value of \textit{Dir} and \textit{Prox}, and we derive the value of \textit{Vrot} from the position of the joystick. A datum collected at time $t$ is a triplet ($vrot^t$, $dir^t$, $prox^t$). A series of such triplets is one instance of our data set $\delta_{\text{push}}$.

The free parameters of the parametric forms can then be identified by computing the means and standard deviations of \textit{Vrot} for each possible position of the obstacle.

\textit{Model utilization}

To render the reactive behaviours just learned, the Bayesian robot controller is called every tenth of a second:

1. The sensors are read and the values of $dir^t$ and $prox^t$ are computed.
2. The Bayesian program is run with the following question.

\begin{equation}
P(Vrot \mid prox^t \land dir^t \land \delta_{\text{push}} \land \pi_{\text{reactive}}) \tag{22}
\end{equation}

The variable $Vrot$ is set to a value $vrot^t$ drawn from this distribution.
3. $vrot^t$ is used as the motor command.

\textit{Bayesian program}

All this information may be summarized by the Bayesian program presented in Fig. 5.

All the necessary information required to deal with the problem is present in this program: nothing is missing, nothing is “hidden” anywhere else in the computer doing the computation.

\textit{Results, lessons and discussion}

\textit{Results} The Khepera learns how to push obstacles in 20 to 30 seconds\textsuperscript{8}. It learns the particular dependencies, corresponding to this specific behaviour, between the sensory variables \textit{Dir} and \textit{Prox} and the motor variable \textit{Vrot}.

This behaviour is largely independent of the particular characteristics of the objects (such as weight, colour, balance or nature)\textsuperscript{9}.

\textsuperscript{7} See movie: http://www.bayesian-programming.org/videoB1Ch2-1.html
\textsuperscript{8} See movie: http://www.bayesian-programming.org/videoB1Ch2-2.html.
\textsuperscript{9} See movie: http://www.bayesian-programming.org/videoB1Ch2-2.html.
Lesson 1: A generic method of building Bayesian programs

In this experiment, we apply a precise three-step method to program the robot:

1. Specification: Define the programmer’s preliminary knowledge about the studied phenomenon.
   a) Choose the pertinent variables.
   b) Decompose the joint distribution.
   c) Define the parametric forms.
2. Identification: Identify the free parameters of the preliminary knowledge.

In the rest of this book, the same method will be used by all authors to program their applications or to build their models.

Lesson 2: Bayesian program = Specification + Data + Question

Numerous different behaviours may be obtained by changing some of the different components of a Bayesian program in the following ways.

- It is possible to change the question, keeping the description unchanged. For instance, if the Prox information is no longer available because of some failure, the robot may still try to push an obstacle knowing only its direction. The query is then as follows.

\[ P(V_{rot} | \text{proxt} \land \text{dirt} \land \delta_{push} \land \pi_{reactive}) \]  \hspace{1cm} (23)

- It is possible to change the data, keeping the preliminary knowledge unchanged. For instance, with the exact same preliminary knowledge \( \pi_{reactive} \), we taught the robot to follow contours. The only thing we changed was the way to drive the robot during the learning phase. For this completely different behaviour, a completely different data set \( \delta_{follow} \)
is obtained, which leads to different parameter values. Questioning the program using these parameters results in a different behaviour of the robot\textsuperscript{10}, as can be seen in Fig. 6.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{fig6.png}
\caption{Contour following (superposed images).}
\end{figure}

- Finally, it is possible to change the specification, which leads to completely different behaviours. Numerous examples will be presented in the rest of this book.

2.3 Sensor fusion

\textit{Goal and experimental protocol}

The goal of this experiment is to integrate the data from the eight light sensors to determine the bearing of a light source.

This will be obtained in two steps. We first specify individual descriptions for each sensor. Then we combine these eight descriptions to form a global one.

\textit{Sensor model Bayesian program}

The model of one sensor is defined by the Bayesian program of Fig 7.

\textsuperscript{10} See movie: \url{http://www.bayesian-programming.org/videoB1Ch2-3.html}. 
### Relevant Variables:

- $L_i$: reading of the $i$th sensor
- $\Theta_i$: bearing of the light source relative to the sensor

#### Decomposition

\[
P(L_i \wedge \Theta_i \mid \pi_{\text{sensor}}) = P(\Theta_i \mid \pi_{\text{sensor}}) \times P(L_i \mid \Theta_i \wedge \pi_{\text{sensor}})
\]

#### Parametric Forms

- $P(\Theta_i \mid \pi_{\text{sensor}}) \equiv \text{Uniform}$
- $P(L_i \mid D_i \wedge \Theta_i \wedge \pi_{\text{sensor}}) \equiv G(\mu(\Theta_i), \sigma)$

#### Identification

No learning, parameters given by chart

#### Question

$P(\Theta_i \mid l_i \wedge \pi_{\text{sensor}})$

---

**Variables:** To build a model of light sensor $i$, we only require two variables: $L_i$, the reading of the $i$th sensor, and $\Theta_i$, the bearing of the light source relative to the sensor.

- $L_i \in \{0, \ldots, 511\}$  \[\langle L_i \rangle = 512\]
- $\Theta_i \in \{-170, \ldots, 180\}$  \[\langle \Theta_i \rangle = 36\]  \hspace{1cm} (24)

**Decomposition:** The decomposition simply specifies that the reading of a sensor obviously depends on the position of the light source.

**Parametric forms:** As we have no a priori information on the position of the source, we state:

\[
P(\Theta_i \mid \pi_{\text{sensor}}) \equiv \text{Uniform}
\]

The distribution $P(L_i \mid \Theta_i \wedge \pi_{\text{sensor}})$ is very easy to specify because it corresponds exactly to the kind of information that the sensor supplier provides: the expected readings of the device when exposed to a light. For Khepera’s light sensors, the supplier gave a chart for the average readings of the sensors according to the bearing, i.e. the function $\mu(\Theta_i)$ (see Fig. 8). We consider that the standard deviation $\sigma$ is a constant independent of $\Theta_i$.

**Sensor fusion Bayesian program**

The fusion model for the eight sensors is defined by the Bayesian program of Fig 9.

**Variables:** The relevant variables are the eight variables $L_i$ (the readings of the eight sensors) and the eight variables $\Theta_i$ (the bearings of the light source relative to the sensors), plus the variable $\Theta$ for the bearing of the light source relative to the robot.
Fig. 8. The mean readings of the light sensors according to the bearing of the light source. These light sensors have the peculiarity of giving a very low reading when in full light and a very high reading when in the dark.

Program 8

 Deg. of light source bearing

<table>
<thead>
<tr>
<th>Intensity Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>100</td>
</tr>
<tr>
<td>200</td>
</tr>
<tr>
<td>300</td>
</tr>
<tr>
<td>400</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Light Source Bearing (deg)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-150</td>
</tr>
<tr>
<td>-100</td>
</tr>
<tr>
<td>-50</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>50</td>
</tr>
<tr>
<td>100</td>
</tr>
<tr>
<td>150</td>
</tr>
</tbody>
</table>

Program 8

<table>
<thead>
<tr>
<th>Relevant Variables:</th>
</tr>
</thead>
<tbody>
<tr>
<td>( L_1, \ldots, L_8, \theta_1, \ldots, \theta_8, \theta )</td>
</tr>
</tbody>
</table>

**Decomposition:**

\[
P(L_1 \land \ldots \land L_8 \land \theta_1 \land \ldots \land \theta_8 \land \theta \mid \pi_{\text{fusion}}) = 
\prod_{i=1}^{8} P(\theta_i \mid \theta \land \pi_{\text{fusion}}) \times P(L_i \mid \theta_i \land \pi_{\text{sensor}})
\]

**Parametric Forms:**

\[
P(\theta \mid \pi_{\text{fusion}}) \equiv \text{Uniform}
\]

\[
P(\theta_i \mid \theta \land \pi_{\text{fusion}}) \equiv \text{Dirac function describing the position of sensor } i
\]

\[
P(L_i \mid \theta_i \land \pi_{\text{fusion}}) \equiv P(L_i \mid \theta_i \land \pi_{\text{sensor}})
\]

**Identification:**

No learning

**Question:**

\[
P(\theta \mid l_8 \land \ldots \land l_1 \land \pi_{\text{fusion}})
\]

Fig. 9. Sensor fusion BP

Decomposition: Knowing the bearing \( \theta \) of the light source relative to the robot is sufficient to compute the bearing \( \theta_i \) relative to the sensor, as we know the position of each sensor on the robot. Consequently:

\[
P(\theta_i \mid \theta_{i-1} \land \ldots \land \theta_1 \land \theta \land \pi_{\text{fusion}}) = P(\theta_i \mid \theta \land \pi_{\text{fusion}}).
\]  \hspace{1cm} (26)
Furthermore, we consider that if we know $\Theta_i$ (the bearing of the light source relative to the sensor), then we have all the necessary information about the reading $L_i$.

$$P(L_i \mid L_{i-1} \land \ldots \land L_1 \land \Theta_8 \land \ldots \land \Theta_1 \land \Theta \land \pi_{\text{fusion}}) = P(L_i \mid \Theta_i \land \pi_{\text{fusion}})$$

(27)

These simplifications may seem peculiar, as obviously the readings of the different light sensors are not independent. The exact meaning of these equations is that we consider the position of the light source to be the main reason for the contingency of the readings. Consequently, we state that, knowing $\Theta_i$, the readings $L_i$ are independent. The position of the light source is the cause of the readings, and knowing the cause, the consequences are independent. This is, indeed, a very strong hypothesis. The sensors may be correlated for numerous other reasons. For instance, ambient temperature influences the functioning of any electronic device and consequently correlates their responses. However, we choose, as a first approximation, to disregard all these other factors.

**Parametric forms:** We do not have any a priori information on $\Theta$.

$$P(\Theta \mid \pi_{\text{fusion}}) \equiv \text{Uniform}$$

(28)

Knowing the bearing of the light source relative to the robot and the position of the sensor, we may compute exactly the bearing relative to the sensor. $P(\Theta_i \mid \Theta \land \pi_{\text{fusion}})$ is a Dirac distribution.

$P(L_i \mid \Theta_i \land \pi_{\text{fusion}})$ is obtained from the model of each sensor as specified in the previous section.

$$P(L_i \mid \Theta_i \land \pi_{\text{fusion}}) \equiv P(L_i \mid \Theta_i \land \pi_{\text{sensor}})$$

(29)

**Identification:** As there are no free parameters, no identification is required.

**Question:** To find the position of the light source, the standard query is.

$$P(\Theta \mid l_8 \land \ldots \land l_1 \land \pi_{\text{fusion}})$$

(30)

**Results, lessons and discussion**

**Results:** Figure 10 presents the results obtained for a light source with a bearing of $10^\circ$. The eight peripheral figures present the distributions $P(L_i \mid \Theta_i \land \pi_{\text{sensor}})$ corresponding to the eight light sensors. The central schema presents the result of the fusion: the distribution $P(\Theta \mid l_8 \land \ldots \land l_1 \land \pi_{\text{fusion}})$. Even poor information coming from each separate sensor may combine as a certainty.
Lesson 3: Breaking the complexity using conditional independence

The conditional independence hypothesis that permits the transformation of:

\[ P(L_1 \land \ldots \land L_8 \land \Theta_1 \land \ldots \land \Theta_8 \land \Theta \mid \pi_{\text{fusion}}) \]  

(31)

into:

\[ P(\Theta \mid \pi_{\text{fusion}}) \times \prod_{i=1}^{8} [P(\Theta_i \mid \Theta \wedge \pi_{\text{fusion}}) \times P(L_i \mid \Theta_i \wedge \pi_{\text{fusion}})] \]  

(32)

is one of the main tools available to simplify the treated problem. More than any clever inference algorithm, it is the essential way to keep computation tractable. For instance, here the size of the search space for the joint distribution (31) is \(36^9 \times 512^8 \simeq 2^{109}\), when the size of the search space for the decomposition (32) is \(36 + (512 \times 36) \times 8 \simeq 2^{16}\).

Lesson 4: Calling Bayesian subroutines

The specification \(P(L_i \mid \Theta_i \wedge \pi_{\text{fusion}}) \equiv P(L_i \mid \Theta_i \wedge \pi_{\text{sensor}})\), where a distribution appearing in a decomposition is defined by a question to another Bayesian program, may be seen as the probabilistic analogue of a subroutine call in regular programming. This Bayesian subroutine call mechanism will play the same role as the usual one: it will allow us to build complex Bayesian programs as hierarchies of embedded calls to successively simpler Bayesian programming building blocks. Numerous examples will be found in this book.
Lesson 5: Sensor fusion method  In the above experiment, we have seen a simple instance of a general method for carrying out data fusion. The key point of this method is in the decomposition of the joint distribution, which has been considerably simplified under the hypothesis that “knowing the cause, the consequences are independent”. This is a very strong hypothesis, although it may be assumed in numerous cases. We presented this method using a very simple case for didactic purposes, but it can also be very effective in more complicated problems in a large variety of applications. In this book, for example, examples of data fusion can be found in nearly all chapters.

Lesson 6: No inverse and no ill-posed problems in the probabilistic framework  In this experiment, another fundamental advantage of Bayesian programming is clearly evident: the description is neither a direct nor an inverse model. Mathematically, all variables appearing in a joint distribution play exactly the same role. This is why any question may be asked of a description. Consequently one may define the description in one way \( P(L_i \mid \Theta_i \land \pi_{fusion}) \) and question it in the opposite way \( P(\Theta \mid L_8 \land \ldots \land L_1 \land \pi_{fusion}) \).

In theory, any inverse problem may be solved when expressed in a probabilistic framework. In practice, some of these inverse problems may require high computational resources. However, this is a major difference from non-probabilistic modelling, where the inverse problem may be solved only in rare cases. Furthermore, there are no ill-posed problems in a probabilistic framework. If a question has several solutions, the probabilistic answer will simply have several peaks.

2.4 Behaviour combination

Goal and experimental protocol

In this experiment, we want the robot to go back to its base, where it can recharge its batteries. This will be obtained with no further teaching. As the robot’s base is lit, the light gradient usually gives good hints on its direction. Consequently, we will obtain the homing behaviour by combining the obstacle avoidance behaviour and a phototaxy behaviour. By programming this behaviour, we will illustrate one possible way to combine Bayesian programs that make use of a command variable.

Homing Bayesian program

The Bayesian program for homing is given in Fig 11.

Variables: We use Dir, Prox, \( \Theta \) and Vrot, the four variables already used in the two composed behaviours (avoidance and phototaxy). We also require a new variable \( H \) that acts as a command to switch from avoidance to phototaxy.
Decomposition: Should only depend on Prox.

Close to obstacles, we want the robot to avoid them. Hence, we consider that independent of one another.

Parametric forms:

Far from any objects, we want the robot to move toward the light. Very

Finally, we also assume that \( Vrot \) must depend on the other four variables. These programmer choices lead to the following decomposition.

\[
P(Dir \land Prox \land \Theta \land Vrot \land H \mid \pi_{\text{homing}}) =
\]
\[
P(Dir \land Prox \land \Theta \mid \pi_{\text{homing}})
\]
\[
\times P(H \mid Prox \land \pi_{\text{homing}})
\]
\[
\times P(Vrot \mid Dir \land Prox \land \Theta \land H \land \pi_{\text{homing}})
\]

Parametric forms: We have no a priori information about either the direction and distance of objects or the direction of the light source. Consequently, we state:

\[
P(Dir \land Prox \land \Theta \mid \pi_{\text{homing}}) \equiv \text{Uniform.}
\]
$H$ is a command variable to switch from avoidance to phototaxy. This means that when $H = \text{avoidance}$, the robot should behave as it learned to do in the description $P(V_{\text{rot}} \mid \text{Dir} \land \text{Prox} \land \pi_{\text{avoidance}})$, and when $H = \text{phototaxy}$, the robot should behave according to the description $P(V_{\text{rot}} \mid \theta \land \pi_{\text{phototaxy}})$.

Therefore, we state:

\[
P(V_{\text{rot}} \mid \text{Dir} \land \text{Prox} \land \theta \land [H = \text{avoidance}] \land \pi_{\text{homing}}) \\
\equiv P(V_{\text{rot}} \mid \text{Dir} \land \text{Prox} \land \pi_{\text{avoidance}}) \\
P(V_{\text{rot}} \mid \text{Dir} \land \text{Prox} \land \theta \land [H = \text{phototaxy}] \land \pi_{\text{homing}}) \\
\equiv P(V_{\text{rot}} \mid \theta \land \text{Prox} \land \pi_{\text{phototaxy}}).
\] (36)

We want a smooth transition from phototaxy to avoidance as we move closer and closer to objects. Hence, we finally state (see Fig. 12):

\[
P([H = \text{avoidance}] \mid \text{Prox} \land \pi_{\text{homing}}) = \text{Sigmoid}_{\alpha, \beta}(\text{Prox}) \\
(\alpha = 9, \beta = 0.25).
\] (37)

Of course, we have:

\[
P([H = \text{phototaxy}] \mid \text{Prox} \land \pi_{\text{homing}}) = 1 - P([H = \text{avoidance}] \mid \text{Prox} \land \pi_{\text{homing}}).
\] (38)

**Identification:** As there are no free parameters, no identification is required.
Question: While Khepera is returning to its base, we do not know in advance when it should avoid obstacles or when it should go toward the light. Consequently, to render the homing behaviour, we will use the following question where $H$ is unknown.

$$P(V_{rot} \mid \text{dir} \land \text{prox} \land \theta \land \pi_{homing})$$ (39)

According to the marginalization rule (5), the answer to this question may be computed by summing on the missing variable $H$.

$$P(V_{rot} \mid \text{dir} \land \text{prox} \land \theta \land \pi_{homing}) = \sum_{H} P(V_{rot} \land H \mid \text{dir} \land \text{prox} \land \theta \land \pi_{homing})$$ (40)

Using the conjunction rule (3), we obtain:

$$P(V_{rot} \mid \text{dir} \land \text{prox} \land \theta \land \pi_{homing}) = \sum_{H} \frac{P(V_{rot} \land H \land \text{dir} \land \text{prox} \land \theta \land \pi_{homing})}{P(H \land \text{dir} \land \text{prox} \land \theta \land \pi_{homing})}$$. (41)

Replacing the joint distribution by its decomposition leads to:

$$P(V_{rot} \mid \text{dir} \land \text{prox} \land \theta \land \pi_{homing}) = \sum_{H} P(H \mid \text{prox} \land \pi_{homing}) \times P(V_{rot} \mid \text{dir} \land \text{prox} \land \theta \land H \land \pi_{homing})$$. (42)

Finally, developing for the two possible values of $H$, we obtain:

$$P(V_{rot} \mid \text{dir} \land \text{prox} \land \theta \land \pi_{homing}) = P([H = \text{avoidance}] \mid \text{prox} \land \pi_{homing}) \times P(V_{rot} \mid \text{dir} \land \text{prox} \land \theta \land \pi_{avoidance}) + P([H = \text{phototaxy}] \mid \text{prox} \land \pi_{homing}) \times P(V_{rot} \mid \theta \land \pi_{phototaxy})$$. (43)

This final equation shows that the robot executes a movement that results from the weighted combination of avoidance and phototaxy.

Far from any objects ($\text{prox} = 0$), $P([H = \text{phototaxy}] \mid [\text{prox} = 0] \land \pi_{homing}) = 1$, the robot uses pure phototaxy.

$$P(V_{rot} \mid \text{dir} \land \text{prox} \land \theta \land \pi_{homing}) = P(V_{rot} \mid \theta \land \pi_{phototaxy})$$ (44)

Very close to objects ($\text{prox} = 15$), $P([H = \text{phototaxy}] \mid [\text{prox} = 15] \land \pi_{homing}) = 0$, the robot uses pure avoidance.

$$P(V_{rot} \mid \text{dir} \land \text{prox} \land \theta \land \pi_{homing}) = P(V_{rot} \mid \text{dir} \land \text{prox} \land \pi_{avoidance})$$ (45)

In between, it combines the two behaviours.
Results, lessons and discussion

Results Efficient homing behaviour is obtained with the above programming, and Fig. 13 shows one illustration\textsuperscript{11}.

Figures 14 and 15 present the probability distributions obtained when the robot must avoid an obstacle on the left with a light source also on the left. As the object is on the left, the robot must turn right to avoid it. This is what happens when the robot is close to the objects (see Fig. 14). However, when the robot is further from the object, the presence of the light source on the left influences the way the robot avoids obstacles. In that case, the robot may turn left despite the presence of the obstacle (see Fig. 15).

Lesson 7: A probabilistic if–then–else In this experiment, we present a simple instance of a general method for combining descriptions to obtain a new combined behaviour. This method uses a command variable $H$ to switch from one composing behaviour to another. A probability distribution on $H$ knowing some sensory variables should then be specified or learned. The new description is finally used by asking questions in which $H$ is unknown. The resulting sum on the different cases of $H$ does the combining.

This shows that Bayesian robot programming allows easy, clear and rigorous specification of such combinations. This seems to be an important benefit compared with other methods that have great difficulty in combining

\textsuperscript{11} A video of the example illustrated here can be seen at http://www.bayesian-programming.org/videoB1Ch2-4.html and a more complex one at http://www.bayesian-programming.org/videoB1Ch2-5.html
Fig. 14. Homing behaviour. Khepera is close to an object on its left, and the light source is also on its left. The top left distribution shows the knowledge on $V_{rot}$ given by the phototaxy description; the top right is $V_{rot}$ given by the avoidance description; the bottom left shows the knowledge of the “command variable” $H$; finally the bottom right shows the resulting combination on $V_{rot}$.

Fig. 15. Homing behaviour. Khepera is further from the object on its left. This figure is structured in the same way as Fig. 14.
behaviours, such as Brooks’ subsumption architecture (Brooks [1986], Maes [1990]) or neural networks.

Description combination appears to implement naturally a mechanism similar to “Hierarchical Mixture of Experts” [Jordan and Jacobs, 1994] and is also closely related to mixture models (see McLachlan and Peel [2000] for a reference document on mixture models and see Bessière [2003] for details of the relation between description combination and mixture models).

Finally, from a programming point of view, description combination can be seen as a probabilistic if–then–else construction. $H$ is the condition. If $H$ is known with certainty, then we have a normal branching structure. If $H$ is known with some uncertainty through a probability distribution, then the two possible consequences are automatically combined using weights proportional to this distribution.

## 3 Bayesian inference principle and tools

Bayesian programming as proposed and exemplified in the previous section offers both a mathematical formalism and a programming methodology for building programs and models for sensory–motor systems in the presence of incompleteness and uncertainty.

However, we want complete working solutions able either to drive a robot or to simulate a biological process.

Consequently, we require an inference engine to automate computations and a programming language to translate the mathematical models into computer language.

The ProBT® API offers both. The purpose of this section is to present very briefly the principle of this API. We have kept this part very short and schematic as the technical aspects of the computer implementation are not central to this book.

### 3.1 Inference engine principles

Given a description and a question, the role of the inference engine is to answer the question.

The description gives a means of computing the joint probability distribution on the relevant variables. The joint probability distribution is expressed as a product of simpler terms according to the decomposition.

$$P(X_1 \land X_2 \land \ldots \land X_N \mid \pi) = P(L_0 \mid \pi) \times P(L_1 \mid R_1 \land \pi) \times \ldots \times P(L_K \mid R_K \land \pi)$$

The values of all the parameters of all the distributions appearing in this product are known; they have been either fixed by human experts or learned by experience.
The question is specified by:

\[ P(\text{Search} \mid \text{known} \land \pi). \]  

(47)

Applying the marginalization rule (5) to the question, we obtain:

\[ P(\text{Search} \mid \text{known} \land \pi) = \sum_{\text{Free}} P(\text{Search} \land \text{Free} \mid \text{known} \land \pi). \]  

(48)

Applying the conjunction rule (3) to this last expression, we obtain:

\[ P(\text{Search} \mid \text{known} \land \pi) = \sum_{\text{Free}} \frac{P(\text{Search} \land \text{Free} \land \text{known} \mid \pi)}{P(\text{known} \mid \pi)}. \]  

(49)

Applying the marginalization rule (5) to the denominator, we obtain:

\[ P(\text{Search} \mid \text{known} \land \pi) = \frac{\sum_{\text{Free}} P(\text{Search} \land \text{Free} \land \text{known} \mid \pi)}{\sum_{\text{Search,Free}} P(\text{Search} \land \text{Free} \land \text{known} \mid \pi)}. \]  

(50)

This expression explains why we make so much effort in the description to be able to compute the joint distribution. Indeed, if the joint distribution can be computed, then any possible question can be answered.

Most of the time, exact values of the probabilities are not necessary. Rather, these probabilities can be compared, either to draw according to the distribution or to find the best value. In these cases, the denominator appears to be a normalization constant \( Z \) that is not explicitly computed.

\[ P(\text{Search} \mid \text{known} \land \pi) = \frac{1}{Z} \times \sum_{\text{Free}} P(\text{Search} \land \text{Free} \land \text{known} \mid \pi) \]  

(51)

Finally, the joint distribution can be replaced by the decomposition:

\[ P(\text{Search} \mid \text{known} \land \pi) = \frac{1}{Z} \times \sum_{\text{Free}} P(L_0 \mid \pi) \times P(L_1 \mid R_1 \land \pi) \times \cdots \times P(L_K \mid R_K \land \pi). \]  

(52)

In theory, any probabilistic problem can be solved in this way. In practice, however, the corresponding computation may not be tractable.

The first difficulty is finding the optima of \( P(\text{Search} \mid \text{known} \land \pi) \). This may be very difficult as it is an optimization problem in the search space defined by the conjunction of variables Search. There may be many variables in this conjunction and, consequently, the search space may have a very high dimension and be huge.

Even worse, to evaluate the function to optimize (proportional to the probability) at a single point of this search space, one must evaluate:

\[ \sum_{\text{Free}} P(L_0 \mid \pi) \times P(L_1 \mid R_1 \land \pi) \times \cdots \times P(L_K \mid R_K \land \pi). \]  

(53)
This may also be difficult, as it is an integration problem in the high-dimensional space defined by the conjunction of variables Free. Integrating in a high-dimensional space is very similar to an optimization problem, as the main concern is to find the optimum of the function to integrate (here $P(L_0 \mid \pi) \times P(L_1 \mid R_1 \land \pi) \ldots \times P(L_K \mid R_K \land \pi)$) that contributes the most to the sum.

All the difficulty of probabilistic inference is in these two optimization problems in high-dimensional spaces.

Exact inference has been proved to be NP-hard by Cooper [1990], as has the general problem of approximate inference [Dagum and Luby, 1993].

Numerous heuristics and restrictions to the generality of the possible inferences have been proposed to achieve acceptable computation times. A review of the main algorithms is presented in Bessière [2003]. Most of the algorithms take the point of view of graphical models and are based on graph manipulation.

In ProBT®, we preferred an algebraic point of view.

The ProBT® inference engine is a unifying framework for exact and approximate Bayesian inference. To solve a question, it processes in two phases: the first consists of symbolic simplification, which leads to a drastic reduction in the amount of computation required by symbolically transforming the expression (52) to a simpler one; and in the second phase, this simplified expression is computed effectively.

ProBT® includes most of the classical Bayesian inference techniques, but it is based on three original algorithms.

- The “Successive Restrictions Algorithm” (SRA, see Mekhnacha et al. [2006] et Mekhnacha et al. [2007] for details) is the main component of the symbolic part of ProBT®, and it achieves, when possible, dramatic symbolic simplification of the computation. It is an improvement of the generalized distributive law algorithm proposed by Aji and McEliece [2000].
- The second algorithm concerns the problem of numerically representing high-dimensional probability distributions. It aims to make the computational cost (memory use and computation time) more tractable while providing the capacity of generalization and the anytime property. This algorithm is based on a data structure called “MRBT” (for Multi-Resolution Binary Tree). A patent protects it [Bessière, 2002].
- The third algorithm is called “MCSEM” (for Monte Carlo Simultaneous Estimation and Optimization). It aims to solve the problem of maximizing a posteriori distributions for high-dimensional problems involving (in the general case) high-dimensional integrals (or sums). It solves this double integration/optimization problem using an adaptive genetic algorithm. The problem of integration is approached using a stochastic Monte Carlo method. The accuracy of this numerical estimation of integrals is controlled by the optimization process to reduce the computation time. It was first proposed by Mekhnacha [1999].
3.2 ProBT® library

The programming language part of ProBT® is a collection of classes, methods and functions in C++.

All the mathematical concepts required to define a Bayesian program have their equivalent in the ProBT® API, including variables, conjunction of variables, probability, conditional probability, decomposition, parametric forms, specifications, descriptions and questions. The resulting programs are direct translations of the mathematical models, encoded in computer language.

For instance, the Bayesian program corresponding to simple sensor fusion on $N$ sensory variables for robot localization defined in Fig 16 is translated to the following ProBT® program.

---

```cpp
#include <mu_sigma_from_k.h>

translations of the mathematical models, encoded in computer language.

For instance, the Bayesian program corresponding to simple sensor fusion on $N$ sensory variables for robot localization defined in Fig 16 is translated to the following ProBT® program.

```
\[ \text{Number of observations} = \frac{\text{argc} - 1}{2}; \]

```cpp
int number_of_observations = (argc - 1)/2;

//******************************/
Variables
******************************/
plRealType position(-10,10,100);
plRealType distance(0.0,10.0,50);
plSymbol X("X",position);
plArray Z("Z",distance,1,number_of_observations);

//******************************/
Parametric Forms
******************************/
plUniform PX(X);
vector<plExternalFunction>& my_functions(number_of_observations);
vector<plExternalFunction>& f_mu(number_of_observations);
vector<plExternalFunction>& f_sigma(number_of_observations);
plComputableObjectList PZ;
plValues Z_values(Z);

for(int i = 0; i < number_of_observations; i++) {
    my_functions[i] = new mu_sigma_from_k(atof(argv[i+1]));
    f_mu[i] = new plExternalFunction(X,my_functions[i],&mu_sigma_from_k::dist_mu);
    f_sigma[i] = new plExternalFunction(X,my_functions[i],&mu_sigma_from_k::dist_sigma);
    plCndNormal PZi(Z(i),X,*f_mu[i],*f_sigma[i]);
    PZ *= PZi;
    Z_values[Z(i)] = atof(argv[number_of_observations+i+1]);
}

//******************************/
Decomposition
******************************/
plJointDistribution jd(X"Z",PX,PZ);

//******************************/
Question
******************************/
plCndKernel Question;
plKernel InstQuestion, CompQuestion;
jd.ask(InstQuestion,X,Z);
Question_instantiate(InstQuestion,Z_values);
InstQuestion.compile(CompQuestion);
}
```

ProBT® is a commercial product sold by the ProBAYES12 company. It is also available free for research and teaching purposes (see www.Bayesian-Programming.org). Numerous examples and online documentation may be found on this last web site.

### 3.3 Bayesian programming vs Bayesian network

At first, the programming syntax presented in the previous section may seem less convenient than the graphical interface of standard Bayesian network software.

The absence of an evident human–machine interface is not an oversight but a choice.

This choice was made for four main reasons.

- We think that graphical representations impose supplementary constraints that issue neither from the rules of probability nor from the logic of the problem. For instance, the rules of probability allow us to specify a decomposition including a distribution with two or more variables on the left part

---

12 www.ProBAYES.com
of the conditioning mark (as, for example, \( P(X \land Y \mid \pi) \)). This is not possible in a Bayesian network graphical representation without introducing an intermediate variable.

- The algebraic notation used in BP is very convenient for expressing iteration or recurrences. This greatly simplifies the specification of models that include the same submodel duplicated several times, such as Bayesian filters or Hidden Markov Models (HMMs).

- Bayesian programs offer most of the facilities that allow classical programming to build complex software from simple bricks. Two salient instances are conditional constructions such as the “if-then-else” and “subroutine calls” that have both been presented in this chapter.

- The ProBT® library is intended to be included in other programs, not to be used on its own. Consequently, what is really important is to produce a piece of code that can be easily inserted in or combined with the host program.

We have proved that Bayesian programming is a generalization of Bayesian networks (see Bessière [2003]). As shown in Fig. 17, all Bayesian network models may be restated as Bayesian programs. The opposite is not true: some Bayesian programs may not be represented graphically.

**Fig. 17.** BP is a generalization of BN (reprinted from J. Diard’s PhD thesis [Diard, 2003]).

The rest of this book will provide numerous examples of the expressive power of Bayesian programming.
References


Part II

Robotics
The *CyCab*: Bayesian navigation on sensory–motor trajectories

Cédric Pradalier\(^1\) and Pierre Bessière\(^2\)

\(^1\) CSIRO ICT Centre  
\(^2\) CNRS - Grenoble Université

1 Introduction

Autonomous navigation of a mobile robot is a widely studied problem in the robotics community. Most robots designed for this task are equipped with onboard sensor(s) to perceive the external world (sonars, laser telemeters, camera). Two main approaches to autonomous navigation have been proposed: reactive navigation, where the robot uses only its current perceptions to move and explore without colliding (e.g. Arkin [1998] or Bonasso et al. [1995]), and servoed navigation, in which the robot is given a preplanned reference trajectory and uses some closed-loop control law to follow it (e.g. Laumond et al. [1989] or Lamiraux et al. [1999]). In servoed problems, two classes of approaches can again be separated: state-space tracking (e.g. Hermosillo et al. [2003b,a]) and perception-space tracking (e.g. Malis et al. [2001] or Chaumette [1994]).

State-space tracking implies two capabilities: first, to be given a reference trajectory in the state space, and second, to be able to localize the robot, also in the state space. Conversely, perception-space tracking implies that the trajectory is defined with respect to perception only, hence avoiding the need for global localization. A specific application of perception tracking is visual servoing, classically implemented as the convergence of the observed image to a fixed reference image.

In this chapter, we are specifically interested in the case of perceptual tracking of a sensory–motor trajectory (SMT) with a mobile robot. We assume that:

- the reference trajectory is defined as a sequence of observations perceived by onboard sensors while the robot is moving; and
- no localization system (either GPS or landmark-based) is available to perform tracking.

This situation is interesting for at least three reasons: first, because the trajectory is not defined with respect to a Cartesian frame, we can ignore the
complex task of global localization; second, such trajectories can be naturally
and easily learnt from examples; and third, this approach can be seen as a
hypothesis on how biological entities memorize and represent paths.

This chapter is organized as follows. Sections 2 and 3 introduce the notions
and definitions used in this article. The elementary modules required for safe
navigation on a sensory–motor trajectory are developed in Sections 4 to 7.
Finally, Section 8 presents some experiments on a simulated platform and
some results on our real platform: the CyCab.

2 Problem statement

2.1 General scenario

The application we consider in this paper has two stages.

- Learning an SMT: the robot is driven manually from a configuration $A$
to a configuration $B$. While it is driven, it records its perceptions from
  onboard sensors (e.g. a camera or a laser range finder) and the commands
  sent by the driver.
- Navigating on the SMT: we now assume that the robot starts in a con-
  figuration $C$ in the neighbourhood of the SMT. Our goal is then to reach
  configuration $B$ while following the SMT as closely as possible.

Figure 1 gives an intuitive representation of our objectives for a simulated
vehicle in a car park.

![Figure 1](image_url)

**Fig. 1.** Navigating on the SMT from $A$ to $B$: starting from $C$, the objective is to
follow the black track to reach $B$. 
2.2 Definitions

Definition 1. Sensory–motor trajectory (SMT): Given two sets $\mathcal{O}$ and $\mathcal{U}$ containing respectively the perceptions and the actions of the robot, we define a sensory–motor trajectory $T_{sm}$ as a function of time with values in the Cartesian product of the robot command space $\mathcal{U}$ and its observation space $\mathcal{O}$. Formally:

$$T_{sm} : [0, t_{\text{max}}] \rightarrow \mathcal{O} \times \mathcal{U} \quad t \mapsto (T_{sm}(t).Z, T_{sm}(t).U).$$

Definition 2. Temporal position: Consider an SMT $T_{sm}$. The temporal position $\tau^t$ is a time in $[0, t_{\text{max}}]$ such that, at time $t$, $T_{sm}(\tau^t).Z$ and $T_{sm}(\tau^t).U$ are considered respectively as reference observation and reference action by a robot following the SMT.

Definition 3. Difference of viewpoint: Let $Z_1$ be an observation received in configuration $C_1$, and let $Z_2$ be an observation received in $C_2$. We define the difference of viewpoint between $Z_1$ and $Z_2$ as the difference of configuration $C_2 - C_1$.

Definition 4. Tracking error: Consider an SMT $T_{sm}$ and an observation $Z^t$. The tracking error $\xi^t$ is the difference of viewpoint between $Z^t$ and the reference observation $T_{sm}(\tau^t).Z$.

2.3 Problems to solve

To navigate on a sensory–motor trajectory, our robot must be equipped with several core competencies:

1. it must be able to localize itself on the trajectory;
2. it must be able to control its movement to follow the trajectory while avoiding collisions with unexpected obstacles; and
3. it must be able to estimate online its confidence with respect to its localization hypotheses.

3 Outline of the approach

To solve these problems, we implement a set of specialized functions.

- Initialization: when the system starts, we only assume that the robot is in a neighbourhood of the SMT. Consequently, using the first observation $Z^0$, we must estimate the initial temporal location $\tau^0$ and the initial tracking error $\xi^0$.
- Localization: during its movement, the system must keep track of the temporal location $\tau^t$ – i.e. temporal localization – and of the tracking error $\xi^t$ – i.e. egocentric localization. To this end, the system compares executed commands and real observations with their reference counterparts.
• Command generation: computation of the vehicle controls for tracking the SMT.

<table>
<thead>
<tr>
<th>Name</th>
<th>Output</th>
<th>Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialization</td>
<td>$\tau^0, \xi^0$</td>
<td>$Z^0, T_{sm}$</td>
</tr>
<tr>
<td>Localization</td>
<td>$\tau^t$</td>
<td>$Z^t, U^{t-dt}, T_{sm}$</td>
</tr>
<tr>
<td>Relative</td>
<td>$\xi^t$</td>
<td>$Z^t, T_{sm}(\tau^t), Z$</td>
</tr>
<tr>
<td>Command</td>
<td>$U^t$</td>
<td>$\xi^t, T_{sm}(\tau^t), U$</td>
</tr>
<tr>
<td>Confidence</td>
<td>Conf $^t$</td>
<td>$\xi^t, Z^t, T_{sm}(\tau^t), Z, Conf^{t-dt}$</td>
</tr>
</tbody>
</table>

Table 1. Specification of the required functions for navigation on a SMT.

In this chapter, we will show how to implement all these basic competencies using Bayesian modelling. Bayesian reasoning often raises some fears when dealing with vehicular robots or heavy machinery: can we guarantee safety and performance when using a method based on probabilities? To answer this question, we want to stress the fact that using Bayesian modelling does not impose non-deterministic behaviours. Bayesian modelling is only a way to express knowledge about the robot, its sensors and its environment. From this knowledge, we can choose to take deterministic or probabilistic decisions.

4 Relative localization

4.1 Context and vehicle model

Fig. 2. CyCab robot and models used throughout this article.

Our general context is inspired by our experimental platform.
Vehicle

Our experimental platform is an autonomous electric vehicle (see Fig. 2.a) derived from a golf cab: the CyCab. This 300kg vehicle is capable of speeds up to 4m/s. The model we use represents the CyCab as a rectangular bounding box, in a darker stroke with a triangle inside in Fig. 2.b.

Environment

We conducted our experiments in the car park of our institute, among parked or manoeuvring cars and walking pedestrians. Moreover, we assumed that, in this environment, it is sometimes possible to perceive some landmarks. In practice, these landmarks are vertical cylinders covered with reflective sheets. We model this environment by a set of polygonal obstacles (straight lines on Fig. 2.b) and a set of punctual landmarks (scattered discs).

Sensors

The CyCab is equipped with a laser rangefinder, mounted on the vehicle’s front (see Fig. 2.a and b). Its laser beam sweeps the half-plane in front of the CyCab about 50 centimetres above the ground and returns distances to nearest obstacles and the associated intensity of the returning beam. Figure 2.c illustrates the sensor output in the specific situation depicted in Fig. 2.b. Furthermore, we implemented a landmark extractor that reliably processes the raw data to extract landmark positions in the sensor frame. In the sequel, we will ignore the raw data and consider that a perception is a set of landmark positions in the sensor frame. Such a perception is depicted in Fig. 2.c.

This context should not be seen as restricting the application field of our results but rather as an example extended throughout this chapter.

4.2 Design of a localization model

A Bayesian model for localization links robot localization and robot perception. Its key part is a sensor model, i.e. a model that can predict expected measurements knowing the robot’s localization. The first part of this section will describe such a model. We will then see how to use it actually to perform the localization.

When a map of the environment is available, it is defined in some fixed frame. The goal of localization is then to compute the robot’s position in this frame.

When a robot is following an SMT, we do not have such a map, but localization with respect to a fixed frame is not required. Because the robot is tracking a trajectory, it simply evaluates its tracking error $\xi^t$. It does this by comparing the current observation $Z^t$ and the reference observation $T_{sm}(\tau^t).Z$. 
To simplify notation, we will use $Z_{t \leftarrow ref} = T_{sm}(\tau^t).Z$. Furthermore, except when explicitly specified otherwise, we will consider all variables to refer to the current time $t$. Consequently, we will omit $t$ whenever possible.

### 4.3 Bayesian sensor model

The sensor model that we now consider is a probability distribution $P(Z \mid \xi \land Z_{ref})$. This distribution expresses our knowledge about the expected observation, knowing that the robot is observing $Z_{ref}$ with a difference of viewpoint $\xi$. If this expectation does not contain any uncertainty, then the probability distribution will be a Dirac distribution $^3$. Otherwise when we want to express greater uncertainty, the distribution will be flatter.

In the following, we will show the step-by-step construction of the sensor model that we use in our application. Recall the variables’ meanings.

- $\xi$: the difference of viewpoint between the current observation and the reference one, composed of the $(x, y)$ position in the plane and the heading $\theta$.
- $Z$, resp. $Z_{ref}$: an observation, resp. reference observation, as a set of observed landmark positions in the sensor frame: they can be described either with their polar coordinates $(\rho, \alpha)$, or with their Cartesian coordinates $(x, y)$.

**Basic model**

The simplest model that we can imagine occurs when only one landmark $L_1$ has been observed in $Z_{ref}$. In this case, the sensor model is usually defined as a Gaussian model ($G$) centred around the distances and bearings that we can expect knowing the difference of viewpoint.

$$P_t(\mid \xi \land L_1) = G(||(x, y) - L_1||, \sigma_\rho) \times G(|\arg((x, y) - L_1) - \theta, \sigma_\alpha)$$

This model is parameterized by the standard deviations of the Gaussian distributions $\sigma_\rho$ and $\sigma_\alpha$. Basically, they express the accuracy that we expect from our sensor: larger standard deviations indicate a less accurate sensor.

Using this model, we can now use the Bayesian program in Fig. 3 to answer the question $P(\xi \mid Z \land Z_{ref})$.

Figure 4 shows a reference observation, a real observation and the resulting distribution $P(\xi \mid Z \land Z_{ref})$ for values of $\xi$ in $[-5, 5] \times [-5, 5]$ (in meters). To represent this distribution fully, we would require a $4D$ graph. In the left-hand part of the figure, each arrow represents a value of $\xi = (x_\xi, y_\xi, \theta_\xi)$: it starts from $(x_\xi, y_\xi)$ and has the orientation $\theta_\xi$. The length of each arrow is proportional to the likelihood of the tracking error that it represents.

$^3$ A Dirac distribution is zero everywhere except for one value.
To help visualize the shape of the distributions, we also add the marginalized distribution $P(\xi_x \land \xi_y \mid Z \land Z_{ref})$ on the right-hand part of the figure.

Given one reference landmark and one observation, it is not possible to identify one single difference of point of view. As partly shown in Fig. 4, the set of possible location forms a fuzzy spiral in the 3D configuration space $(x, y, \theta)$. 
Taking into account unmodelled events

In our car park environment, with our sensor, it can happen that a vehicle’s light or number plate generates a sensor output similar to a landmark. Such sensor outputs, inconsistent with the sensor model, are often called “outliers” or “false positives”. In the context of probabilistic localization, it is essential to take these into account. Actually, our sensor model gives us both positions that are likely with regard to a given observation and those that are unlikely because of their inconsistency with the observation. Consequently, if not dealt with properly, an outlier will classify a whole part of the configuration space as extremely unlikely, including the correct position.

To express the fact that an observation might indeed be an outlier, we add a Boolean variable $F$ (as False). When $F = 1$, the current observation is assumed to be a false positive. In this case, we cannot expect any peculiar observation. This can be expressed using a uniform distribution. The probabilistic sensor model becomes:

$$P_2([Z = (\rho, \alpha)] | \xi \land F \land Z_{ref}) = \begin{cases} [F = 0] & \rightarrow P_1(Z | \xi \land Z_{ref}) \\ [F = 1] & \rightarrow \text{Uniform}(\rho) \text{Uniform}(\alpha). \end{cases}$$

In the Bayesian program, the joint distribution becomes:

$$P(Z \land \xi \land F \land Z_{ref}) = P(Z_{ref})P(\xi)P(F)P_2(Z | \xi \land F \land Z_{ref}). \quad (1)$$

The experimental calibration of $P(F)$ is a difficult problem because we introduced this term to account for unmodelled terms. Empirically, we chose $P([F = 1]) = 0.2$, i.e. we consider that 20% of the observations can be outliers.

Integrating landmark identification

We now consider the case where several landmarks, $Z_{ref} = \{L_1, \ldots, L_n\}$, were observed in the reference observation but only one landmark is found in the current observation. The problem is then to find which of the reference landmarks has been observed. To this end, we introduce a new probabilistic variable $M \in \{1, \ldots, n\}$ (as Matching). When $M = k$, we expect $Z$ to be an observation of $L_k$. For non-outlier observations, the probabilistic sensor model becomes:

$$P_3([Z = (\rho, \alpha)] | \xi \land [M = k] \land F \land Z_{ref}) = P_2(Z | \xi \land F \land L_k). \quad (2)$$

In the Bayesian program, the joint distribution becomes:

$$P(Z \land \xi \land M \land F \land Z_{ref}) = P(\xi)P(F)P(M | \xi \land Z_{ref} \land F) \times P(Z_{ref})P_3(Z | \xi \land M \land Z_{ref} \land F).$$

When the observation is not an outlier ($F = 0$), the term $P(M | \xi \land Z_{ref} \land F)$ provides a means of expressing some knowledge about sensor limitations such
as range or field of view. Knowing $\xi$ and $Z_{\text{ref}}$, a landmark $L_k$ may be out of range. In this case, we can say that $P([M = k] \mid \xi \land Z_{\text{ref}})$ should be small. Nevertheless, in general, we do not want to privilege some identifications and, consequently, we use a uniform distribution over $M$.

When the observation is an outlier ($F = 1$), the matching variable is no longer relevant, and $P(M \mid \xi \land Z_{\text{ref}} \land F)$ can be unspecified. This is made explicit in the complete expression of the localization question:

$$P(\xi \mid Z \land Z_{\text{ref}}) \propto P(\xi) \times \left[ (P(F = 0) \sum_{k=1}^{n} P(M = k)P(Z \mid \xi \land [F = 0] \land [M = k] \land Z_{\text{ref}}) \right]$$

$$+ P(F = 1)P(Z \mid \xi \land [F = 1] \land Z_{\text{ref}}).$$

Figure 5 shows a reference observation with two landmarks, a real observation with one landmark, and the resulting distribution $P(\xi \mid Z \land Z_{\text{ref}})$ as in Fig. 4. Without knowing the data association, there is no single localization hypothesis but rather two equiprobable spirals in the configuration space, corresponding to the two possible data associations. On the $P(\xi \land \xi_y \mid Z \land Z_{\text{ref}})$ graph, these two spirals are visible as two arcs of circles. We can notice an increase in the likelihood at their intersection. This is because for this position, observation $Z$ is equally supported by two distinct orientations.

The influence of unmodelled events is also visible on these figures. A configuration’s likelihood does not decrease to zero when considering configurations far from the main arcs of circles in the $P(\xi_x \land \xi_x \mid Z \land Z_{\text{ref}})$ graph: it is bounded below by the configuration likelihood that would result from the observation being a false positive.
Multiple landmark observations

When reference and real observations contain several landmarks, we must use a set of variables $Z_i$, $M_i$ and $F_i$ for each observed landmark. Then, each observation is described with the sensor model presented above. The resulting joint distribution is:

$$P(\xi Z_{1:p} \wedge M_{1:p} \wedge F_{1:p} \wedge Z_{\text{ref}}) = P(Z_{\text{ref}})P(\xi) \times \prod_{i=1}^{p} P(F_i)P(M_i \mid \xi \wedge Z_{\text{ref}} \wedge F_i)P(Z_i \mid \xi \wedge M_i \wedge Z_{\text{ref}} \wedge F_i).$$

![Diagram](image)

Fig. 6. Building a probabilistic localization: multiple landmarks model.

Figure 6 shows a two-landmarks reference observation, a three-landmarks real observation and the resulting distribution $P(\xi \mid Z \wedge Z_{\text{ref}})$ as in Fig. 4. After fusing the data from all three observations, a set of robot configurations around $\left(1.5, -1, 20^\circ\right)$ is much more likely than any others. Nevertheless, the configuration spirals that would result from one of the observations being a false positive are still visible, in lighter gray on the $P(\xi_x \wedge \xi_y \mid Z \wedge Z_{\text{ref}})$ graph. If all the observations are false positives, then all tracking errors are equiprobable. The localization hypotheses corresponding to this situation appear in a darker shade on the graph.

It is also interesting to notice that as a false positive, observation $Z_3$ has little influence on the final distribution. It only introduces a secondary hypothesis, visible as a small set of arrows and a slightly lighter patch behind the main peak on the distribution graph. This hypothesis corresponds to the most likely localization, should $Z_2$ be an outlier.
Interest of Bayesian localization

The main interest of the model above is its ability to capture all the localization hypotheses in one computation. After this computation, all the uncertainties on the sensor measure, on the data association and on the presence of outliers are converted into the localization space. In this space, the uncertainty is much easier to handle, especially in the context of some Bayesian filtering. In our application, for instance, the localization is performed by integrating this sensor model in a particle filter.

Weakness of alternative hypotheses

Using Fig. 6, we have stressed the Bayesian model’s ability to extract all the localization hypotheses in one computation. Nevertheless, we should notice that the likelihood of alternative hypotheses is extremely low: several orders of magnitude below the main hypothesis. There are several reasons to try to avoid this situation:

- this low likelihood is not because of the low probability of false-positives but because of the size of the observation space (see below);
- within the context of Bayesian filtering, configurations with such a low likelihood will certainly be discarded; and
- finally, we can wonder if it is really relevant to waste so much computation power to deal with such a small part of the likelihood space.

If we develop the localization equation, we have the following.

\[ P(\xi \mid Z \land Z_{ref}) \propto \prod_{i=1}^{p} \begin{cases} 0.8 \times P([F_i = 0]) \times P(Z_i \mid \xi \land Z_{ref} \land [F_i = 0]) & \text{Gaussian} \\ + 0.2 \times P([F_i = 1]) \times P(Z_i \mid \xi \land Z_{ref} \land [F_i = 1]) & \text{Uniform} \end{cases} \]

Alternative localization hypotheses occur when one of the observations, assume the \( k \)th, is inconsistent with a given position. The Gaussian part of the \( k \)th sum is then close to zero, and only the uniform part remains. Unfortunately, if the observation space is large, the uniform value on this space is very small, and consequently the corresponding position likelihood becomes very small as well, with only a small influence of the false positive distribution.

4.4 Diagnosis model

To build a sensor model that gives a fairer influence to the false-positive variable, we chose to describe our sensor model with a diagnosis variable. Instead of expressing the expected observation knowing the position, we want to use a diagnosis variable to evaluate the consistency between an observation and a position. In practice, this consistency is represented by a Boolean variable \( I \).
In the single-landmark/single-observation model with false positives, the consistency model is as follows.

\[
P(\mathcal{Z} \mid Z_{\text{ref}} = L_1) \wedge Z = (\rho, \alpha) \wedge \xi \wedge \mathcal{F} = \begin{cases} 
[F = 0] & \exp(-\frac{1}{2} \frac{(\rho - \|\xi - x\|^2 - L_1\|^2)}{\sigma_{\rho}^2}) \times \exp(-\frac{1}{2} \frac{(\alpha - \text{arg}((\xi, \xi) - L_1))^2)}{\sigma_{\alpha}^2}) \\
[F = 1] & 0.5 
\end{cases}
\]  

(3)

In this equation, \(\sigma_{\rho}\) and \(\sigma_{\alpha}\) act as standard-deviation parameters in Gaussian distributions, expressing the expected accuracy of the sensor. In the true positive case, with \(F = 0\), as an observation approaches the expected observation, the consistency likelihood will approach 1.

The false-positive case is the interesting part of this model. As in the previous model, we cannot expect any particular observation in this case, and consequently, we choose a consistency likelihood with as much a-priori knowledge as possible: 0.5. It may be argued that 0.5 is a high likelihood for the consistency of any conjunction of observation and position. The key advantage of this value is that it does not depend on the size of the observation space.

Figure 7 shows the localization results with a diagnosis model, in the same situation as that shown in Fig. 6. The global shape of the resulting distribution is similar to that in Fig. 6. However, we can observe that alternative localization hypotheses have a much stronger influence: here we do not require a
logarithmic colour scale to be able to see the alternative hypotheses. In the context of a Bayesian filtering scheme, this sensor model will consequently be more robust to false positives.

4.5 Remarks on Bayesian localization

We have shown how to build a Bayesian sensor model for localization. Such a model can be used to project all the information available in the observations into the localization space:

- all the data association hypotheses;
- the localization hypotheses accounting for possible false positives;
- the geometrical consistency of the observations;
- the uncertainty on all the above points is now converted into an uncertainty in the localization space, which is much easier to handle, especially in a filtering context.

5 System initialization

The first application of our sensor model is the initial localization of the robot in the SMT. Let us recall our objectives: we assume that we know an SMT going from A to B and, at system initialization, the robot is at an unknown position C in the neighbourhood of the trajectory.

Using our sensor model, we will be able to estimate the temporal position $\tau^0$ corresponding to C. Formally, we want to compute the distribution $P(\tau^0 | Z^0 \mathcal{T}_{sm})$, that is, the distribution on $\tau^0$ knowing the initial observation $Z^0$ (made at C) and the SMT $\mathcal{T}_{sm}$. To this end, we define the Bayesian program presented in Fig. 8.

Two points must be considered in this program. First, assuming that configuration C is in the neighbourhood of the SMT implies that the tracking error $\xi^0$ is small. To express this knowledge formally, we define $P(\xi^0)$ as a Gaussian distribution, centred on zero. The covariance of this distribution quantifies the “neighbourhood” notion.

Second, we use the Bayesian sensor model defined in the previous sections to express the distribution $P(Z^0 | \tau^0 \xi^0 \mathcal{T}_{sm})$. This is achieved with $Z^0_{ref} = \mathcal{T}_{sm}(\tau^0)$.Z and:

$$P(Z^0 | \tau^0 \xi^0 \mathcal{T}_{sm}) = P(Z^0 | Z^0_{ref} \xi^0).$$

4 Estimation of $P(\tau^0 | Z^0 \mathcal{T}_{sm})$

Using the Bayesian program shown in Fig. 8, we can compute a numerical approximation of $P(\tau^0 | Z^0 \mathcal{T}_{sm})$. Figure 9 shows a typical result of this distribution.
From this estimated distribution (called $\hat{P}$ below), we must extract a single value $\hat{\tau}^0$ for $\tau^0$. Depending on the shape of the distribution, the complexity of this extraction can range from easy to difficult or even impossible. Several cases are possible.

1. $\hat{P}$ is unimodal and strongly peaked. In this case, there is no ambiguity, and $\hat{\tau}^0$ is the mode of the distribution.

2. $\hat{P}$ is unimodal but widely spread (e.g. a Gaussian with a large standard deviation). Then, the expectation $E[\tau^0]$ may be used as $\hat{\tau}^0$. This situation is often caused by a pause in the reference trajectory, resulting in several successive similar observations.

3. $\hat{P}$ has several strong peaks. This is the consequence of some perceptual aliasing in the nominal trajectory. In this case, we must choose one of the modes. This choice may be based on some rules on the distribution (peak width, peak height, entropy,...) or from some active perception manoeuvre.

4. $\hat{P}$ has multiple wide modes or is close to a uniform distribution. This happens when the system cannot find observations in $T_{sm}$ close to the current observation. This is probably the result of an initial position too far from the neighbourhood of $T_{sm}$. A failure report should be generated.

Discussion

One point should be noted here. Even when $\hat{P}$ has a strong peak, there is no guarantee that we are indeed observing some part of $T_{sm}$. Nevertheless, without other information, we believe that we must start moving as if we
were confident with respect to our first estimation, while keeping in mind that future observations may contradict this estimation. This will be further discussed in Section 6.

5.1 Tracking

Once the system has computed the initial localization, it can start navigating on the SMT: it can simply track its temporal localization and its tracking error, and rely on a well-tuned controller (this will be discussed in Section 7).

The goal of the tracking is to maintain an estimate of $\tau^t$ and $\xi^t$ according to the sequence of observations $Z^t = Z^{0:t}$ and the sequence of controls applied to the system $U^t = U^{0:t}$. Expressed this way, this goal is exactly the objective of a Bayesian filter.

$$P(\tau^t | Z^t \land T_{sm}) \propto P(Z^t | \tau^t \land T_{sm}) \times \int \int_{\xi^t_{t-1} \land \tau^t_{t-1}} P(\xi^t_{t-1} \land \tau^t_{t-1} | Z^{t-1} \land U^{t-1}) P(\xi^t \land \tau^t | \xi^t_{t-1} \land \tau^t_{t-1} \land U^t) \, d\xi \, d\tau$$

The practical computation of the above filter in real time is not possible for trajectories longer than a couple of seconds. Some simplifications are required:
• $\tau^t$ and $\xi^t$ are estimated independently;
• only the most likely $\tau^t$ is used in the computations; and
• the Bayesian filter estimating $\xi^t$ is implemented using the well-known particle filter (also called condensation filter), assuming that $\tau^t$ is perfectly known.

6 Self-confidence estimation

As shown in Section 5, our trajectory tracking system is based on a first estimation of the initial temporal position. Phenomena such as perceptual aliasing\(^4\) mean that the system cannot guarantee that this initial estimation is correct. However, the robot must start moving, gathering information while following its sensory–motor trajectory. We use information gained during this movement to track a variable that expresses the confidence of the system with respect to its previous assumptions and its current localization.

Formally, variable $\text{Conf}^t \in \{0, 1\}$ will represent system self-confidence at time $t$: when $P(\text{Conf}^t = 1) = 1$, the system is fully confident in its localization (this should mean that it has collected a great deal of evidence); conversely, $P(\text{Conf}^t = 1) = 0$ expresses quasi-certainty that some failure has occurred, such as a wrong initial localization or an environment change.

In many problems where state estimation is involved, model comparison is used to diagnose system state (see Murphy [1998] or Lerner et al. [2000] for instances). In the remainder of this section, we will show how model comparison can be used to maintain an estimate of the system self-confidence.

6.1 Model comparison

Principle

Let us assume that we work with a variable $A$ that can be evaluated with two distinct models. We can build the following joint distribution.

$$P(A \text{ Model}) = P(\text{Model})P(A | \text{Model})$$

$P(\text{Model})$ expresses our prior knowledge about which model is the best, and $P(A | \text{Model})$ evaluates the probability distribution on $A$ knowing which model is used. We can then use Bayes’ rule to compute $P(\text{Model} | [A = a])$, i.e. given a real observation $a$ of $A$, what is the model that best explains $A = a$?

\(^4\) Environment perceptual ambiguity.
Application

Self-confidence, as defined above, can be used as a switch between two models: one that expresses which observation can be expected given full confidence in temporal localization, and one that expresses full distrust.

Formally, we define the probability distribution of observation $Z^t$ according to the system’s self-confidence: $P(Z^t \mid \text{Conf}^t)$. If $\text{Conf}^t = 0$, because the system does know that it knows nothing about its localization, $P(Z^t \mid [\text{Conf}^t = 0])$ is a “minimal knowledge” uniform distribution, otherwise, $P(Z^t \mid [\text{Conf}^t = 1])$ is set to the complete sensor model $P(Z^t \mid \mathcal{T}_{sm})$ as defined in sec. 4.3.

Tracking

Like previous tracking approaches in this chapter, self-confidence tracking will be implemented with a Markovian Bayesian filter. $P(Z^t \mid \text{Conf}^t)$, as defined above, gives us an observation model, so we simply define a transition model $P(\text{Conf}^t \mid \text{Conf}^{t-1})$ to obtain:

$$P(\text{Conf}^t \mid Z^t) \propto P(Z^t \mid \text{Conf}^t) \sum_{\text{Conf}^{t-1}} P(\text{Conf}^t \mid \text{Conf}^{t-1})P(\text{Conf}^{t-1}).$$  \hspace{1cm} (5)

$$\begin{array}{c|cc}
\text{Conf}^t & 0 & 1 \\
\hline
0 & 1-\lambda & \delta \\
1 & \lambda & 1-\delta \\
\end{array}$$  \hspace{1cm} (6)

$P(\text{Conf}^t \mid \text{Conf}^{t-1})$ is defined by eq. 6. Parameters $\delta$ and $\lambda$ are called respectively the doubting rate and the trusting rate, because they express the proportion of confidence, going respectively from 1 to 0 and from 0 to 1, between two time steps.

6.2 Using innovation

Self-confidence tracking implemented as described above works well, except for an unwanted behaviour that appears when the robot does not move: if the current observation is well supported by the confident model, self-confidence rapidly converges to 1. This behaviour is not satisfying because the confidence changes without obtaining new information.

Because we are convinced that self-confidence should only increase when the system is able to predict a challenging observation, we designed a slightly different model that can implement the desired behaviour.
Observation model

Instead of using confidence as the model switch, we introduce a new switch variable \( \text{Mode} \in \{0, 1, 2\} \). From this variable, we build an observation model \( P(Z^t | \text{Mode}) \) such that:

- \( P(Z^t | [\text{Mode} = 0]) \) is a uniform distribution (equiv. to \( P(Z^t | [\text{Conf}^t = 0] \) in previous model);
- \( P(Z^t | [\text{Mode} = 1]) = P(Z^t | Z^{t-1} \land U^{t-1}) \) expresses the expected observation knowing only the last observation and the displacement; and
- \( P(Z^t | [\text{Mode} = 2]) = P(Z^t | \tau^t \land T_{sm}) \) expresses the expected observation knowing the maximum information: sensory–motor trajectory, temporal position and tracking error distribution.

From this model and a uniform prior \( P(\text{Mode}) \), we define a joint distribution: \( P(Z^t \land \text{Mode}) = P(\text{Mode})P(Z^t | \text{Mode}) \) from which we can compute \( P(\text{Mode} | Z^t) \) using Bayes’ rule.

Confidence evolution model

For \( i \in \{0, 1, 2\} \), let us call \( M_i = P([\text{Mode} = i] | Z^t) \), and \( Q_i = \frac{M_i}{M_0} \). Knowing \( Q_1 \) and \( Q_2 \), we can predict how confidence should evolve, and then design a probabilistic model that implements this behaviour.

- If \( Q_2 < 1 \), observation prediction knowing the sensory–motor trajectory and temporal position is worse than that without prior knowledge. As in Section 6.1, confidence should decrease in this case: smaller values of \( Q_2 \) relative to 1 indicate bigger decreases.
- If \( Q_1 \geq Q_2 \geq 1 \), observation prediction is better knowing only the last observation than with maximum knowledge. This means that the current observation does not reflect any innovation with respect to the previous one. As there is no reason to change confidence without new evidence or counter-evidence, confidence should stay constant.
- If \( Q_2 > Q_1 \geq 1 \), the current observation not only is better predicted by the maximum knowledge observation model but also contains innovation with respect to previous observations. This ability to predict innovative observation should increase self-confidence. Furthermore, a bigger difference between \( Q_2 \) and \( Q_1 \) means that the current observation was very difficult to predict knowing only previous observations. Thus, larger values of \( Q_2 - Q_1 \), should increase confidence more.

To implement this behaviour, we use the Bayesian program defined in Fig. 10. This program is based on a definition of \( P(\text{Conf}^t | \text{Conf}^{t-1} Q_1 Q_2) \) similar to equation 6, except that the doubling rate \( \delta \) and trusting rate \( \lambda \) are now functions of \( Q_1 \) and \( Q_2 \).
Bayesian navigation of a car-like robot

\[
\delta(Q_1, Q_2) = \begin{cases} 
0 & \text{if } Q_2 \geq 1 \\
\min(1, k_\delta(1 - Q_2)) & \text{otherwise}
\end{cases}
\quad (7)
\]

\[
\lambda(Q_1, Q_2) = \begin{cases} 
0 & \text{if } Q_1 \geq Q_2 \\
\min(1, k_\lambda(Q_2 - Q_1)) & \text{otherwise}
\end{cases}
\quad (8)
\]

<table>
<thead>
<tr>
<th>Relevant Variables:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conf(^t) : Confidence at time (t)</td>
</tr>
<tr>
<td>Conf(^{t-1}) : Confidence at time (t - 1)</td>
</tr>
<tr>
<td>(Q_1, Q_2) : Observations, as def. in 6.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Decomposition:</th>
</tr>
</thead>
</table>
| \(P(\text{Conf}^t \land \text{Conf}^{t-1} \land Q_1 \land Q_2) = \)
| \(P(Q_1 \land Q_2) P(\text{Conf}^{t-1})\) |
| \(P(\text{Conf}^t \mid \text{Conf}^{t-1} \land Q_1 \land Q_2)\) |

<table>
<thead>
<tr>
<th>Parametric Forms:</th>
</tr>
</thead>
<tbody>
<tr>
<td>(P(Q_1 \land Q_2)) : Undefined</td>
</tr>
<tr>
<td>(P(\text{Conf}^{t-1})) : From previous iteration</td>
</tr>
<tr>
<td>(P(\text{Conf}^t \mid \text{Conf}^{t-1} \land Q_1 \land Q_2)) : See sec. 6.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Identification:</th>
</tr>
</thead>
<tbody>
<tr>
<td>(k_\delta) and (k_\lambda) are adjusted empirically.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Question:</th>
</tr>
</thead>
<tbody>
<tr>
<td>(P(\text{Conf}^t \mid Q_1 \land Q_2))</td>
</tr>
</tbody>
</table>

Fig. 10. Bayesian program for self-confidence tracking

An illustration of the self-confidence estimator’s results will be discussed with Figs. 12, 13 and 15.

7 Movement

7.1 Trajectory following

Navigation on an SMT requires a trajectory-following controller. Knowing a temporal position estimate \(\tau^t\), we can extract reference controls \(U_{\text{ref}}(t) = T_{sm}(\tau^t)U\). Then, with reference controls \(U_{\text{ref}}(t)\) and configuration error \(\xi^t\), we can apply a well-tuned control law given by control theory so that the robot can accurately replay its sensory–motor trajectory.

As we wanted to design a fully Bayesian application, we used inspiration from fuzzy logic control(Klein [1999], Fraichard and Garnier [2000]) to build a probabilistic control law, expressed as a Bayesian data fusion problem: \(P(U \mid \xi \land U_{\text{ref}})\) being expressed as a Bayesian fusion of \(P(U \mid U_{\text{ref}})\) and \(P(U \mid \xi)\). This controller will not be developed in this chapter, but we refer the interested reader to Pradalier et al. [2005], Pradalier et al. [2003] and Pradalier [2003] for details of the implementation.
7.2 Obstacle avoidance

Because our navigation system was designed to work in a moderately dynamic environment, we use controls computed for trajectory following as inputs in our obstacle avoidance module.

This module was presented in Pradalier et al. [2005] and Pradalier [2003]). Its principle is similar to such methods as Dynamic Window (Fox et al. [1997]) and Ego-Kinematic Space (Minguez et al. [2002]). Its specific advantage is mainly its expression as a Bayesian inference problem, making it particularly well suited for integration in this book’s framework.

Basically, it takes as inputs data from the proximity sensors and the desired commands decided by an upper-level module (such as the trajectory following described above) and uses them to compute the commands actually applied to the robot. These commands are designed to follow the desired commands as much as possible while preserving security. We refer readers to the citations above for more details about this module.

8 Experimental results

8.1 Software architecture

Our complete application architecture is presented in Fig. 11. Each Bayesian class, represented by a big rounded box, expresses a distribution $P(\text{outputs} | \text{inputs})$ from which a unique value is synthesized, to reduce complexity and
to achieve real-time operation. Using a multi-threaded implementation, and optimized Bayesian inference software, state tracking and controls generation can be done at about 50Hz.

Notice that the output of the confidence estimation module is not connected to any other module. It is not yet clear how this information can be integrated in the control. One possibility is to make the maximum speed decrease with decreasing confidence, to make the robot more cautious when its self-confidence is low. Smarter integration of the confidence estimation within the navigation process is a subject for further research. The confidence value is currently only used as a “health status” in a human interface display.

8.2 Results on simulated platform

![Fig. 12. Successful initialization and replay.]

Results presented in Figures 12 and 13 are computed on a simulated Cy-Cab, equipped with a simulated sensor, which tries to mimic as closely as possible the behaviour of the real sensor, particularly its limited field of view (180 degrees).

Both replays are made with respect to the same sensory–motor trajectory in the same environment (Figs. 12 and 13, upper left). The only difference is the initial orientation. In both situations, the initial position is a point located 1.5 m from the middle of the trajectory while the orientations differ by 180 degrees (filled triangle in Fig. 12 and 13, lower left).
Fig. 13. Failed initialization and replay with failure evolution.

From the initial perception, an initial temporal localization is computed (Figs. 12 and 13, upper right). This estimation is quasi-certain in Fig. 12, but there are more ambiguities in Fig. 13, in which the most probable $\tau(0)$ is the one with the least unlikely match with observation.

In both cases, replay is started from the initial temporal localization, with inhibited obstacle avoidance. In Fig. 12, the initial position was quite close to the nominal trajectory (1.5 m, 0.05 radian), so trajectory tracking rapidly brings the robot to its nominal trajectory (lower left), and confidence rises to 0.9: quasi-full confidence that localization was good and replay successful. Conversely, in Fig. 12, the initial position was far from the reference ($\pi$ radians), so the initial temporal localization was wrong. In this case, trajectory tracking tries to follow the hypothesized trajectory, but evidence against the current hypothesis accumulates and confidence rapidly decreases toward zero.

Figure 14 gives an overview of the robustness of the approach in a more realistic environment: a simulated parking area (buildings hashed, cars in dark gray, landmarks as white circles). The left-hand side of the figure displays the reference trajectory with the robot’s starting position (in light gray) and the initial environment. Before starting the replay of the trajectory, we remove 10 of the 34 landmarks, and we move one of the cars to make it interfere with the reference trajectory. In the right-hand side of the figure, the reference trajectory is shown in bold black whereas the executed one is shown in dashes. Tracking is performed accurately while there is no risk of collision, even with strong curvature and missing landmarks. Deformations of the trajectory oc-
8.3 Results on real platform

Figures 15 and 16 illustrate the results of the replayed trajectory on our real car-like robot. Movies of the obtained results may also be found at the following URLs: http://www.bayesian-programming.org/videoB1Ch3-1.html and http://www.bayesian-programming.org/videoB1Ch3-2.html. Our landmark detector is used as sensory input, and obstacle avoidance is used to check whether the proposed controls are safe. Trajectory replay is executed accurately at 2 m/s, moving at a few tenths of a centimetre from parked cars.

The crossing of a pedestrian, in Fig. 16, is handled gracefully by the application: the trajectory is deformed until the only safe action is to give way to the pedestrian.

9 Conclusions

We have shown how behavioural navigation on a sensory–motor trajectory can be expressed as a fully Bayesian application: temporal and spatial localization, control generation, obstacle avoidance and failure diagnosis were successfully implemented and integrated on a simulated robot and on a car-like autonomous vehicle in the car park of our institute.
Fig. 15. Sensory–motor replay on a car-like vehicle: experiment 1.
It is important to note that we used probabilistic reasoning to design a behaviour that is computationally efficient, predictable and safe for the vehicle and its environment.

References


1 Introduction

Perception of and reasoning about dynamic environments is pertinent for mobile robotics and still constitutes one of the major challenges. To work in these environments, the mobile robot must perceive the environment with sensors; measurements are uncertain and normally treated within the estimation framework. Such an approach enables the mobile robot to model the dynamic environment and follow the evolution of its environment. With an internal representation of the environment, the robot is thus able to perform reasoning and make predictions to accomplish its tasks successfully. Systems for tracking the evolution of the environment have traditionally been a major component in robotics. Industries are now beginning to express interest in such technologies. One particular example is the application within the automotive industry for adaptive cruise control [Coué et al., 2002], where the challenge is to reduce road accidents by using better collision detection systems. The major requirement of such a system is a robust tracking system. Most of the existing target-tracking algorithms use an object-based representation of the environment. However, these existing techniques must explicitly consider data association and occlusion. In view of these problems, a grid-based framework, the Bayesian occupancy filter (BOF) [Coué et al., 2002, 2003], has been proposed.

1.1 Motivation

In classical tracking methodology [Bar-Shalom and Fortman, 1988], the problem of data association and state estimation are major problems to be addressed. The two problems are highly coupled, and an error in either component leads to erroneous outputs. The BOF makes it possible to decompose this highly coupled relationship by avoiding the data association problem, in the sense that the data association is handled at a higher level of abstraction.
In the BOF model, concepts such as objects or tracks do not exist; they are replaced by more useful properties such as occupancy or risk, which are directly estimated for each cell of the grid using both sensor observations and some prior knowledge.

It might seem strange to have no object representations when objects obviously exist in real life environments. However, an object-based representation is not required for all applications. Where object-based representations are not pertinent, we argue that it is more useful to work with a more descriptive, richer sensory representation rather than constructing object-based representations with their complications in data association. For example, to calculate the risk of collision for a mobile robot, the only properties required are the probability distribution on occupancy and velocities for each cell in the grid. Variables such as the number of objects are inconsequential in this respect.

This model is especially useful when there is a need to fuse information from several sensors. In standard methods for sensor fusion in tracking applications, the problem of track-to-track association arises where each sensor contains its own local information. Under the standard tracking framework with multiple sensors, the problem of data association will be further complicated: as well as the data association between two consecutive time instances from the same sensor, the association of tracks (or targets) between the different sensors must be taken into account as well.

In contrast, the grid-based BOF will not encounter such a problem. A grid-based representation provides a conducive framework for performing sensor fusion [Moravec, 1988]. Different sensor models can be specified to match the different characteristics of the different sensors, facilitating efficient fusion in the grids. The absence of an object-based representation allows easier fusing of low-level descriptive sensory information onto the grids without requiring data association.

Uncertainty characteristics of the different sensors are specified in the sensor models. This uncertainty is explicitly represented in the BOF grids in the form of occupancy probabilities. Various approaches using the probabilistic reasoning paradigm, which is becoming a key paradigm in robotics, have already been successfully used to address several robotic problems, such as CAD modelling [Mekhnacha et al., 2001] and simultaneous map building and localization (SLAM) [Thrun, 1998, Kaelbling et al., 1998, Arras et al., 2001].

In modelling the environment with BOF grids, the object model problem is nonexistent because there are only cells representing the state of the environment at a certain position and time, and each sensor measurement changes the state of each cell. Different kinds of objects produce different kinds of measures, but this is handled naturally by the cell space discretization.

Another advantage of BOF grids is their rich representation of dynamic environments. This information includes the description of occupied and hidden areas (i.e. areas of the environment that are temporarily hidden to the sensors by an obstacle). The dynamics of the environment and its robustness relative to object occlusions are addressed using a novel two-step mechanism.
that permits taking the sensor observation history and the temporal consistency of the scene into account. This mechanism estimates, at each time step, the state of the occupancy grid by combining a prediction step (history) and an estimation step (incorporating new measurements). This approach is derived from the Bayesian filter approach [Jazwinsky, 1970], which explains why the filter is called the Bayesian occupancy filter (BOF).

The five main motivations in the proposed BOF approach are as follows.

• Taking uncertainty into account explicitly, which is inherent in any model of a real phenomenon. The uncertainty is represented explicitly in the occupancy grids.

• Avoiding the “data association problem” in the sense that data association is to be handled at a higher level of abstraction. The data association problem is to associate an object $o_t$ at time $t$ with $o_{t+1}$ at time $t+1$. Current methods for resolving this problem often do not perform satisfactorily under complex scenarios, i.e. scenarios involving numerous appearances, disappearances and occlusions of several rapidly manoeuvring targets. The concept of objects is nonexistent in the BOF and hence avoids the problem of data association from the classical tracking point of view.

• Avoiding the object model problem, that is, avoiding the need to make assumptions about the shape or size of the object. It is complex to define what the sensor could measure without a good representation of the object. In particular, a big object may give multiple detections whereas a small object may give just one. In both cases, there is only one object, and that lack of coherence causes multiple-target tracking systems, in most cases, to work properly with only one kind of target.

• An increased robustness of the system relative to object occlusions, appearances and disappearances by exploiting at any instant all relevant information on the environment perceived by the mobile robot. This information includes the description of occupied and hidden areas (i.e. areas of the environment that are temporarily hidden to the sensors by an obstacle).

• A method that could be implemented later on dedicated hardware, to obtain both high performance and decreased cost of the final system.

1.2 Objectives of the BOF

We claim that in the BOF approach, the five previous objectives are met as follows.

• Uncertainty is taken into account explicitly, thanks to the probabilistic reasoning paradigm, which is becoming a key paradigm in robotics.

• The data association problem is postponed by reasoning on a probabilistic grid representation of the dynamic environment. In such a model, concepts such as objects or tracks are not needed.

• The object model problem is nonexistent because there are only cells in the environment state, and each sensor measurement changes the state of each
cell. The different kinds of measures produced by different kinds of object are handled naturally by the cell space discretization.

- The dynamics of the environment and its robustness relative to object occlusions are addressed using a novel two-step mechanism that permits taking the sensor observation history and the temporal consistency of the scene into account.
- The Bayesian occupancy filter has been designed to be highly parallelized. A hardware implementation on a dedicated chip is possible, which will lead to an efficient representation of the environment of a mobile robot.

This chapter presents the concepts behind BOF and its mathematical formulation, and shows some of its applications.

- Section 2 introduces the basic concepts behind the BOF.
- Section 2.1 introduces Bayesian filtering in the 4D occupancy grid framework [Coué et al., 2006].
- Section 2.2 describes an alternative formulation for filtering in the 2D occupancy grid framework [Tay et al., 2007].
- Section 3 shows several applications of the BOF.
- Section 4 concludes this chapter.

2 Bayesian occupation filtering

The consideration of sensor observation history enables robust estimations in changing environments (i.e. it allows processing of temporary objects, occlusions and detection problems). Our approach for solving this problem is to make use of an appropriate Bayesian filtering technique called the Bayesian occupancy filter (BOF).

Bayesian filters Jazwinsky [1970] address the general problem of estimating the state sequence \( x_k, k \in \mathbb{N} \) of a system given by:

\[
x_k = f_k(x_{k-1}, u_{k-1}, w_k),
\]

where \( f_k \) is a possibly nonlinear transition function, \( u_{k-1} \) is a “control” variable (e.g. speed or acceleration) for the sensor that allows it to estimate its own movement between time \( k - 1 \) and time \( k \), and \( w_k \) is the process noise. This equation describes a Markov process of order one.

Let \( z^k \) be the sensor observation of the system at time \( k \). The objective of the filtering is to estimate recursively \( x^k \) from the sensor measurements:

\[
z^k = h^k(x^k, v^k)
\]

where \( h^k \) is a possibly nonlinear function and \( v^k \) is the measurement noise. This function models the uncertainty of the measurement \( z^k \) of the system’s state \( x^k \).
In other words, the goal of the filtering is to estimate recursively the probability distribution $P(X^k \mid Z^k)$, known as the posterior distribution. In general, this estimation is done in two stages: prediction and estimation. The goal of the prediction stage is to compute an a priori estimate of the target’s state known as the prior distribution. The goal of the estimation stage is to compute the posterior distribution, using this a priori estimate and the current measurement of the sensor.

Exact solutions to this recursive propagation of the posterior density do exist in a restrictive set of cases. In particular, the Kalman filter [Kalman, 1960, Welch and Bishop] is an optimal solution when the functions $f^k$ and $h^k$ are linear and the noise values $w^k$ and $v^k$ are Gaussian. In general, however, solutions cannot be determined analytically, and an approximate solution must be computed.

In this case, the state of the system is given by the occupancy state of each cell of the grid, and the required conditions for being able to apply an exact solution such as the Kalman filter are not always verified. Moreover, the particular structure of the model (occupancy grid) and the real-time constraint imposed on most robotic applications lead to the development of the concept of the Bayesian occupancy filter. This filter estimates the occupancy state in two steps, as depicted in Fig. 1.

In this section, two different formulations of the BOF will be introduced. The first represents the state space by a 4-dimensional grid, in which the occupancy of each cell represents the joint space of 2D position and 2D velocity. The estimation of occupancy and velocity in this 4D space are described in Section 2.1.

The second formulation of the BOF represents the state space by a 2-dimensional occupancy grid. Each cell of the grid is associated with a probability distribution on the velocity of the occupancy associated with the cell. The differences between the two formulations are subtle. Essentially, the 4D formulation permits overlapping objects with different velocities whereas the...
2D formulation does not allow for overlapping objects. The estimation on velocity and occupancy in this 2D grid are described in Section 2.2.

2.1 The 4D Bayesian occupation filter

The 4-dimensional BOF takes the form of a gridded histogram with two dimensions representing positions in 2D Cartesian coordinates and the other two dimensions representing the orthogonal components of the 2-dimensional velocities of the cells. As explained previously in Section 2, the BOF consists of a prediction step and an estimation step in the spirit of Bayesian filtering.

Based on this approach, the evolution of the BOF at time $k$ occurs in two steps:

1. the prediction step makes use of both the result of the estimation step at time $k-1$ and a dynamic model to compute an a priori estimate of the grid; and
2. the estimation step makes use of both this prediction result and the sensor observations at time $k$ to compute the grid values.

The next two subsections will explain the prediction and estimation steps of the 4D BOF respectively.

Estimation in the 4D BOF

The estimation step consists of estimating the occupancy probability of each cell of the grid, using the last set of sensor observations. These observations represent preprocessed information given by a sensor. At each time step, the sensor is able to return a list of detected objects, along with their associated positions and velocities in the sensor reference frame. In practice, this set of observations could also contain two types of false measurements: false alarms (i.e. when the sensor detects a nonexistent object) and missed detections (i.e. when the sensor does not detect an existing object).

Solving the static estimation problem can be done by building a Bayesian program. The relevant variables and decomposition are as follows.

- $C^k$: The cell itself at time $k$; this variable is 4-dimensional and represents a position and a speed relative to the vehicle.
- $E^k_C$: The state of the cell $C$ at time $k$; whether it is occupied.
- $Z$: The sensor observation set; one observation is denoted by $Z_s$, and the number of observation is denoted by $S$; each variable $Z_s$ is 4-dimensional.
- $M$: The “matching” variable; it specifies which observation of the sensor is currently used to estimate the state of a cell.

The decomposition of the joint distribution of these variables can be expressed as:
\begin{equation}
P(C^k E^k_C Z M) = P(C^k)P(E^k_C | C^k)P(M) \times \prod_{s=1}^{S} P(Z_s | C^k E^k_C M)
\end{equation}

Parametric forms can be assigned to each term of the joint probability decomposition.

- \(P(C^k)\) represents the information on the cell itself. As we always know the cell for which we are currently estimating the state, this distribution may be left unspecified.
- \(P(E^k_C | C)\) represents the a priori information on the occupancy of the cell. The prior distribution may be obtained from the estimation of the previous time step.
- \(P(M)\) is chosen uniformly. It specifies which observation of the sensor is used to estimate the state of a cell.
- The shape of \(P(Z_s | C^k E^k_C M)\) depends on the value of the matching variable.
  - If \(M \neq s\), the observation is not taken from the cell \(C\). Consequently, we cannot say anything about this observation. \(P(Z_s | C^k E^k_C M)\) is defined by a uniform distribution.
  - If \(M = s\), the form of \(P(Z_s | C^k E^k_C M)\) is given by the sensor model. Its goal is to model the sensor response knowing the cell state. Details on this model can be found in Elfes [1989].

It is now possible to ask the Bayesian question corresponding to the searched solution. Because the problem to solve consists of finding a good estimate of the cell occupancy, the question can be stated as the probability distribution on the state of cell occupancy, conditioned on the observations and the cell itself:

\begin{equation}
P(E^k_C | Z C^k)
\end{equation}

The result of the inference can be written as:

\begin{equation}
P(E^k_C | Z C^k) \propto \sum_{M=1}^{S} \left( \prod_{s=1}^{S} P(Z_s | E^k_C C^k M) \right).
\end{equation}

During inference, the sum on these variables allows every sensor observation to be taken into account during the update of the state of a cell. It should be noted that the estimation step is performed without any explicit association between cells and observations; this problematic operation is replaced by the integration on all the possible values of \(M\).

Figure 2 shows the estimation step expressed as a Bayesian program.

**Prediction in the 4D BOF**

The goal of this processing step is to estimate an a priori model of the occupancy probability at time \(k\) of a cell using the latest estimation of the occupancy grid, i.e. the estimation at time \(k-1\).
The relevant variable specifications are the same as those of the estimation stage except for the variable \( U \), which represents the “control” input of the CyCab at time \( k - 1 \). For example, it could be a measurement of its instantaneous velocity at time \( k - 1 \).

The decomposition of the joint distribution can therefore be expressed as follows.

\[
P(C_k E_{C_k}^{k-1} E_{C_k}^{k-1} U^{k-1})
\]

\[
= P(C_{k-1}) \times P(U^{k-1}) \times P(E_{C_k}^{k-1} | C^{k-1})
\times P(C_k | C^{k-1} U^{k-1}) \times P(E_{C_k}^{k} | E_{C_k}^{k-1} C^{k-1} C^{k})
\]

The parametric forms for each of the decomposition terms are as follows.

- \(P(C_{k-1})\) and \(P(U^{k-1})\) are chosen as uniform distributions.
- \(P(E_{C_k}^{k-1} | C^{k-1})\) is given by the result of the estimation step at time \( k - 1 \).
- \(P(C_k | C^{k-1} U^{k-1})\) is given by the dynamic model. It represents the probability that an object has moved from cell \( C^{k-1} \) to cell \( C_k \). This movement is because of the object itself and the robot’s movement between
times $k - 1$ and $k$. To define this model, we suppose a constant velocity model subject to zero-mean Gaussian errors for the moving objects.

- $P(E_C^k | E_C^{k-1}C^{k-1}C^k)$ represents the probability that an existing object at time $k - 1$ (i.e. $[E_C^{k-1} = 1]$) still exists at time $k$ (i.e. $[E_C^k = 1]$). As we consider that objects cannot disappear, Dirac functions are chosen for these distributions.

The problem to be solved is to find an estimate of the occupancy probability for each cell of the grid. This problem can be solved by posing the following question.

$$P(E_C^k | C^k U^{k-1})$$

(5)

This question (eq. 5) can be expressed as follows.

$$P(E_C^k | C^k U^{k-1}) \propto \sum_{c_{k-1}^{c_{k-1}}} P(C^k | C^{k-1}U^{k-1}) P(E_C^{k-1} | C^{k-1}).$$

(6)

Unfortunately, for most cases, this expression cannot be expressed analytically, and so it cannot be computed in real time. This is why an approximate solution of the integral term must be computed. Our approach to this computation assumes that only a few points are required to approximate the integral. Thus, for each cell of the grid at time $k - 1$, we can compute the probability distribution $P(C^k | C^{k-1}U^{k-1})$; then a cell $C^k$ is drawn according to this probability distribution; finally, cell $C^{k-1}$ is used to update the predicted state of cell $C^k$. It should be noted that the complexity of this algorithm increases linearly with the number of cells in our grid and ensures that the most informative points are used to compute the sum appearing in (6).

The prediction step can hence be expressed as the Bayesian program in Fig. 3.

An illustration of the BOF can be found in Fig. 4. The figures represent a dynamic scene containing two moving obstacles along with the results from the prediction and estimation stages. It also demonstrates the robustness of the BOF in occlusion.

The first row describes the experimental conditions: the sensor (a Sick laser rangefinder) is immobile, and it observes two objects $O1$ and $O2$ moving in opposite directions. In the situation depicted by Fig. 4-c1, $O1$ is temporarily hidden by $O2$ (and thus $O1$ is not detected by the sensor).

The second and the third rows show the results of the prediction step and the estimation step respectively. Only the cells of the grid corresponding to a relative speed equal to $(\dot{x} = 0.0m/s, \dot{y} = 1.0m/s)$, which is close to the speed of $O1$, are shown. The occupancy probabilities of the related cells are represented by the grey levels.

In this example, an area of “high occupancy probability”, which corresponds to the moving objects, is well characterized in Figs. 4-a2 and 4-a3. One can also notice that the areas hidden by the moving objects have occupancy probability values equal to 0.5. Similar results can be found from Figs.
An alternative formulation presents the BOF as 2D grids instead of the previous formulation of 4D grids. This model of the dynamic grid is different from the approach adopted in the original BOF formulation by Coué et al. [2006]. Their grid model is in 4-dimensional space whereas the 2D BOF [Tay et al., 2007] models the grid in 2-dimensional space. A subtle difference is that the 4D BOF allows the representation of overlapping objects but the 2D BOF does not. A more obvious difference is the ability to infer velocity distributions from the approach adopted in the original BOF formulation by Coué et al.

2.2 The 2D Bayesian occupancy filter

An alternative formulation presents the BOF as 2D grids instead of the previous formulation of 4D grids. This model of the dynamic grid is different from the approach adopted in the original BOF formulation by Coué et al. [2006]. Their grid model is in 4-dimensional space whereas the 2D BOF [Tay et al., 2007] models the grid in 2-dimensional space. A subtle difference is that the 4D BOF allows the representation of overlapping objects but the 2D BOF does not. A more obvious difference is the ability to infer velocity distributions...
Fig. 4. A short sequence of a dynamic scene. The first row describes the situation: a moving object is temporarily hidden by a second object. The second row shows the predicted occupancy grids, and the third row shows the result of the estimation step. The grids show $P(E_C^k = 1 | x \ y \ [\dot{x} = 0.0 \ [\dot{y} = 1.0])$

in the 2D BOF model, which is absent in the 4D BOF model as it requires the specification of the dynamics of the cells.

The 2D BOF can also be expressed as a Bayesian program. In the spirit of Bayesian programming, we start by defining the relevant variables.

- $C$ is an index that identifies each 2D cell of the grid.
- $A$ is an index that identifies each possible antecedent of the cell $c$ over all the cells in the 2D grid.
- $Z_t \in \mathcal{Z}$ where $Z_t$ is the random variable of the sensor measurement relative to the cell $c$.
- $V \in \mathcal{V} = \{v_1, \ldots, v_n\}$ where $V$ is the random variable of the velocities for the cell $c$ and its possible values are discretized into $n$ cases.
$O, O^{-1} \in O \equiv \{\text{occ}, \text{emp}\}$ where $O$ represents the random variable of the state of $c$ being either “occupied” or “empty”. $O^{-1}$ represents the random variable of the state of an antecedent cell of $c$ through the possible motion through $c$. For a given velocity $v_k = (v_x, v_y)$ and a given time step $\delta t$, it is possible to define an antecedent for $c = (x, y)$ as $c^{-k} = (x - v_x \delta t, y - v_y \delta t)$.

The following expression gives the decomposition of the joint distribution of the relevant variables according to Bayes’ rule and dependency assumptions.

$$P(C, A, Z, O, O^{-1}, V) = P(A)P(V|A)P(C|V, A)P(O^{-1}|A)P(O|O^{-1})P(Z|O, V, C)$$ (7)

The parametric form and semantics of each component of the joint decomposition are as follows.

- $P(A)$ is the distribution over all the possible antecedents of the cell $c$. It is chosen to be uniform because the cell is considered reachable from all the antecedents with equal probability.
- $P(V|A)$ is the distribution over all the possible velocities of a certain antecedent of the cell $c$; its parametric form is a histogram.
- $P(C|V, A)$ is a distribution that explains whether $c$ is reachable from $[A = a]$ with the velocity $[V = v]$. In discrete spaces, this distribution is a Dirac with value equal to one if and only if $c_x = a_x + v_x \delta t$ and $c_y = a_y + v_y \delta t$, which follows a dynamic model of constant velocity.
- $P(O^{-1}|A)$ is the conditional distribution over the occupancy of the antecedents. It gives the probability of the possible previous step of the current cell.
- $P(O|O^{-1})$ is the conditional distribution over the occupancy of the current cell, which depends on the occupancy state of the previous cell. It is defined as a transition matrix: $T = \begin{bmatrix} 1 - \epsilon & \epsilon \\ \epsilon & 1 - \epsilon \end{bmatrix}$, which allows the system to use the null acceleration hypothesis as an approximation; in this matrix, $\epsilon$ is a parameter representing the probability that the object in $c$ does not follow the null acceleration model.
- $P(Z|O, V, C)$ is the conditional distribution over the sensor measurement values. It depends of the state of the cell, the velocity of the cell and obviously the position of the cell.

In the 2D BOF, the Bayesian question will be the probability distribution on the occupation and velocity for each cell of the grid.

$$P(O \mid Z \ C)$$

$$P(V \mid Z \ C)$$

The 2D BOF can be formulated as the Bayesian program in Figure 5.
The aim of filtering in the BOF grid is to estimate the occupancy and grid velocity distributions for each cell of the grid, $P(O, V | Z, C)$.

Figure 6 shows how Bayesian filtering is performed in the 2D BOF grids. The two stages of prediction and estimation are performed for each iteration. In the context of the BOF, prediction propagates cell occupation probabilities for each velocity and cell in the BOF grid ($P(O, V | Z, C)$). During estimation, $P(O, V | C)$ is updated by taking into account its observation $P(Z | O, V, C)$ to obtain its final Bayesian filter estimation $P(O, V | Z, C)$. The result from the Bayesian filter estimation is then used for prediction in the next iteration.

From Fig. 6, the difference between the 2D BOF and the 4D BOF is clearly illustrated. First, the 2D BOF defines the velocity of the cell occupation as a variable in the Bayesian program. The velocity is not expressed as a variable in the Bayesian program in the 4D BOF, but it is rather defined as a prior dynamic model to be given $P(C^k | C^{k-1}, U^{k-1})$ (Fig. 3). The 2D BOF is thus
capable of performing inference on both the occupation and the velocity of the cell’s occupation. Second, the 2D BOF inherently expresses the constraint of a single occupation and velocity for each cellular decomposition of the 2D Cartesian space. However, in the 4D BOF, there is the possibility of expressing occupation of a cell in 2D Cartesian space with different velocities. The reduction in complexity from four dimensions to two reduces the computational complexity.

When implementing the 2D BOF, the set of possible velocities is discretized. One way of implementing the computation of the probability distribution is in the form of histograms. The following equations are based on the discrete case. Therefore, the global filtering equation can be obtained by:

\[
P(V, O|Z, C) = \frac{\sum_{A, O^{-1}} P(C, A, Z, O, O^{-1}, V)}{\sum_{A, O^{-1}, V} P(C, A, Z, O, O^{-1}, V)},
\]

which can be equivalently represented as:

\[
P(V, O, Z, C) = P(Z|O, V, C) \left[ \sum_{A, O^{-1}} P(A)P(V|A)P(C|V, A)P(O^{-1}|A)P(O|O^{-1}) \right].
\]

The summation in the above expression represents the prediction; its multiplication with the first term, \(P(Z|O, V, C)\), gives the Bayesian filter estimation.

The global filtering equation (eqn. 8) can actually be separated into three stages. The first stage computes the prediction of the probability measure for each occupancy and velocity:
The Bayesian occupation filter \[\alpha(\text{occ}, v_k) = \sum_{A,O^{-1}} P(A)P(v_k|A)P(C|V,A)P(O^{-1}|A)P(\text{occ}|O^{-1}),\]

\[\alpha(\text{emp}, v_k) = \sum_{A,O^{-1}} P(A)P(v_k|A)P(C|V,A)P(O^{-1}|A)P(\text{emp}|O^{-1}).\]

Equation 9 is performed for each cell in the grid and for each velocity. Prediction for each cell is calculated by taking into account the velocity probability and occupation probability of the set of antecedent cells, which are the cells with a velocity that will propagate itself in a certain time step to the current cell.

With the prediction of the grid occupancy and its velocities, the second stage consists of multiplying by the observation sensor model, which gives the unnormalized Bayesian filter estimation on occupation and velocity distribution:

\[\beta(\text{occ}, v_k) = P(Z|\text{occ}, v_k)\alpha(\text{occ}, v_k),\]

\[\beta(\text{emp}, v_k) = P(Z|\text{emp}, v_k)\alpha(\text{emp}, v_k).\]

Similarly to the prediction stage, these equations are performed for each cell occupancy and each velocity. The marginalization over the occupancy values gives the likelihood of a certain velocity:

\[l(v_k) = \beta(\text{occ}, v_k) + \beta(\text{emp}, v_k).\]

Finally, the normalized Bayesian filter estimation on the probability of occupancy for a cell \(C\) with a velocity \(v_k\) is obtained by:

\[P(\text{occ}, v_k|Z,C) = \frac{\beta(\text{occ}, v_k)}{\sum_{v_k} l(v_k)}.\]

The occupancy distribution in a cell can be obtained by the marginalization over the velocities and the velocity distribution by the marginalization over the occupancy values:

\[P(O|Z,C) = \sum_{V} P(V,O|Z,C),\]

\[P(V|Z,C) = \sum_{O} P(V,O|Z,C).\]

3 Applications

The goal of this section is to show some examples of applications using BOF\(^5\). Two different experiments are shown. The first is on estimating collision dan-

\(^5\) Different videos of these applications may be found at the following URLs: http://www.bayesian-programming.org/videoB1Ch4-
ger, which is in turn used for collision avoidance. The 4D BOF was used for the first experiment. The second is on object-level human tracking using a camera and was based on the 2D BOF.

3.1 Estimating danger

The experiments on danger estimation and collision avoidance were conducted using the robotic platform CyCab. CyCab is an autonomous robot fashioned from a golf cab. The aim of this experiment is to calculate the danger of collision with dynamic objects estimated by the BOF, followed by a collision avoidance manoeuvre.

The cell state can be used to encode some relevant properties of the robot environment (e.g. occupancy, observability and reachability). In the previous sections, only the occupancy characteristic was stored; in this application, the danger property is encoded as well. This will lead to vehicle control by taking occupancy and danger into account.

![Cells with high danger probabilities. For each position, arrows model the speed.](image)

**Fig. 7.** Cells with high danger probabilities. For each position, arrows model the speed.

For each cell of the grid, the probability that this cell is hazardous is estimated; this estimation is done independently of the occupancy probability. Let $P(D^k | C^k)$ be the probability distribution associated with the cell $C^k$.

Fig. 8. Scenario description: the pedestrian is temporarily hidden by a parked car.

of the vehicle environment, where $D_k^X$ is a boolean variable that indicates whether this cell is hazardous or not.

Fig. 9. Snapshots of the experimental pedestrian avoidance scenario (see Extension 1 for the video).

Basically, both “time to collision” and “safe travelling distance” may be seen as two complementary relevant criteria to be used for estimating the danger to associate with a given cell. In our current implementation, we are using the following related criteria, which can easily be computed: (1) the closest point of approach (CPA), which defines the relative positions of the
pair (vehicle, obstacle) corresponding to the “closest admissible distance” (i.e. safe distance); (2) the time to the closest point of approach (TCPA), which is the time required to reach the CPA; and (3) the distance at the closest point of approach (DCPA), which is the distance separating the vehicle and the obstacle when the CPA has been reached. In some sense, these criteria give an assessment of the future relative trajectories of any pair of environment components of the types (vehicle, potential obstacle).

These criteria are evaluated for each cell at each time step \( k \), by taking into account the dynamic characteristics of both the vehicle and the potential obstacles. In practice, both TCPA and DCPA are estimated under the hypothesis that the related velocities at time \( k \) remain constant; this computation can easily be done using some classical geometrical algorithms (see for instance: http://softsurfer.com/algorithms.htm).

The goal is to estimate the “danger probability” associated with each cell of the grid (or in other terms, the probability for each cell \( C^k \) that a collision will occur in the near future between the CyCab and a potential obstacle in \( C^k \)). Because each cell \( C^k \) represents a pair (position, velocity) defined relative to the CyCab, it is easy to compute the TCPA and DCPA factors, and in a second step to estimate the associated danger probability using given intuitive user knowledge. In the current implementation, this knowledge roughly states that when the DCPA and the TCPA decrease, the related probability of collision increases. In future versions of the system, such knowledge can be acquired with a learning phase.

Figure 7 shows the cells for which the danger probability is greater than 0.7 in our CyCab application; in the figure, each cell is represented by an arrow: its tail indicates the position, and its length and direction indicate the associated relative speed. This figure exhibits quite reasonable data: cells located near the front of the CyCab are considered as having a high danger probability for any relative velocity (the arrows are pointing in all directions); the other cells having a high “oriented” danger probability are those having a relative speed vector oriented towards the CyCab. Because we only consider relative speeds when constructing the danger grid, the content of this grid does not depend on the actual CyCab velocity.

3.2 Collision avoidance behaviours

This section describes the control of the longitudinal speed of the autonomous vehicle (the CyCab), for avoiding partially observed moving obstacles having a high probability of collision with the vehicle. The implemented behaviour consists of braking or accelerating to adapt the velocity of the vehicle to the level of risk estimated by the system.

As mentioned earlier, this behaviour derives from the combination of two criteria defined on the grid: the danger probability associated with each cell \( C^k \) of the grid (characterized by the distribution \( P(D^k_C | C^k) \)), and the occupancy probability of this cell (characterized by the posterior distribution
\[ P(E_C^k | Z^k C^k) \]. In practice, the most hazardous cell that is considered as probably occupied is searched for; this can be done using the following equation:

\[
\max_{C^k} \{ P(D_C^k | C^k), \text{ with } P(E_C^k | C^k) > 0.5 \}.
\]

Then the longitudinal acceleration/deceleration to apply to the CyCab controller can be decided according to the estimated level of danger and to the actual velocity of the CyCab.

Figure 8 depicts the scenario used for experimentally validating the previous collision avoidance behaviour on the CyCab. In this scenario, the CyCab is moving forward, the pedestrian is moving from right to left, and for a small period of time, the pedestrian is temporarily hidden by a parked car.

Figure 9 shows some snapshots of the experiment (see also Extension 1, which shows the entire video): the CyCab brakes to avoid the pedestrian, then it accelerates as soon as the pedestrian has crossed the road.

**Fig. 10.** Velocity of the CyCab during the experiment involving a pedestrian occlusion.

Figure 10 shows the velocity of the CyCab during this experiment. From \( t = 0 \) s to \( t = 7 \) s, the CyCab accelerates, up to \( 2 \) m/s. At \( t = 7 \) s, the pedestrian is detected; as a collision could possibly occur, the CyCab decelerates. From \( t = 8.2 \) s to \( t = 9.4 \) s, the pedestrian is hidden by the parked car; thanks to the BOF results, the hazardous cells of the grid are still considered as probably occupied; in consequence the CyCab still brakes. When the pedestrian reappears at \( t = 9.4 \) s, there is no longer a risk of collision, and the CyCab can accelerate.

### 3.3 Object-level tracking

Experiments were conducted based on video sequence data from the European project CAVIAR. The selected video sequence presented in this paper is taken from the interior of a shopping centre in Portugal. An example is shown in the first column of Fig. 11. The data sequence from CAVIAR, which is freely available from the Web\(^6\), gives annotated ground truths for the detection of

\(^6\) http://groups.inf.ed.ac.uk/vision/CAVIAR/CAVIARDATA1/
the pedestrians. Another data set is also available, taken from the entry hall of INRIA Rhône Alpes.

Based on the given data, the uncertainties, false positives and occlusions have been simulated. The simulated data are then used as observations for the BOF. The BOF is a representation of the planar ground of the shopping centre within the field of view of the camera. With the noise and occlusion by simulated bounding boxes that represent human detections, a Gaussian sensor model is used, which gives a Gaussian occupation uncertainty (in the BOF grids) of the lower edge of the image bounding box after being projected onto the ground plane.

Recalling that there is no notion of objects in the BOF, object hypotheses are obtained from clustering, and these object hypotheses are used as observations on a standard tracking module based on the joint probabilistic data association (JPDA).

Previous experiments based on the 4D BOF technique (Section 3.1) relied on the assumption of a given constant velocity, as the problem of velocity estimation in this context has not been addressed. In particular, the assumption that there could only be one object with one velocity in each cell was not part of the previous model. In this current experiment, experiments were conducted based on the 2D BOF model, which gives both the probability distribution on the occupation and the probability distribution on the velocity.

The tracker is implemented in the C++ programming language without optimizations. Experiments were performed on a laptop computer with an Intel Centrino processor with a clock speed of 1.6 GHz. It currently tracks with an average frame rate of 9.27 frames/s. The computation time required for the BOF, with a grid resolution of 80 cells by 80 cells, takes an average of 0.05 s. The BOF represents the ground plane of the image sequence taken from a stationary camera and represents a dimension of 30 m by 20 m.

The results in Fig. 11 are shown in time sequence. The first column of the figures shows the input image with the bounding boxes, each indicating the detection of a human after the simulation of uncertainties and occlusions. The second column shows the corresponding visualization of the Bayesian occupancy filter. The colour intensity of the cells represents the occupation probability of the cell. The little arrows in each cell give the average velocity calculated from the velocity distribution of the cell. The third column gives the tracker output given by a JPDA tracker. The numbers in the diagrams indicate the track numbers. The sensor model used is a 2D planar Gaussian model projected onto the ground. The mean is given by the centre of the lower edge of the bounding box.

The characteristics of the BOF can be seen from Fig. 11. The diminished occupancy of a person further away from the camera is seen from the data in Figs. 11(b) and 11(e). This is caused by the occasional instability in human detection. The occupancy in the BOF grids for the missed detection diminishes gradually over time rather than disappearing immediately as it does...
Fig. 11. Data sequence from project CAVIAR with simulated inputs. The first column displays camera image input with human detection, the second column displays the BOF grid output, and the third column displays tracking output. Numbers indicate track numbers.
with classical occupation grids. This mechanism provides a form of temporal smoothing to handle unstable detection.

A more challenging occlusion sequence is shown in the last three rows of Fig. 11. Because of a relatively longer period of occlusion, the occupancy probability of the occluded person becomes weak. However, with an appropriately designed tracker, such problems can be handled at the object tracker level. The tracker manages to track the occlusion at the object tracker level as shown in Fig. 11(i)(l)(o).

4 Conclusion

In this chapter, we introduced the Bayesian occupation filter, its different formulations and several applications.

- The BOF is based on a gridded decomposition of the environment. Two variants were described, a 4D BOF in which each grid cell represents the occupation probability distribution at a certain position with a certain velocity, and a 2D BOF in which the grid represents the occupation probability distribution and each grid is associated with a velocity probability distribution of the cell occupancy.
- The estimation of cell occupancy and velocity values is based on the Bayesian filtering framework. Bayesian filtering consists of two main steps, the prediction step and the estimation step.
- The 4D BOF allows representation of several “objects”, each with a distinct velocity. There is also no inference on the velocity for the 4D BOF. In contrast, the 2D BOF implicitly imposes constraints in having only a single “object” occupying a cell, and there is inference on velocities for the 2D BOF framework. Another advantage of the 2D BOF framework over the 4D BOF is the reduction in computational complexity as a consequence of the reduction in dimension.
- There is no concept of objects in the BOF. A key advantage of this is “avoiding” the data association problem by resolving it as late as possible in the pipeline. Furthermore, the concept of objects is not obligatory in all applications.
- However, in applications that require object-based representation, object hypotheses can be extracted from the BOF grids using methods such as clustering.
- A grid-based representation of the environment imposes no model on the objects found in the environment, and sensor fusion in the grid framework can be conveniently and easily performed.

We would like to acknowledge the European project carsense: IST-1999-12224 “Sensing of Car Environment at Low Speed Driving”, Carsense [January 2000–December 2002] for the work on the 4D BOF [Coué et al., 2006].
References


Topological SLAM

Adriana Tapus\textsuperscript{1} and Roland Siegwart\textsuperscript{2}

\textsuperscript{1} Interaction Lab, University of Southern California (USC)
\textsuperscript{2} Autonomous Systems Lab, Swiss Federal Institute of Technology Zürich (ETHZ)

1 Introduction

In all our daily activities, the natural surroundings that we inhabit play a crucial role. Many neurophysiologists have dedicated their efforts towards understanding how our brains can create internal representations of physical space. Both neurobiologists and roboticists are interested in understanding the behaviour of intelligent beings like us and their capacity to learn and use their knowledge of the spatial representation to navigate. The ability of intelligent beings to localize themselves and to find their way back home is linked to their internal “mapping system”. Most navigation approaches require learning and consequently entail memorizing information. Stored information can be organized into cognitive maps - a term introduced for the first time in [Tolman, 1948]. Tolman advocates that the animals (rats) do not learn space as a sequence of movements; instead, the animal’s spatial capabilities rest on the construction of maps, which represent the spatial relationships between features in the environment.

Various methods have been proposed for representing environments in the framework of autonomous navigation, from precise geometric maps based on raw data or lines to purely topological maps using symbolic descriptions. Each of these methods is optimal with respect to some characteristics but can be very disappointing with respect to others. Most current approaches make a trade-off between precision and global distinctiveness. Precision and distinctiveness have strong links with the level of abstraction of the features used for navigation.

These levels of abstraction are represented in a pyramidal form, as depicted in Figure 1. It can be noticed that at higher levels in the hierarchy, the geometric information is reduced and the distinctiveness increases. For global localization and mapping, high distinctiveness is important, whereas for local action, precise geometric relations with the environment are more critical.

Our method uses fingerprints of places to create a qualitative model of the environment (i.e. partially geometric feature representation – the third level of
the hierarchy). The fingerprint approach, by combining the information from all sensors available to the robot, reduces perceptual aliasing and improves the distinctiveness of places. Here, we approach the Simultaneous Localization and Mapping (SLAM) problem, which is of a “chicken and egg” nature – to localize the robot, a map is necessary, and to update the map, the position of the mobile robot is needed. The objective of the work presented in this chapter is to enable autonomous navigation without relying on maps learned a priori, without using artificial landmarks and by employing a semantic spatial representation that allows a more natural interface between humans and robots. Therefore, this paper describes a new method for incremental and automatic topological mapping and global localization with Partially Observable Markov Decision Processes (POMDP), using fingerprints of places.

The rest of this chapter is structured as follows. Section 2 discusses some relevant related work. Section 3 presents a short review of the fingerprint concept and how it is encoded, generated and combined with the uncertainty of features. Sections 4 and 5 are dedicated to the new cognitive navigation approach with fingerprints of places. Experimental results are presented in Section 6. The indoor and outdoor systems use two 180° laser range finders and an omnidirectional camera for feature extraction. In Section 7, we discuss the similarities between the fingerprint approach and the hippocampus. Finally, Section 8 draws conclusions and discusses further work.

2 Related work

A robust navigation system requires a spatial model of the physical environment such as a metric or topological map. Approaches using metric maps are appropriate when the robot must know its location accurately in terms of its
coordinates. However, the state of the robot can also be represented in a more qualitative manner, similar to the way humans do it.

Although the literature related to SLAM is vast, we concentrate here on the papers that we consider most important and that have directly influenced our thinking and research work.

The SLAM problem, in the form of the construction of maps while the robot moves through the environment and localization with respect to the partially built maps, was introduced in robotics in a seminal paper by Smith and Cheeseman [Smith and Cheeseman, 1986] in 1986. One of the first systems implemented was developed by Moutarlier and Chatila [Moutarlier and Chatila, 1989]. This approach used Extended Kalman Filters (EKF) to estimate the posterior over the robot pose and maps. Leonard and Durrant-Whyte in [Leonard and Durrant-Whyte, 1992] proposed a stochastic method similar to SLAM. Metric maps are spatial representations that have been extensively studied in the robotics community. The stochastic map technique to perform SLAM [Castellanos and Tardos, 1999], [Dissanayake et al., 2001], [Leonard and Durrant-Whyte, 1992], and the occupancy grid approaches [Thrun, 1998] are typical examples of this kind of space representation. More recent vision-based metric approaches have used Scale Invariant Feature Transform (SIFT) features [Se et al., 2002]. The SIFT approach detects and extracts local feature descriptors that are invariant to illumination changes, image noise and rotation and scaling. All these metric methods are used with high-precision sensors. Thus, mapping yields a precise representation of the environment, and consequently localization is accurate. However, metric SLAM can become computationally very expensive for large environments. Thrun, in [Thrun, 2000], proposed probabilistic methods that make the metric mapping process faster and more robust. However, metric approaches also suffer from other shortcomings. One negative aspect of metric maps is that they are not easily extendable to be usable for higher-level, symbolic reasoning.

Topological approaches to SLAM attempt to overcome the drawbacks of geometric methods by modelling space using graphs. Significant progress has been made since the seminal paper by Kuipers [Kuipers, 1978], in which an approach based on concepts derived from a theory of human cognitive mapping was described as the body of knowledge representing large-scale space. Kortenkamp and Weymouth in [Kortenkamp and Weymouth, 1994] also used cognitive maps for topological navigation. They defined the concept of gateways, which have been used to mark the transition between two adjacent places in the environment. They used data from sonars combined with vision information to achieve a rich sensory place characterization. In [Hafner, 2000] and [Owen and Nehmzow, 1998], the authors used a model based on a self-organizing map that creates a topological representation of the environment while the robot explores it. Another topological model was described in [Choset and Nagatani, 2001]. The environment was represented with the help of a Generalized Voronoi Graph (G VG), and the robot was localized via a graph-matching process. Most recently, Beeson et al. used Extended Voronoi
Graphs (EVGs) to demonstrate place detection in the context of topological maps [Beeson et al., 2005]. In general, topological maps are less complex and permit more efficient planning than metric maps. Moreover, they are easier to generate. Maintaining global consistency is also easier with topological maps than with metric maps. However, the main problems with using topological maps are the perceptual aliasing (i.e. observations at multiple locations are similar) and the automatic establishment of a minimal topology (nodes).

Researchers have also integrated both the metric and topological paradigms, thereby obtaining a hybrid system. Thrun, in [Thrun, 2000], used occupancy grid-based maps to build a metric map. The topological map was extracted from the grid-based map. Learning a topological representation depends on learning a geometric map, which relies on the odometric capability of the robot. However, in large environments, it is difficult to maintain the consistency of the metric map, because of the drift in the odometry. In Tomatis et al. [2003] conceived a hybrid representation, similar to work mentioned above, consisting of a global topological map with local metric maps associated with each node for precise navigation. Another hierarchical multi-resolution approach allowing for high precision for metric mapping using a relative map filter and distinctiveness for topological mapping with fingerprints of places was presented in [Martinelli et al., 2003]. The authors of [Lisien et al., 2003] extended the model described in [Choset and Nagatani, 2001] to H-SLAM (i.e. Hierarchical SLAM) by combining the topological and feature-based mapping techniques. Another hybrid approach was described in [Dufourd et al., 2004]. Their model combined different representations (i.e. frontier-based, space-based, grid-based and topological), improving SLAM robustness and creating a more complex and useful spatial representation for reasoning and path planning.

3 Environmental modelling with fingerprints of places

Representing and interpreting a scene from the environment is a hard task. Humans use various sensory cues to extract crucial information from the environment. This is processed in the cortex of the brain to obtain a high-level representation of what has been perceived. Intuitively, it appears that humans represent knowledge in a hierarchical fashion.

With a view to having robots as companions of humans, we are motivated towards developing a knowledge representation system along the lines of what we know about ourselves. However, while recent research has shown interesting results, we are still far from having concepts and algorithms that can represent and interpret space and cope with the complexity of the environment.

Most of the time, with the exception of reactive-behaviour-based navigation, a space representation of the environment is needed to localize the robot. The notion of the fingerprint of a place is introduced here. The concept of fingerprints of places combines the information from all sensors available to the
robot and thereby enables a reduction in perceptual aliasing and improves the distinctiveness of places. This qualitative approach to representing the environment is defined and described in the followings subsections.

3.1 Fingerprint of a place: Definition

Just as each person has a unique fingerprint, each location in the environment has a unique set of characteristics associated with it. Of course, when relying on the limited perceptual capabilities of a machine, it is difficult to guarantee the unique distinction between two similar places. Our system assumes that a fingerprint of the current location can be created and that the sequence generation methods can be made insensitive to small changes in robot position. However, this characterization of the environment is especially interesting when used within a topological framework. In this case, the distinctiveness of the observed location plays an important role in reliable localization and consistent mapping. A fingerprint of a place is a circular list of features, where the ordering of the set matches the relative ordering of the features around the robot. We denote the fingerprint sequence using a list of characters, where each character represents an instance of a specific feature type (see Figure 2).

![Fig. 2. Fingerprint concept overview (where R – red colour patch, c – corner, G – green colour patch, d – door, Y – yellow colour patch, B – blue colour patch).]

3.2 Fingerprint encoding

As previously mentioned, a fingerprint of a place is a circular list of features that the robot can perceive around itself. In this work, a fingerprint is created by assuming that a set of feature extractors can identify significant features in the environment around the robot. Omnidirectional sensors are preferred because the orientation as well as the position of the robot may not be known a priori. In this work, we choose to extract colour patches and vertical edges from visual information, and corners from a laser scanner. The letter v is used
to characterize an edge, the letters $A, B, C, ..., P$ to represent hue bins and the letter $c$ to characterize a corner feature. A corner feature is defined as the extremity of a line-segment extracted with the Douglas–Peucker algorithm [Douglas and Peucker, 1973].

### 3.3 Fingerprint generation

Fingerprint generation is performed in three steps, as shown in Figure 3. The extraction of the different features (e.g. vertical edges, corners and colour patches) from the sensors is the first step of the fingerprint generation process. The extracted features are ordered in a sequence depending on their angular position ($0...360^\circ$). In the second step, a new type of feature, the virtual feature $f$ is introduced. This reflects the correspondence between a corner (detected with the laser scanner) and an edge (detected in the unwrapped omnidirectional image). To represent large angular distances between successive fingerprint elements ($> 20^\circ$ degrees, in our case), the notion of an empty space feature is added. This is denoted in the fingerprint sequence by the character $n$. This insertion is the last step of the fingerprint generation process.

### 3.4 Uncertainty modelling

The interaction between a mobile robot and its surroundings is performed by means of exteroceptive sensor data. Sensors are imperfect devices, and thus the measurements always contain errors. This can be modelled by associating uncertainty with the data. For this reason, probabilities are used to model the uncertainty of the geometric features extracted from the environment. We define the uncertainty $u_{\text{feature}}$ as the probability of a feature being present in the environment when the robot perceives it. In our fingerprint approach, this idea is incorporated by associating every observed feature (for each of the different types of features mentioned above) with an uncertainty measure. These uncertainty measures are modelled by experience for each type of feature presented in Figure 3: vertical edges, colours, corners (extremities of the segments), $f$ features and $n$ features. More details can be found in [Tapus et al., 2004].

In this way, the uncertainty of the features used in the fingerprints of places is calculated. The limitation of this method resides in the models, which are difficult to define, especially for our definition of uncertainty, which cannot be directly derived from the physical characteristics of the sensors.

### 4 Topological localization

Finding an efficient solution to the robot localization problem will have a tremendous impact on the manner in which robots are integrated into our
Fig. 3. Fingerprint generation. (a) Panoramic image with the vertical edges and colour patches detected, denoted by $v$ and $A...P$, respectively. (b) Laser scan with extracted corners $c$; (c) the first three graphs depict the position (0 to 360°) of the colours ($I$-light blue, $B$-orange and $E$-light green), vertical edges and corners, respectively. The fourth graph describes the correspondence between the vertical edge features and the corner features. By regrouping all these results together and adding the empty space features, the final fingerprint is: $cI f v n c v v n c v v B n c v v B n v B c c E$.

daily lives. Most tasks for which robots are well suited demand a high degree of robustness in their localizing capabilities before they are actually applied in real-life scenarios.

In this work, we elected to represent the environment in a topological fashion. The topological map can be viewed as a graph of places, in which for each node, information is stored concerning the visible landmarks and the way to reach other places connected to it. The fingerprint of a place can easily be used to represent places and therefore the nodes in the topological framework. This section presents a Bayesian localization algorithm for a topological (fingerprint-based) environment model. The approach also describes how multimodal perception increases the reliability of topological localization (using the Bayesian programming formalism) for mobile robots.

The Bayesian approach to localization with the fingerprints of places, presented here, is composed of two phases. In the first phase of supervised learning, the robot inspects several locations, denoted by $Loc$. From each location $loc \in Loc$, the robot extracts the fingerprint data, as explained in Section 3.3,
and stores them along with the name of the location in a database, denoted by the symbol $\delta$. In the second phase, application, the robot localizes itself in the environment. To answer the question “Where am I?”, the robot will extract the features comprising the fingerprint of its surroundings: the set of vertical edges $VE$, the set of colour patches $CP$, and the set of corners $C$, and will solve the question corresponding to probabilistic robot localization given as:

$$loc^* = \arg \max_{loc \in Loc} P(loc \mid C \land VE \land CP \land Fp \land \pi \land \delta). \quad (1)$$

This means that if fingerprints of places are associated with each location, then the actual location of the robot may be recovered by comparing the features composing the fingerprint of the current place with the database of known locations $Loc$. The location $loc^*$ that maximizes the probability measure is chosen. The preliminary knowledge is summarized by $\pi$. In the following, we show how it can be solved by the Bayesian programming technique.

Figure 4 illustrates the Bayesian program used for Bayesian fingerprint matching. As mentioned above, several features are used in the fingerprints of places: $VE$, $CP$, and $C$. The variable $Fp$ represents the fingerprint string constructed over all the features. Although this adds some redundancy to the system, it simultaneously introduces valuable information about the relative order of the features. These variables, $VE$, $CP$, $C$ and $Fp$, are independent
of one another, but they are location dependent and it is these dependencies that give rise to the Bayesian program formulation shown in Figure 4.

The decomposition (see Bayesian Program) described above involves three different kinds of probability distributions.

- Because no \textit{a priori} information about locations is available, it is considered that each location is equally probable, and consequently the probability of a location given all the prior knowledge is expressed as a uniform distribution.

- The probability of one feature \( f \), where \( f \in \{VE, CP, C\} \), given the location and all the \textit{a priori} knowledge, is given as the likelihood of the new feature data \( f \) with respect to the distribution of the same feature as that encountered at the given location during the learning phase. This distribution is calculated as a Mixture of Gaussians (MOG) in angle space, optimizing the mixture parameters by making use of the Expectation Maximization (EM) algorithm. More details about these two concepts are described in the following subsections.

- To calculate the probability of the fingerprint sequence \( Fp \) given the location and all prior knowledge \( P(Fp \mid \text{loc} \land \pi \land \delta) \), the global alignment algorithm is used.

The equations from the parametric forms will be used to solve the basic question described in the Bayesian program (see Figure 4).

### 4.1 Mixture of Gaussians

Mixture of Gaussians (MOG) is a widely used approach when estimating the distribution of data. An MOG in the parameters \( \theta \) is a probability density function, which results from combining \( k \) Gaussian probability density functions in a weighted sum:

\[
P_{\text{MOG}(\theta)}(x) = \sum_{i=1}^{k} w_i \ P(x \mid \theta_i),
\]

\[
\theta_i = \{w_i, \mu_i, \sigma_i\},
\]

where \( w_i \) is the weight, \( \mu_i \) is the mean and \( \sigma_i \) is the standard deviation of the \( i \)th mixture component, which itself is a Gaussian probability density function given by the formula:

\[
P(x \mid \theta_i) = N(x, \mu_i, \sigma_i) = \eta \ exp\left(-\frac{(\mu_i - x)^2}{2\sigma_i^2}\right).
\]

The normalization factor \( \mu \) turns the Gaussian function into a probability distribution function by guaranteeing that the integral over the function evaluates to one:
In angle space, $\mu$ is the inverse of the integral from $-\pi$ to $\pi$ over the unnormalized Gaussian function of:

$$\mu = \frac{1}{\text{erf}\left(\frac{\sqrt{2\pi}}{2\sigma_i}\right)}.$$  

where $\text{erf}(x)$ is the error function:

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt.$$  

Because $P_{\text{MOG}}$ is also a probability density function, the weights $w_i$ must sum to one, such that the integral over the distribution is one:

$$\sum w_i = 1,$$

$$\int P_{\text{MOG}}(x \mid \theta_i) = 1.$$  

The parameters of the complete MOG are then:

$$\theta_{\text{MOG}} = \{\theta_1...\theta_n\} = \{w_1...w_n, \mu_1...\mu_n, \sigma_1...\sigma_n\}.$$  

The MOG is a compromise between the efficient but parametric models on one side, and the flexible but expensive nonparametric methods like histograms or kernel methods on the other.

### 4.2 Expectation maximization

Finding the optimal parameters $\theta_{\text{MOG}}$ of a MOG over a set of data points $X$ is not trivial. A widely used approach to solving this problem is the Expectation Maximization (EM) algorithm [Bilmes, 1997].

This algorithm starts with an initial estimate of the parameters $\theta$ and improves upon them iteratively. The algorithm proceeds in two steps.

- **E-step:** Calculates the complete data likelihood given the known data $X$ and the current parameters $\theta$.
- **M-step:** Calculates the new parameters $\theta^{\text{new}}$ that maximize the joint probability $P(X \land Y \mid \theta)$, where $Y$ is the hidden data, which in our case are the knowledge about the probability that the $i$th data point $x_i$ was generated by the $k$th mixture component.
The “improvement” is defined in the sense that the log-likelihood of the data $X$ increases with respect to the new parameters $\theta^{\text{new}}$. For mixtures of Gaussians, it is possible to derive the new parameters $\theta^{\text{new}}$ analytically. The resulting formulas merge the E-step and M-step and are given by:

$$w_k^{\text{new}} = \frac{1}{N} \sum_{i=1}^{n} P(k \mid x_i \land \theta),$$  

$$\mu_k^{\text{new}} = \mu_k^{\text{old}} + \frac{\sum_{i=1}^{n} \text{dist}_{\text{AS}}(\mu_k^{\text{old}}, x_i) P(k \mid x_i \land \theta)}{\sum_{i=1}^{n} P(k \mid x_i \land \theta)},$$  

$$\sigma_k^{\text{new}} = \sigma_k^{\text{old}} + \frac{\sum_{i=1}^{n} P(k \mid x_i \land \theta) \text{dist}_{\text{AS}}(\mu_k^{\text{old}}, x_i)^2}{\sum_{i=1}^{n} P(k \mid x_i \land \theta)},$$

where $N$ is the number of data points, $N = |X|$, and $\text{dist}_{\text{AS}}(a, b)$ is the distance function in angle space. It takes two angles, $a$ and $b$, and returns the shortest way to go from $a$ to $b$. The sign of the distance is positive if going clockwise and negative if going counterclockwise. The iteration is typically terminated when the increase of the log-likelihood falls below some threshold value $\varepsilon$.

### 4.3 Global alignment with uncertainty

Matching the strings of two fingerprints of places is not an easy task. Usually strings do not match exactly because the robot may not be exactly located on a map point and/or some changes in the environment or perceptual errors may have occurred. Many string-matching algorithms can be found in the literature, but they generally require the strings to have the same length. Some of them allow a level of mismatch, such as the $k$-mismatch matching algorithms and string matching with $k$ differences [Aho, 1990], [Baeza-Yates and Navarro, 1999]. The first allows matches where up to $k$ characters in the pattern do not match the text, and the second requires that the pattern have an edit distance\(^3\) from the text of $k$ or fewer elements. One of the main problems of the above methods is that they do not consider the nature of the features and the specific nature of the mismatch that occurred.

The likelihood of specific types of mismatch errors must be taken into account. For instance, confusing a red patch with a blue patch is more serious than confusing a red patch with a yellow patch. The standard algorithms are quite sensitive to insertion and deletion errors that cause the string lengths to vary significantly. The method adopted previously in the fingerprint approach [Lamon et al., 2001] for sequence matching is the minimum energy algorithm usually used in stereovision [Kanade and Ohta, 1985].

\(^3\) The edit distance of two strings, $s_1$ and $s_2$, is defined as the minimum number of point mutations required to change $s_1$ into $s_2$, where a point mutation is one of: change a letter, insert a letter, and delete a letter.
The approach used in this work is an extension of the global alignment algorithm, typically used for DNA sequence matching [Needleman and Wunsch, 1970], that incorporates uncertainties into its formalism.

The global alignment algorithm finds an alignment between two strings such that the total cost, computed using a cost function for aligning two characters, is minimized. We can distinguish five elements that together constitute the global alignment algorithm (see Figure 5):

- The alphabet $A$, typically a set of letters, that is not empty.
- The two strings that are to be aligned: the first is composed of $m$, the second of $n$ letters of the alphabet.
- The occlusion symbol is used to represent a space inserted into the string.
- The cost function gives the cost for the match between two symbols of the alphabet, including the occlusion symbol $\varepsilon$.
- The cost matrix is used to keep the minimal cost of a match between the first $i$ letters of the first string and the first $j$ letters of the second string, keeping this value in element $(i, j)$ of matrix $V$.

The values of the cost function, $f_{cost}(a, b)$, are calculated in our experiments as a function of the similarity between characters $a$ and $b$. In other words, similar characters have lower penalties for mismatching. It only remains to calculate the values of the elements of the cost matrix, which is realized through dynamic programming. Initially the edges of the matrix (i.e. $V(0, j)$ and $V(i, 0)$) are initialized with the cumulative cost of occlusions. (see equations 14 and 15) This reflects the fact that it is not known a priori how many letters must be jumped in one or the other string to obtain the best solution.

The base conditions of the algorithm are:

$$V(0, j) = \sum_{1 \leq k \leq j} f_{cost}(\varepsilon, S_2(k)),$$

$$V(i, 0) = \sum_{1 \leq k \leq i} f_{cost}(S_1(k), \varepsilon).$$

For $i$ and $j$ both strictly positive, the recurrence relation is:
Topological SLAM

\[ V(i, j) = \min \begin{cases} 
V(i-1, j-1) + f_{cost}(S_1(i), S_2(j)) \\
V(i-1, j) + f_{cost}(S_1(i), \varepsilon) \\
V(i, j-1) + f_{cost}(\varepsilon, S_2(j)) 
\end{cases} \]  

(16)

The three cases that can be distinguished from the above relation are as follows.

- Aligning \( S_1(i) \) with \( S_2(j) \). The score in this case is the score of the operation \( f_{cost}(S_1(i), S_2(j)) \) plus the score of aligning \( i-1 \) elements of \( S_1 \) with \( j-1 \) elements of \( S_2 \), namely, \( V(i-1, j-1) + f_{cost}(S_1(i), S_2(j)) \).
- Aligning \( S_1(i) \) with an occlusion symbol in string \( S_2 \). The score in this case is the score of the operation \( f_{cost}(S_1(i), \varepsilon) \) plus the score of aligning the previous \( i-1 \) elements of \( S_1 \) with \( j \) elements of \( S_2 \) (because the occlusion is not an original character of \( S_2 \)), \( V(i-1, j) + f_{cost}(S_1(i), \varepsilon) \).
- Aligning \( S_2(j) \) with an occlusion symbol in string \( S_1 \). Similarly to the previous case, the score will be \( V(i, j-1) + f_{cost}(\varepsilon, S_2(j)) \).

If strings \( S_1 \) and \( S_2 \) are of length \( n \) and \( m \) respectively, then the cost of their optimal alignment with the global alignment is the value of the cell \((n, m)\).

The global alignment with uncertainty algorithm changes only the cost function \( f_{cost}(a, b) \) described earlier. The cost function is adapted to take into account the corresponding uncertainty of features, and it is expressed as \( f_{cost}(a, \text{uncert}_a, b, \text{uncert}_b) \). The goal of adding uncertainties to the string-matching algorithm is to improve the distinctiveness of places.

![Fig. 6. An example of the global alignment algorithm with uncertainty: F1 and F2 are the two fingerprints of places stored in the database of known locations. FObs is the observed fingerprint of place.](image)

The example depicted in Figure 6 shows the improvement obtained by the new fingerprint matching with uncertainty algorithm. Even if the two fingerprints of places from the map are similar (i.e. \( \text{string1} \) and \( \text{string2} \)), the
uncertainty of the features will determine the stored fingerprint of place that best matches the observed fingerprint of place (i.e. stringObs).

5 Topological SLAM

Navigation, described by Gallistel in [Gallistel, 1990] as the capacity to localize oneself with respect to a map, is an elementary task that an autonomous mobile robot must carry out. Therefore, map building is the task of creating representations of the environment that the robot is moving in. The maps thus built are used for localizing the mobile robot. In this section, the methods used to build topological maps and globally localize the robot automatically to perform SLAM are described.

5.1 Topological mapping

While navigating in the environment, the robot first creates and then updates the global topological map. One of the main issues in topological map building is detecting when a new node should be added to the map. Most of the existing approaches to topological mapping place nodes periodically in either space (displacement, $\Delta d$) or time ($\Delta t$) or alternatively attempt to detect important changes in environment structure. None of these methods results in an optimal topology.

In contrast, the approach presented in this work is based directly on the differences in the perceived fingerprints. A new node is introduced in the map whenever an important change in the perceived and extracted fingerprint occurs. This is possible using the fingerprints of places. A heuristic is applied to compare whether the fingerprint of a new location is similar to the last one that was mapped (see Figure 7).

The process of introducing a new node in the topological map is illustrated in Figure 8.

A new node is introduced into the topological map whenever the dissimilarity of the newly perceived fingerprint is larger than a threshold. Each node will therefore be composed of a set of similar fingerprints of places. To compact the current representation even more, a unique identifier named the mean fingerprint is generated for each node. This technique of clustering fingerprints of places into a single representation enables the construction of a very distinctive and compact representation of the environment. Thus, a new node contains all posterior knowledge about the environment acquired between the previous node and the present one [Tapus and Siegwart, 2005]. We defined a threshold $\theta$ as the maximum allowable dissimilarity (i.e. $1 - prob\_matching$) between the fingerprints. The value of $prob\_matching$ is calculated with the global alignment with uncertainty algorithm. The value of the threshold is determined experimentally. The incremental nature of the approach permits incremental additions to the map and yields the most up-to-date map at any time.
**Fig. 7.** Adding a new node automatically to the topological map while moving in an unexplored environment. The image is composed of seven measurement points (i.e. fingerprints of places) represented by the black points. The blue points depict the data given by the laser rangefinder, and they are used as references only. The mapping system includes all the fingerprints of places in a node until a significant change in the environment occurs and the dissimilarity between the fingerprints is greater than the threshold $\theta$.

**Fig. 8.** Flow-chart of the algorithm for detecting new topological nodes.
5.2 Global topological localization with POMDP

The localization method presented in Section 4 compares the observed features encoded in the fingerprints of places with the map fingerprints. Only the exteroceptive sensory information contained in fingerprints of places is used for matching, without taking into account the motion of the robot or previous estimations.

Hence, for topological navigation, a Partially Observable Markov Decision Process (POMDP) model [Cassandra et al., 1996] is used here. POMDP integrates both the robot’s motion and exteroceptive sensor report data to estimate the pose distribution. The probability of being in a place is calculated as a function of the last probability distribution and the current action and observation.

POMDP is defined as \( \langle S, A, T, O \rangle \), where:

- \( S \) is a finite set of environment states;
- \( A \) is a finite set of actions;
- \( T(s, a, s') \) is a transition function between environment states based on the performed action;
- \( O \) is a finite set of possible observations; and
- \( OS \) is an observation function.

With this information, the probability of being in a state \( s' \) (belief state of \( s' \)) after making observation \( o \), while performing action \( a \), is given by:

\[
SE_{S_{t+1}}^{s'} = \frac{OS(o, s') \sum_{s \in S} T(s, a, s') SE_{S_{t}}^{s}}{P(o \mid a \land SE_{t}^{t})},
\]  

(17)

where \( SE_{S_{t}}^{s} \) is the belief of state \( S \) at the last step, \( SE_{t}^{t} \) is the belief state vector at the last step, and \( P(o \mid a \land SE_{t}^{t}) \) is the normalizing factor.

The key idea is to compute a discrete approximation of a probability distribution over all possible poses in the environment. An important feature of this localization technique is the ability to localize the robot globally within the environment. More details about this approach can be found in [Cassandra et al., 1996].

In our approach, the set of observations \( O \) is composed of the fingerprints of places generated by the robot in the environment. These observations are very distinctive because of the design of fingerprints of places. An observation contains information given by the exteroceptive sensors and designates a subset of the world state. The information for the observation function \( OS \) within the topological framework is given by the fingerprint matching algorithm, global alignment with uncertainty.

The probabilistic observation function is given as follows:

\[
OS(o, s') = P([O^{t} = o] \mid [S^{t} = s']) = P([O^{t} = f_{obs}] \mid [S^{t} = f_{map\_i}]) =
\]

\[
= \frac{1}{Z_{Global\_Alignment\_Uncert}(f_{obs}, f_{map\_i})},
\]  

(18)
where $\textit{GlobalAlignmentUncert}$ gives the probability of matching two fingerprints and it is calculated using the global alignment with uncertainty algorithm described in Section 4.3. $Z$ is the normalization factor. $f_{\text{obs}}$ is the observed fingerprint, and $f_{\text{map}_i}$ is the map fingerprint, corresponding to node $i$. The normalization factor $Z$ is described as:

$$Z = \sum_{s \in S} OS(o, s).$$

(19)

5.3 Indoor control strategy

The computation of an optimal POMDP control strategy for large environments is computationally intractable. To obtain suboptimal solutions, simple heuristic control strategies have been proposed [Cassandra et al., 1996]. An example of such a strategy is the most likely state (MLS). This means that the world state $s$ with the highest probability is found and the action $a$ that is optimal for that state is executed. However, in this work, the entropy of the probability distribution over the states of the topological map is used. The entropy of a probability distribution $SE$ is:

$$H(SE) = - \sum_{s \in S} SE^t_S \log SE^t_S,$$

(20)

where $SE^t_S \log SE^t_S = 0$ when $SE^t_S = 0$. Lower values indicate more certain distributions. When the robot is confused, the entropy is high, so the POMDP is confident about its state if the entropy is smaller than a fixed threshold:

$$H(SE) < \Psi,$$

(21)

where $\Psi$ is the threshold calculated in the experiment. When the robot is confident, the action that is optimal for that state is executed. For instance, for indoor environments, if the POMDP is not confident about its state, the robot uses mid-line following if the preceding action was mid-line following and leave the room if the previous action was go to the centre of free space. The robot tries to reach and follow the corridor, where it expects to find more information. Mid-line following, leave the room and go to the centre of free space are simple indoor exploration tools that we have developed. More details can be found in [Tapus, 2005].

5.4 Map update

While navigating in the environment, the robot first creates and then updates the topological map. The entropy of a probability distribution (see Equation (20)) is used here. Therefore, the strategy for updating the map will be the following.
When the entropy of the belief state is low enough, the map will be updated and so the fingerprint and the uncertainty of the features will also be updated.

If the entropy is above the threshold, then updating will not be allowed, and the robot will try to reduce the entropy by continuing navigation with localization.

Similarly to Tomatis et al. [2003], when the robot feels confident concerning its state, it can decide whether an extracted feature is new by comparing the observed fingerprint with the fingerprint from the map corresponding to the most confident state. This can happen either in an unexplored portion of the environment, or in a known portion where new features appear because of environmental dynamics. The features from the fingerprint come with their extraction uncertainty $u_{\text{feature}}$. When a feature is re-observed, the uncertainty of the feature from the map fingerprint is weight averaged with the uncertainty of the extracted one. The weight depends on the type of feature. Because the extraction of features from the laser scanner is more reliable than those extracted with the camera, a higher weight is given to them. In our case, we choose to represent that as follows:

$$u_{\text{feature\_map}}^t = \begin{cases} u_{\text{feature\_map}}^{t-1} + u_{\text{feature}}^t, & \text{feature} \in \text{laser features} \\ \frac{2}{3}u_{\text{feature\_map}}^{t-1} + \frac{2}{3}u_{\text{feature}}^t, & \text{feature} \in \text{camera features} \end{cases}$$

(22)

Otherwise, if the robot does not see an expected feature, the uncertainty is decreased. The following equation expresses our choice for decreasing the uncertainty of a feature:

$$u_{\text{feature\_map}}^t = u_{\text{feature\_map}}^{t-1} - 0.1.$$  

(23)

When the uncertainty of a feature from a map fingerprint is below a minimum threshold, than the feature is deleted, allowing in this way for dynamics in the environment.

5.5 Closing the loop

One fundamental problem in SLAM is the identification of a place previously visited when the robot returns to it. This is known as the “closing-the-loop” problem, because the robot’s trajectory loops back on itself. For topological maps, this means that if a place (i.e. a node) has been visited before, and the robot returns to it, the robot should recognize it (see Figure 9).

In Thrun [1998] this was achieved by means of the EM algorithm, which ensures global consistency. The authors Choset and Nagatani [2001] use the structural characteristics of the map to determine the loop closing (i.e. the degree of vertices and the order of incident edges). Contrary to other methods used for solving this problem, which are usually based on perceptions, in
our approach, loops are identified and closed with the help of the localization method. To accomplish consistency of the topological map, a method similar to the one described in Tomatis et al. [2003] is used. In this work, the method employed is a nonexplicit loop-closing algorithm. Our loop-closing method is based on the localizer (i.e. POMDP). The robot is moving through the environment and incrementally builds the topological map. As soon as the robot returns to a previously visited place (i.e. node), the probability distribution should split. Two candidate hypotheses should appear: one for the new place (i.e. node) currently being created by the robot (e.g. in Figure 9, node Q) and another one for the previously created node already present in the map (e.g. in Figure 9, node A). As soon as the POMDP is not confident, the algorithm tracks the two highest probability distributions, showing that the distribution diverged in two peaks. A loop is thus identified if the probability distribution given by the localizer converges to two peaks. To detect where the loop was closed, the two hypotheses are backtracked with localization until a single one remains.

Fig. 9. The loop closing problem. The robot starts in place A and after moving through the environment arrives in place Q. The question to answer is: Has the robot returned to a previously visited place (i.e. is place A equivalent to place Q?)?

6 Experimental results

Our approach for topological SLAM using the fingerprint of places technique has been implemented and evaluated in various real-world indoor and outdoor
Both mobile platforms (indoor - BIBA robot and outdoor – “SMART” vehicle) are equipped with two 180° laser range finders and an omnidirectional camera. The omnidirectional camera system uses an equiangular mirror–camera system to image 360° in azimuth and up to 110° in elevation.

The first set of experiments demonstrates the robustness of the mapping module in two indoor scenarios and the first attempts to map urban outdoor environments. In particular, we illustrate the construction of distinctive and compact maps. They are composed only of local features, which is an important advantage of this fingerprint-based mapping technique. Localization, kidnapping and loop closing have also been tested and validated.

### 6.1 Indoor topological mapping

The first indoor experiment was conducted in a portion of our institute building shown in Figure 10, while the second experiment was performed in another building on the EPFL campus depicted in Figure 11. In the first test setup, the robot started at point $S$ and ended at point $E$, as illustrated in Figure 10, the distance travelled being 75 m. For the second test (see Figure 11), the robot travelled a distance of 67 m. While the robot explored the environment, it recorded, at every $\Delta d$ (in our case $\Delta d = 15$ cm), data readings from sensors (i.e. an image from the omnidirectional camera and a scan from the laser scanner), to extract the fingerprints. The robot used **mid-line following** behaviour in the hallways and **centre of the free space** behaviour in the open spaces.

We assume that the position in the room with the maximum free space around it is the one with the highest probability of extracting numerous and characteristic features [Lamon et al., 2003]. This ensures high distinctiveness of the observations. The threshold $\theta$, defined as the maximum allowable dissimilarity between fingerprints and used for automatic mapping, was evaluated experimentally. It was estimated from supervised experiments in a small portion of the environment (i.e. 5 m). Once this threshold was determined, it was fixed for the rest of the indoor experiments.

Figure 10 shows the topological map obtained by the system in the first test environment, superimposed on an architectural sketch of the environment. The resulting map is composed of 20 nodes as shown in the Figure 10. Each node is represented by a mean fingerprint that is an aggregation of all the fingerprints composing the respective node. Typically, the nodes are positioned in rooms and in hallways.

Four cases merit some additional discussion. The first special node is the one between Room 2 and Room 3. This node is justified because a door that connects the two hallways is present. A new node is introduced in the hallway

---

4 Two videos illustrating these experiments can be found at the following URLs: http://www.bayesian-programming.org/videoB1Ch5-1.html and http://www.bayesian-programming.org/videoB1Ch5-2.html
between Room 4 and Room 5. The robot detected important changes in the environment because of the vertical pillar present in the corridor. Another node that deserves attention is the hallway node between Room 7 and Room 8. The door of Room 8 is open (opening into the hallway), obstructing the view of the robot and making the environment very different in front of the door and behind it. A new node is therefore automatically introduced by the mapping system. The distance in the corridor between Room 8 and the end point $E$ is quite large. Because the robot detects distinguishing features caused by the changes in this portion of the environment, a new node specifying this is required. The doors of some rooms remained closed at the time of experimentation; this explains why no node is present in front of those rooms (see Figure 10). When the robot traversed the same path in the opposite direction (i.e. from $E$ to $S$), the same nodes were detected, and even if some doors previously open were closed, the robot succeeded in correctly localizing itself and updating the map accordingly.

Figure 11 shows the second test environment with the corresponding topological map, formed using the approach outlined in this work. The mapping system added a new node automatically each time a very distinctive measure (i.e. a distinctive fingerprint) was encountered. The graph-like map thus obtained contains eight nodes, as shown in Figure 11b. The same threshold used for the first test was also employed here, indicating the generality of the overall method. The representations obtained (see Figure 10 and 11) reproduce correctly the structure of the physical space, in a manner that is compatible with the topology of the environment.

### 6.2 First attempts at outdoor topological mapping

Compared with indoor environments, urban outdoor environments present many challenges for an autonomous vehicle. Coarse localization is often available from GPS. Most of the time, it is more useful to know the position of the robot with respect to buildings, trees, intersections and so on than its exact latitude and longitude. To validate our approach and to show its robustness, we also tested it in an outdoor environment. The approach was tested in a part of the EPFL campus (a highly structured environment), shown in Figure 12, on a trajectory 1.65 km long. The system mounted on the “SMART” vehicle acquired data from both the lasers and the omnidirectional camera every 110 ms. A new threshold for outdoor environments was established experimentally in a small portion of the campus. Different thresholds can be used as functions of the desired granularity of the environment. High-granularity maps, with numerous nodes, may be obtained by setting small thresholds. Alternatively, setting high values for the threshold yields maps with fewer nodes (low granularity). The outdoor threshold for obtaining high-granularity maps is the same as the one used for indoor environments. To create maps with fewer nodes, the outdoor threshold is set three times larger than the
Fig. 10. (a) Floor plan of the first environment in which the experiments were conducted. The robot starts at point $S$ and ends at point $E$. The trajectory length is 75 m. During this experiment, the robot collected 500 data sets (i.e. images and scans) from the environment. The extracted topological map is superimposed on an architectural sketch of the environment. (b) The extracted topological map given by our method, superimposed on the raw scan map.
Fig. 11. (a) The second test environment with the trajectory travelled by the robot. (b) The map of the second test environment with the graph representing the topological map.

Fig. 12. (a) The outdoor test environment (part of the EPFL campus) with the 1.65 km trajectory travelled by the SMART vehicle. The magnifying glass shows the part of the environment used for the outdoor topological map exemplification. (b) The low-granularity outdoor topological map superimposed on an architectural sketch of a part of the EPFL campus.
indoor threshold. We obtained a map consisting of 209 nodes for high granularity and one of 64 nodes containing only the big changes in the environment (i.e. intersections and new buildings). A small example is depicted in Figure 12(b), which represents a low-granularity topological map obtained for a 200 m section of the environment (i.e. the zoomed view of the magnifying glass shown in Figure 12(a)). The map contains seven nodes. It can be noticed that the nodes are usually placed in front of buildings, at crossings and when “big” changes occur, e.g. a building disappears from the field of view of the vehicle, and driving signs, lamp posts and trees appear.

6.3 Indoor global localization with POMDP

The quality of the topological maps obtained with our fingerprint-based technique can be evaluated by testing the localization on them. To test localization, more than 1000 fingerprint samples, acquired while the robot was travelling a new path of 250 m, were used to localize the robot globally with the POMDP. A mission is considered successful if the classified place, which corresponds to the world state with the highest probability, is the same as the correct location in the real-world environment.

The results are summarized in Table 1. Notice that POMDP localization results in a 100% success rate whereas matching without POMDP is limited to 81%.

The kidnapping problem (i.e. recovering from a lost position – the robot thinks that it is in a position where it is not) has also been tested. This was performed seven times, and the robot recovered successfully all seven times after one or two steps because of the very distinctive observations that correspond to the fingerprints of places.

The robot also succeeded in closing all loops. As explained earlier, because the offices are quite small, the fingerprints of places are very similar, and thus a single node per room is enough. Because a node contains posterior knowledge about its environment that is the aggregation of all the fingerprints of places between the last node and the current place where an important change in the environment occurs, the closing-the-loop problem does not appear in these cases (i.e. when one node per office is sufficient).

Table 1. Summary of the indoor localization experiments.

<table>
<thead>
<tr>
<th></th>
<th>1024 samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fingerprints</td>
<td></td>
</tr>
<tr>
<td>Distance Travelled</td>
<td>250 m</td>
</tr>
<tr>
<td>Scenarios</td>
<td>10/10</td>
</tr>
<tr>
<td>Kidnapping</td>
<td>7/7</td>
</tr>
<tr>
<td>Loop Closing</td>
<td>5/5</td>
</tr>
<tr>
<td>Fingerprint Matching</td>
<td>81%</td>
</tr>
<tr>
<td>POMDP Localization</td>
<td>100%</td>
</tr>
</tbody>
</table>
7 Fingerprints of places: A model of hippocampus place cells

In all our daily activities, the space in which we live and move plays a crucial role. Neurobiologists are making great efforts to understand the behaviour of animals and their capacity to learn and use their knowledge of the spatial representation to navigate.

The seminal discovery of place cells, by O’Keefe and Dostrovsky [1971], in the rat hippocampus – cells with firing patterns that are dependent on the location of the animal in the environment – led to the idea that the hippocampus works as a cognitive map of space [O’Keefe and Nadel, 1978]. It was shown in Cho et al. [1998] (for a review see e.g. Redish [1999]) that a lesion of the hippocampus impairs the performance of rodents in a wide variety of spatial tasks, indicating a role for the hippocampus in map-based navigation.

We strongly posit that the framework for topological SLAM proposed in this work organizes spatial maps in cognitive graphs, with nodes corresponding to fingerprints of places, and may be seen as a possible mechanism for the emergence of place cells. Our computational model describes how a mobile agent can efficiently navigate in the environment, by using an internal spatial representation (similar to some extent to hippocampus place cells).

![Fig. 13. The links between the hippocampus and other areas of the brain.](image)
This model builds a topological (qualitative) representation of the environment from the sequence of visited places. Many vision-based systems for place fields using metric information have been extensively discussed in the literature: ([Arleo and Gerstner, 2000], [Hartley et al., 2000] and [Kali and Dayan, 2000] are just some of them).

It is possible to see all through this work that a fingerprint is associated with each distinctive place within the environment. Thus, the result given by the fingerprint matching algorithm is strongly correlated (linked) to the location of the mobile agent in the environment, giving high or the highest probability to the correct place associated with the fingerprint. The firing of place cell units can be seen as the manifestation of fingerprint matching.

An animal closer to the centre of the place field has a higher rate of neural firing. Similarly, if a new observation by the robot (i.e. a new observed fingerprint) more closely resembles a registered (learned) place (i.e. a known fingerprint), the mobile agent has a higher probability of being in a previously explored place.

The methodology presented in this paper can efficiently create representations of places in an environment and locate the robot or animal in the environment. The place cells in the hippocampus accomplish the same task: the activation of a place cell, or perhaps better, of an assembly of place cells connected to each other, indicates that the hippocampus is locating the animal in a certain place. It can be suggested here that the hippocampus may indeed extract place information from its sensory input by constructing fingerprints of places, similarly to the method described in this work (see Figure 13). It is known that in environments rich in landmarks, or features, the hippocampus cognitive map is dominated by the sensory inputs (see e.g. [Gothard et al., 1996], [Battaglia et al., 2004]). Changing the relative position of landmarks can cause a complete change in place cells’ activity (“remapping”), so that a new set of place cells is assigned to a given place, just as would be the case with our fingerprint algorithm [Cressant et al., 2002]. Many theoreticians have proposed models of place cells based on visual inputs, where the visual stream is encoded in metric terms, that is, in terms of the distances between the landmarks, and between each landmark and the agent (e.g. [Arleo and Gerstner, 2000], [Hartley et al., 2000] and [Kali and Dayan, 2000]). Fingerprint representations are based on the relative angular position of the landmarks from a given point of view, a much simpler and more robust measure, and may be able to explain much of the experimental evidence on place cells, at least that in which multiple landmarks were available to the animal.

For the brain to perform fingerprint matching, several building blocks are necessary: first, the identification of the landmarks, which may take place, for example, in the inferotemporal cortex, and second, the determination of the relative position of multiple landmarks, which probably takes place in the parietal lobe [Cressant et al., 2002]. The hippocampus may gather this information and produce a unitary representation (which would correspond to a fingerprint), presumably in terms of an attractor configuration of the
CA3 module (which is very prevalent in recurrent synaptic connections and is thought to work as an attractor network module). At the moment of localization, the current input may be fed into the attractor dynamics, and, if the fingerprint matches one of the previously stored ones, the corresponding attractor is recalled. For a no-match, the attractor dynamics will not produce an attractor state, and this fact may be used to signal a novel situation and trigger the plasticity processes that allow the storage of a new memory.

This vision of hippocampus space representations highlights the role of the hippocampus as a processor of combinatorial information. Its importance transcends the purely spatial domain. In space computation, the hippocampus would process combinations of landmark identity and relative position information and produce an index that could be attached to a physical location.

It is important to mention here that in our scheme, the place representation does not entail any notion of Euclidean space, contrary to the hypotheses in O’Keefe and Dostrovsky [1971] and in a number of more recent works (see the review in [Redish, 1999]). In our view, the computation of places from sensory input (through a fingerprint-like procedure), is integrated by idiothetic information, which plays an important role, especially in conditions in which only poor sensory input is available (for example, in the dark) and in disambiguating situations of perceptual aliasing (see e.g. [Skaggs and McNaughton, 1998]).

The topological navigation framework based on fingerprints of places presented in this work is relevant to robotics, biology and neurophysiology. Our computational model finds a counterpart in neurobiology, being similar in function to the hippocampus, which plays a crucial role in spatial representation. The proposed spatial representation is an incrementally learned representation, based on fingerprints of places; the fingerprint place modelling is comparable with the place coding model in the animal’s (rat’s) hippocampus.

8 Conclusion and future work

This chapter has presented a new technique for topological SLAM using fingerprints of places. The fingerprint provides a compact and distinctive methodology for space representation and place recognition – it permits encoding of a huge amount of place-related information in a single circular sequence of features. We also presented our research framework, which is relevant to robotics, biology and neurophysiology. Our computational model has some foundation in neurobiology, resembling the activity of the hippocampus, which plays a crucial role in spatial representation. This kind of representation is suitable for both indoor and outdoor environments. The experiments verify the efficacy and reliability of our approach. The POMDP localization shown here improves on previous results. Adding the motion of the robot allows us to decrease the pose uncertainty to a level that could never be reached by fingerprint matching alone. A success rate of 100% was obtained for the tests
performed in this work. However, the approach must be tested more extensively in different types of environment to make a real estimation of the quality of the method. In this work, low-level features (such as vertical edges and horizontal lines) were used. An interesting extension of the model is the addition of other modalities and features to the fingerprint framework (e.g. auditory, olfactory, and higher-level features such as doors, tables and fridges). This will help to improve the reliability and accuracy of the method and to add semantics to it.

References


Topological SLAM


B. Lisien, D. Morales, G. Silver, D.and Kantor, I. Rekleitis, and H. Choset. Hierarchical simultaneous localization and mapping. In *IEEE/RSJ Inter-


Probabilistic contextual situation analysis

Guy Ramel$^1$ and Roland Siegwart$^2$

$^1$ Autonomous Systems Lab, Ecole Polytechnique Fédérale de Lausanne
$^2$ Autonomous Systems Lab, Institute of Robotics and Intelligent Systems, ETH Zürich

Mobile robots are gradually appearing in our daily environment. To navigate autonomously in real-world environments and interact with objects and humans, robots face various major technological challenges. Among the required key competencies of such robots is their ability to perceive the environment and reason about it, to plan appropriate actions. However, sensory information perceived from real-world situations is error prone and incomplete and thus often results in ambiguous interpretations. We propose a new approach for object recognition that incorporates visual and range information with spatial arrangement between objects (context information). It is based on using Bayesian networks to fuse and infer information from different data. In the proposed framework, we first extract potential objects from the scene image using simple features—characteristics like colour or the relation between height and width. This basic information is easy to extract but often results in ambiguous situations between similar objects. To resolve ambiguities among the detected objects, the relative spatial arrangement (context information) of the objects is used in a second step. Consider, for example, a cola can and a red trash can that are both cylindrical, have similar ratios between width and height and have very similar colours. Depending on their distances from the robot, they may be hard to distinguish. However, if we further consider their spatial arrangement with other objects, e.g. a table, they might be clearly differentiable, the cola can typically standing on the table and the trash can on the floor. This contextual information is therefore a very efficient way to increase drastically the reliability of object recognition and scene interpretation. Moreover, range information from a laser scanner and speech recognition offer complementary information to improve reliability further. Thus, an approach using laser range data to recognize places (such as corridors, crossings, rooms and doors) using Bayesian programming is also developed for both topological navigation in a typical indoor environment and object recognition.

The proposed approach is verified through different real-world experiences. The characteristics and typical spatial arrangements of the objects in various test scenarios are first used to train a Bayesian network through a series of
example images (learning) and then verified on the test images. By fusing the extracted object probability from images and range data with spatial relations among the objects, ambiguities are reliably solved, and the reliability of the detection is drastically increased. The results show the validity and performance of the proposed Bayesian approach, which combines context information with simple object recognition.

1 Introduction

Unlike the robots in literature and fiction films, which seem to carry out all their tasks more effectively than human beings, real robots encounter a very different reality. These robots are confronted with a major problem: the real world cannot be apprehended in all its details, nor with certainty.

1.1 Perception and uncertainty

One of the principal (and necessary) functions of an autonomous system – alive or artificial – is the perception of its environment. Intended to provide all the necessary information for the execution of the tasks for survival or the resolution of problems, this perception can take place through various modalities: sight (eyes or camera), hearing (ears or microphone), touch (skin or force sensor) or sense of smell, sonar, electric or magnetic field sensors in certain animals, and laser sensors for certain robots.

Each one of these sensors can provide only an incomplete description of the environment. This aspect interferes in the representation of the world and is considered in Lebeltel et al. [2004], Lebeltel [1999].

This problem of the impossibility of acquiring perfect or complete information on the environment imposes a strong constraint on the choice of strategies of inference for an autonomous robot functioning in an unstructured environment and interacting with humans.

1.2 Probabilistic inference

To infer from sets of data, several methods can be used: symbolic models that use, for example, graphs or formal logic paradigms Caplat [2002], Delessert [1988]; neuromimetic models that copy natural systems (neural networks, genetic algorithms) Beal and Jackson [1990], Davalo and Naïm [1990], Freeman and Skapura [1991]; and probabilistic models such as Kalman Filters and Extended Kalman Filters ((E)KFs) Welch and Bishop [2005], Hidden Markov Models (HMMs) Rabiner [1989], Bayesian networks (BNs) Becker and Naïm [1999] or Bayesian programming (BP) Lebeltel et al. [2004].

The approaches used in this chapter to cope with uncertainty and incompleteness belong to probabilistic models; they are Bayesian programming and
Bayesian networks. This choice follows from several observations. First, the world is too complex to apprehend all possible situations and predict them with the programming of a scenario or a classical graphic model. Second, algorithms based on logic paradigms (first-order logic for example) are less useful with uncertain and incomplete data. Third, neuromimetic models such as artificial neural networks (ANN) could be unwieldy for complex tasks. In particular, they contain hidden nodes that make the ANNs very powerful but limit possible semantic interpretation.

Thus, Bayesian networks, which belong to probabilistic methods, are a good compromise to cope with uncertainties and incomplete data. They do not have hidden variables, and they are a causal representation of a problem, giving the programmer a useful way to describe a problem. Moreover, Bayesian networks allow learning.

1.3 About this work

The work described in this chapter belongs to the field of autonomous mobile robotics and of human–robot interactions. In this kind of application, robots are confronted with several types of uncertainties, as enumerated in the following list.

- Consider two objects looking like each other, such as a cola can and a red trash can. They have the same geometrical characteristics and the same colour. With mono vision and according to their distance, they can have the same appearance, and the robot may not be able to differentiate them. Nevertheless, if we consider the relation between these objects and another object such as a table, we can obtain new information called context information: the object is on the floor, the object is on the table. This information will allow the system to differentiate the can from the trash can.

- In several cases, an object will not be correctly detected. For example, under some luminosity conditions, an object will be detected only partially. Hence, the robot will not be able to recognize it. In this kind of situation, context information such as the position within the surroundings (on the floor, at a certain height, ...), the apparent size and the distance to the robot will provide the necessary data to recognize the object.

- Some objects, for example the human body, are too complex to be correctly detected and recognized. Nevertheless, the system can easily detect several candidates for the face, the hands or the arms. By comparison, between these candidates, the system will be able to determine the nature of the candidates and consequently decide whether a human is visible.

This work develops a new approach using Bayesian networks to improve object recognition and scene understanding by merging two kinds of variables. The first kind provides intrinsic information on a detected feature. These variables could be: colour, ratio between height and width, or shape. In other
terms, these are all variables depending on the detected object only. The second category of variables provide context information related to the position of the feature in space and its relationship with other features.

**System description**

The complete system is described in Figure (1). In this figure, five layers are depicted.

- The first layer shows us the modalities used to extract features: the laser scanners to detect the table, the camera to extract all areas of colour that could be interesting objects and the control systems for the camera.
- The second layer corresponds to the preprocessing of the raw data coming from the first layer. It gives the orientation of the camera obtained from the controllers of the motors, the colour-based features detection with the camera and the candidates for tables with the laser scanners.
- The third layer is dedicated to the sensor fusion between the laser scanners and the camera.
- The fourth layer contains the high-level inference using Bayesian networks.
- The last layer consists of all possible actions that the robot could perform with the knowledge of the result.

**1.4 State of the art**

The growing fields of autonomous mobile robotics and human–robot interaction have motivated many studies. In Perzanowski et al. [1998], Perzanowski et al., the authors cleared up an ambiguity such as that in the sentence “go to the waypoint over there” by using other information obtained from gesture recognition Kortenkamp et al. [1996], Pavlovic [1999], Perzanowski et al. [1998], Perzanowski et al.. An auditory signal is converted to a sequence of characters, which is analysed by a natural language processing system called NAUTILUS. Next, gesture recognition gives new information to clarify the sentence. In McGuire et al. [2002], the authors allow an ordinary user to program the robot by demonstrating the tasks. For this, they use a hybrid architecture consisting of statistical methods, a Bayesian network and a finite state machine (FSM). They use the Bayesian network to merge the vision and the speech to understand a sentence such as “the long thin stick” or “the object to the left of the cube”. In Pavlovic [1999], dynamic Bayesian networks are used to analyse and recognize gestures and speech. This allows an operator to control a screen.

Other studies have used context to improve object recognition. In Torralba [2003], Torralba et al. [2003], the authors used common locations (conference room 941, office room 610, main street, ...) to characterize new surroundings (office room, street, ...). These pieces of information were then used to recognize objects: a robot is more likely to find a table in an office room than in
Fig. 1. Description of the complete inference system.

The work presented in this chapter also uses context to understand a scene. In this case, a causal model representing the context of use of an object is realized with a Bayesian network, allowing the robot to learn some characteristics of the context (relationship between objects, relationship between object and the robot) for each object Ramel et al. [2006], Ramel [2006].
Contribution

The main contribution Ramel et al. [2006], Ramel [2006] of this work is in two parts:

- the use of the context of an object to improve its recognition; and
- the use of Bayesian networks to integrate these pieces of contextual information.

One could compare this approach with the high-level reasoning of humans to compensate for low-level feature recognition when they are subjected to illusions or difficult luminosity conditions. These illusions are all the more disabling for robots, such that their sensors and preprocessing are less efficient than those of humans. This work illustrates that information that does not depend on intrinsic characteristics of objects can strongly improve object recognition and scene analysis.

Some additional contributions were made. Table recognition using laser scanners was developed Jensen et al. [2003], Ramel [2006]. Merging this recognition with visual table recognition increases confidence that a real table is present. A second contribution from using laser scanners is topology recognition for locations and recognizing a particular room employing Bayesian programming Tapus et al. [2004]. This can be useful to obtain new information to improve object recognition (it is more probable to find a table in an office room than on the street) Torralba [2003], Torralba et al. [2003]. Finally, a third contribution consists of recognizing elementary features of an image using Bayesian programming Ramel et al. [2006], Ramel [2006].

2 Bayesian networks

The aim of this section is to cover briefly what Bayesian networks are and how they work. Those who want more detail can refer to the literature Becker and Naïm [1999], Jensen [1996], Lauritzen [1996], Pearl [1988], Russel and Norvig [1995].

One important aim of a system interacting with the environment is of course to acquire information and, especially, to analyse and synthesize this information; in other words, to acquire and to use information. Two approaches are commonly used: statistical methods, which allow transition from observations to laws; and artificial intelligence methods, which try to handle knowledge.

Bayesian networks are derived from studies having the goal of integrating the notion of uncertainties into expert systems. Such systems must almost always take uncertainties into account.

Figure (2) shows an example of a Bayesian network. It consists of four nodes: $A$ is the proposition “I forgot to stop the sprinkler”; $P$ is “it rained last night”; $J$ is “the grass in my garden is wet”; $V$ is “the grass in my neighbour’s
Each of these nodes is associated with an aleatory variable (we use the same name for the node and the variable) and its probability distribution. In this example, variables are discrete and binary. The Bayesian network therefore corresponds to a graphical representation of causality between these four variables.

\[ P(Y_1 \land \cdots \land Y_n) = \prod_{i=1}^{n} P(Y_i \mid C(Y_i)), \]  

where \( Y_i \) is the variable corresponding to the node \( i \) and \( C(Y_i) \) is the set of the parents of \( Y_i \).

Then we obtain:

\[ P(A \land P \land J \land V) = \begin{vmatrix} P(A) \\ P(P) \\ P(J \mid A \land P) \\ P(V \mid P) \end{vmatrix}. \]  

To fix the conditional probability distribution of each node, an identification phase is required. In simple cases, we can calculate this distribution, but usually we must train the Bayesian network with a training database of examples. After the training, we can use the Bayesian network to determine the probability distribution of searched nodes knowing some observed nodes. For example:

\[ P(P \mid J \land V). \]  

### 2.1 Bayesian networks vs Bayesian programming

To compare Bayesian networks with Bayesian programming formalism, the Bayesian network can be represented by the following Bayesian program.

In the decomposition, we recognize the joint probability defining a Bayesian network (Eq. 1).
3 Situation analysis and experiments

In the following sections, we describe our experiments and the results obtained. In these results, Bayesian networks were used to build a causal representation of the relationships between several variables: one representing the object to be recognized, the intrinsic variables (depending directly on the object), and the context variables (depending on the usual use of the object).

3.1 Goal and experimental protocol

To determine the effect of using context, we first conducted a reference experiment. This reference experiment consisted of training and using a Bayesian network without context variables. The comparison between this reference experiment and the later ones allows us to validate the usefulness of the context in recognizing an object and analyzing a scene.

For all these experiments, a variable corresponding to the object to be recognized and two other variables corresponding to intrinsic characteristics are used: the colour given by the value \(v\) component of the Hue–Saturation–Value (HSV) space, and the ratio between the height and the width \(h/w\) of the detected object. These two variables proved to be sufficient to validate the method. The component \(v\) of the HSV space is less sensitive to luminosity variations, and the \(h/w\) variable is independent of distance. This choice of simple variables allows simple, fast feature extraction.

In later experiments, several variables that relate to the context were added. The experiments using the context are the following.

- The first experiment uses as context variables the position of the object relative to another detected feature (called the reference object) in the same way as its absolute position relative to the robot. Other variables are the apparent size and distance from the robot. An example is a red cola.
can and a red trash can (Fig 4). These two objects resemble each other, and depending on their distance from the robot, they could be confused. However, if the feature is on a table, it is probably the cola can, and if it is on the floor, it is probably the trash can.

- The second experiment is similar to the first but uses only the absolute position relative to the robot without the position relative to a reference object. This allows the system to recognize an isolated object.
- The third experiment approaches the recognition of objects constituted by several parts. The relationships between these different parts depend on the object itself. Recognition of these parts can improve the recognition of the object. An example is the recognition of a human by recognizing the face, the arms and the hands.

![Fig. 4](a) Shows the result of the detection of a red cola can, of a trash can and of human body components. In this image, the cola can and the trash can are easily distinguishable by their sizes. (b) The two objects look the same, and it is difficult to distinguish them.

### 3.2 Reference experiment

The Bayesian network of Figure (5) illustrates this experiment. It consists of three nodes. Each node is associated with a variable having the same name. These three variables are as follows.

- $Obj_2$: the object we wish to identify. This is a discrete variable corresponding to known objects: trash can, table, cola can and so on.
- $h/w_2$: the ratio between the height and the width (in pixels), on the photo, of the detected object $Obj_2$.
- $Col_2$: the colour of $Obj_2$ given by the mean value of the ($v$) component of the $HSV$ space.

The relationships of these variables are given by the preliminary knowledge of dependencies: $h/w_2$ and $Col_2$ both depend on the variable object $Obj_2$. This
Fig. 5. The Bayesian network for the reference experiment. It consists of three nodes: the object to recognize ($Obj_2$), the colour of this object ($Col_2$), and a geometrical characteristic ($h/w_2$).

is expressed by the graphical representation of Figure (5) and by the following decomposition of the joint probability.

$$P(Obj_2 \wedge h/w_2 \wedge Col_2) = \frac{P(Obj_2|D \wedge C)}{P(h/w_2|Obj_2 \wedge D \wedge C)} \frac{P(h/w_2|Obj_2 \wedge D \wedge C)}{P(Col_2|Obj_2 \wedge D \wedge C)}$$ (4)

In this equation, the variables $D$ and $C$ represent the experimental data and the preliminary knowledge. For each of these elementary distributions, it is necessary to choose a parametric form. The experimental data and preliminary knowledge enable us to do so. By combining many measures of these probability distributions into a histogram, we can determine these parametric forms. For example Figure (6) depicts the Gaussians obtained by iterations from histograms of the colour variable for the table, the cola can and the trash can.

Fig. 6. Probability distributions of the colour variable ($Col_2$) for three objects: the cola can; the trash can; and the table. Recall that variable $Col_2$ corresponds to the average of the $HSV$ space value component of the detected feature.
A Gaussian is also obtained for the geometrical variable \((h/w_2)\). These two variables are continuous. For the variable object \((\text{Obj}_2)\), we choose a uniform distribution, and it is a discrete variable. We obtain:

\[
\text{Obj}_2 = i
\]
\[
i \in \{1, 2, 3, \ldots, N_{\text{Obj}_c}\} = \{\text{cola can, trash can, table, \ldots}\},
\]
where \(N_{\text{Obj}_c}\) is the number of known objects.

\[
P(\text{Obj}_2 | D \land C) = U = \frac{1}{N_{\text{Obj}_c}} \tag{5}
\]

\[
P(h/w_2 | \text{Obj}_2 = i \land D \land C) = G_{\mu(i), \sigma(i)}(h/w_2) \tag{6}
\]

\[
P(\text{Col}_2 | \text{Obj}_2 = i \land D \land C) = G_{\mu(i), \sigma(i)}(\text{Col}_2) \tag{7}
\]

The Gaussian distribution provides a satisfactory representation of variables and is simple to use, but other better distributions may be better, for example, a mixture of Gaussians. It is not the purpose of this chapter to test other distributions.

**Identification**

After determining the structure of the Bayesian network and the parametric forms of all the variables, the parameters of these distributions must be evaluated. This is the aim of identification. Two methods can be used. In the first method, the parameters are calculated from the histograms derived from the experimental data. In the second method, the Bayesian network is trained with a set of examples. All experiments described in this chapter use learning to identify all parameters.

**Use**

Finally, applying the Bayesian network determines the probability distribution of node \(\text{Obj}_2\), knowing the two observed nodes \(h/w_2\) and \(\text{Col}_2\). This corresponds to the following question.

\[
P(\text{Obj}_2 | h/w_2 \land \text{Col}_2 \land D \land C) \tag{8}
\]

Thus, we obtain the probability \(p_i\) that the variable \(\text{Obj}_2\) has the value \(i\).

**Results**

Figure 7 shows six situations analysed by the Bayesian network. In the first two images, the cola can and the trash can have the same apparent size. In
Fig. 7. Several scenes with a table, a cola can and a trash can. In the first two pictures, the trash can and the cola can are of similar apparent size, and it is a challenge to distinguish them correctly. In the next two pictures, only part of the trash can is seen. In the last two pictures, the trash can is close to the robot, and its apparent size is bigger than that of the cola can.

In these tables, we see another category of objects called garbage model (G can, G tab, G door). This category corresponds to artefacts detected by the vision system that do not correspond to the known objects. Adding this garbage model category can be useful to avoid several confusions. In fact, we can avoid detecting artefacts in the environment. For example, in Figure (7), parts of the chair are detected by their red colour.

Table (2) shows many confusions between trash can and cola can (43.9% et 28.6%), but the Bayesian network correctly recognizes the tables (94.7%). This confusion can be explained by the shape and colour similarities between the trash can and the cola can, while the tables are very different.

In Table (3), we can observe the poor recognition of cola cans (74.6%), windows (36%) and doors (39.3%), and a worse recognition of the tables (90.7%) than previously (94.7%). In fact, the great diversity of possible ways to view a window, a door or a table (partially or not, from far or from near, saturated
Table 1. Configuration of the database used to test the Bayesian network. The column No. tot gives the number of examples containing the feature of the first column.

<table>
<thead>
<tr>
<th></th>
<th>No. tot</th>
<th>No. learn</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tables</td>
<td>531</td>
<td>100</td>
</tr>
<tr>
<td>Cola can</td>
<td>153</td>
<td>100</td>
</tr>
<tr>
<td>Windows</td>
<td>722</td>
<td>100</td>
</tr>
<tr>
<td>Doors</td>
<td>435</td>
<td>100</td>
</tr>
<tr>
<td>G can</td>
<td>6063</td>
<td>100</td>
</tr>
<tr>
<td>G tab</td>
<td>3145</td>
<td>100</td>
</tr>
<tr>
<td>G door</td>
<td>209</td>
<td>100</td>
</tr>
<tr>
<td>Total</td>
<td>11258</td>
<td>700</td>
</tr>
</tbody>
</table>

Table 2. Results of the reference experiment for the recognition of tables (reference object) and of two objects with the same appearance: cola cans and trash cans. The columns correspond to the real objects, the rows to the recognized objects.

<table>
<thead>
<tr>
<th></th>
<th>Tables</th>
<th>Cola cans</th>
<th>Trash cans</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tables</td>
<td>0.947</td>
<td>0.000</td>
<td>0.026</td>
</tr>
<tr>
<td>Cola cans</td>
<td>0.009</td>
<td>0.561</td>
<td>0.286</td>
</tr>
<tr>
<td>Trash cans</td>
<td>0.044</td>
<td>0.439</td>
<td>0.688</td>
</tr>
</tbody>
</table>

Table 3. Results of the reference experiment for the recognition of tables, cola cans, windows and doors. The columns correspond to the real objects, the rows to the recognized objects.

<table>
<thead>
<tr>
<th></th>
<th>Tables</th>
<th>Cola cans</th>
<th>Windows</th>
<th>Doors</th>
<th>G tab</th>
<th>G can</th>
<th>G dor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tables</td>
<td>90.7%</td>
<td>5.6%</td>
<td>17.9%</td>
<td>1.3%</td>
<td>36.3%</td>
<td>48.1%</td>
<td>15.3%</td>
</tr>
<tr>
<td>Cola cans</td>
<td>3.6%</td>
<td>74.6%</td>
<td>35.3%</td>
<td>6.7%</td>
<td>26.6%</td>
<td>19.7%</td>
<td>0.6%</td>
</tr>
<tr>
<td>Windows</td>
<td>7.4%</td>
<td>16.7%</td>
<td>36.0%</td>
<td>0.2%</td>
<td>19.1%</td>
<td>27.1%</td>
<td>0.0%</td>
</tr>
<tr>
<td>Doors</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>G tab</td>
<td>0.0%</td>
<td>0.7%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>15.9%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>G can</td>
<td>5.0%</td>
<td>2.4%</td>
<td>10.8%</td>
<td>48.5%</td>
<td>2.1%</td>
<td>5.1%</td>
<td>55.3%</td>
</tr>
<tr>
<td>G dor</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>3.9%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>29.5%</td>
</tr>
</tbody>
</table>

3.3 Improvement by using context

Two Bayesian networks are used in this experiment. The first network is depicted in Figure (8 (a). It has nine nodes. Nodes Obj2, h/w2 and col2 are the
Fig. 8. (a) The first Bayesian network, intended to improve object recognition by using the topological relations between objects and their position relative to the robot. (b) The second Bayesian network, intended to improve object recognition by using only the situation of the object in space relative to the robot. This network therefore does not require a reference object: the relations $P(Obj_2 | Obj_1 \land D \land C)$ and $P(RPos | Obj_1 \land D \land C)$ are suppressed.

same as for the reference experiment. Nodes $Obj_1$, $h/w_1$ and $col_1$ have the same descriptions as nodes $Obj_2$, $h/w_2$ and $col_2$ but correspond to an object detected and used as reference. For example, if we detect a table and a red cylinder to identify, we can use the table as a reference to compare with the red cylinder. The other nodes correspond to the context variables. The second Bayesian network is shown in Figure (8 (b). In this network, we do not use the topological comparison with a reference object. Suppressing the use of a reference object is necessary to be able to recognize an isolated object. In fact, sometimes we can detect only one object. Moreover, many objects, such as windows, are not necessarily associated with another object. The descriptions of all the variables are given below.

$Obj_2$: the object that we wish to identify.
$Obj_1$: a reference object to compare with the object that we wish to identify.
$RPos$: the relative position of the object to identify $Obj_1$ relative to the reference object $Obj_1$. This variable is the normalized difference between the vertical positions, in the picture, of these objects. This variable is shown in Figure (9 (a) and by the equation:

$$RPos = 2 \frac{(Vert\_Pos_{Obj_2} - Vert\_Pos_{Obj_1})}{Haut_{Obj_2} + Larg_{Obj_2}}.$$  

(9)

$VPos$: the absolute vertical position of object $Obj_2$ given by the angle between the horizontal and the line joining the “eyes” of the robot and the object (Figure 9 b). The unit is degrees. This variable is calculated with the equation:

$$VPos = \alpha + \frac{O_y}{H} \cdot Ap,$$

(10)
where $H$ is the height of the image (pixel) and $Ap$ is the vertical aperture of the camera.

**AS:** the apparent size of $Obj_2$ in the image in pixels. This is the average of the length and the width of the feature.

$h/w_i$: the ratio between the height and the width, in pixels, on the image, of the detected object $Obj_i$, where $i \in \{1, 2\}$.

$Col_i$: the colour of $Obj_i$, where $i \in \{1, 2\}$, given by average value of the $(v)$ component of the $HSV$ space.

\[ dy = Vert_{Pos_{Obj_2}} - Vert_{Pos_{Obj_1}} \]

**Fig. 9.** (a) Here one can see the description of the aleatory variable $RPos$, with $dy = Vert_{Pos_{Obj_2}} - Vert_{Pos_{Obj_1}}$. (b) Description of the calculation of the variable $VPos$. $Ap$ the camera aperture, $\alpha$ the gaze direction of the camera relative to the horizontal line, $H$ is the half height of the picture in pixels, and $\overrightarrow{Oy}$ is the vertical coordinate of the object in the picture in pixels. Warning: the projection of the picture is not in the focal plane.

The decomposition of the joint distribution of these two Bayesian networks is given by the two equations below.

\[
P(Obj_1 \land h/w_1 \land Col_1 \land Obj_2 \land h/w_2 \land Col_2 \land RPos \land VPos \land AS)
\]

\[
= P(Obj_1 | D \land C)
+ P(Obj_2 | Obj_1 \land D \land C)
+ P(h/w_1 | Obj_1 \land D \land C)
+ P(Col_1 | Obj_1 \land D \land C)
+ P(h/w_2 | Obj_2 \land D \land C)
+ P(Col_2 | Obj_2 \land D \land C)
+ P(RPos | Obj_1 \land Obj_2 \land D \land C)
+ P(VPos | Obj_2 \land D \land C)
+ P(AS | RPos \land VPos \land Obj_2 \land D \land C)
\]

(11)
\[
P(\text{Obj}_2 \wedge h/w_2 \wedge \text{Col}_2 \wedge \text{RP} \wedge \text{VPos} \wedge \text{AS}) = \]
\[
\begin{align*}
P(\text{Obj}_2 | D \wedge C) \\
P(h/w_2 | \text{Obj}_2 \wedge D \wedge C) \\
P(\text{Col}_2 | \text{Obj}_2 \wedge D \wedge C) \\
P(\text{RP} | \text{Obj}_2 \wedge D \wedge C) \\
P(\text{VPos} | \text{Obj}_2 \wedge D \wedge C) \\
P(\text{AS} | \text{RP} \wedge \text{VPos} \wedge \text{Obj}_2 \wedge D \wedge C)
\end{align*}
\]

To choose the parametric form of each term, the same reasoning, based on histograms, was made as in the reference experiment. Except for the nodes \(\text{Obj}_i\), all variables can be represented by Gaussian distributions. Mixture models may be better for some variables, but simple Gaussian distributions were sufficient to validate our approach. Moreover, a Gaussian distribution is very simple to implement. We therefore obtain the following.

\[
P(\text{Obj}_1 | D \wedge C) = U = \frac{1}{N_{\text{Obj}_c}}\]

\[
P(\text{Obj}_2 | \text{Obj}_1 \wedge D \wedge C) = U = \frac{1}{N_{\text{Obj}_c}}\]

\[
P(\text{Col}_i | \text{Obj}_i = k \wedge D \wedge C) = G_{\mu(k),\sigma(k)}(\text{Col}_i)\]

\[
P(h/w_i | \text{Obj}_i = k \wedge D \wedge C) = G_{\mu(k),\sigma(k)}(h/w_i)\]

\[
P(\text{VPos} | \text{Obj}_2 = j \wedge D \wedge C) = G_{\mu(j),\sigma(j)}(\text{VPos})\]

\[
P(\text{RP} | \text{Obj}_1 = i \wedge \text{Obj}_2 = j \wedge D \wedge C) = G_{\mu(i,j),\sigma(i,j)}(\text{RP})\]

\[
P(\text{AS} | \text{RP} = i \wedge \text{VPos} = j \wedge \text{Obj}_2 = k \wedge \text{Dist} = l \wedge D \wedge C) = G_{\mu(i,j,k,l),\sigma(i,j,k,l)}(\text{AS})\]

**Remark**

Generally, \(\text{Obj}_1\) is not an observed variable. It is known by the Bayesian network only through the variables \(h/w_1\) and \(\text{col}_1\). Nevertheless, the robot can use the laser scanner to detect and recognize tables, so the node becomes observed, and the belief in the presence of a table is increased. This emphasizes the usefulness of sensor fusion.

**Identification**

Similarly to the reference experiment, we use a database of examples to train the Bayesian network. From these examples, the network determines all the parameters of the distributions by using the *Maximum Likelihood parameter Estimation*. 
Use

Finally, the use of these Bayesian networks determines the probability distribution

\[ P(Obj_2|Obj_1 \land h/w_1 \land Col_1 \land h/w_2 \land Col_2 \land RPos \land Vpos \land AS \land D \land C) \]  \hspace{1cm} (20)

for the first Bayesian network using comparisons between an object to recognize and a reference object, and

\[ P(Obj_2|h/w_2 \land Col_2 \land RP os \land Vpos \land AS \land D \land C) \]  \hspace{1cm} (21)

for the second Bayesian network without a reference object.

Results

Two results are presented in this section. The first result is the one obtained by the Bayesian network of Figure (8 (a) (with reference object). Its goal was to recognize the table, the cola can and the trash can (Fig. 7), and the results are shown in Table (4). By comparison with the reference experiment (Table 2), we remark that the recognition of the trash cans and the cola cans is strongly improved, reaching 92.6\% for the cola cans and 98.0\% for the trash cans. As for the tables, they are still well recognized.

<table>
<thead>
<tr>
<th>Table</th>
<th>Cola can</th>
<th>Trash can</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table</td>
<td>0.977</td>
<td>0.003</td>
</tr>
<tr>
<td>Canette</td>
<td>0.007</td>
<td>0.926</td>
</tr>
<tr>
<td>Poubelle</td>
<td>0.016</td>
<td>0.701</td>
</tr>
</tbody>
</table>

\textbf{Table 4.} Results of the experiment with spatial context (and reference object) for the recognition of tables (reference object) and of two objects with the same appearance: cola cans and trash cans. The columns correspond to the real objects, the lines to the recognized objects.

The second results are those obtained by the Bayesian network of Figure (8 (b) (without reference object). The goal was to recognize the table, the cola can, the windows and the doors. The results are shown in Table (5). There too the results are strongly improved.

These results suggest that the use of context is a useful approach to compensate for the simplicity of object detection and/or “visual illusions”. In fact, in many luminosity conditions, the objects can be partially hidden, or their colour can be saturated. Moreover, the use of complex visual recognition systems can impose a high processing load for an embedded system.
### Table 5

<table>
<thead>
<tr>
<th></th>
<th>Tables</th>
<th>Cola cans</th>
<th>Windows</th>
<th>Doors</th>
<th>G tab</th>
<th>G can</th>
<th>G door</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tables</td>
<td>93%</td>
<td>0.0%</td>
<td>3.6%</td>
<td>2.8%</td>
<td>7.9%</td>
<td>3.4%</td>
<td>29%</td>
</tr>
<tr>
<td>Cola cans</td>
<td>0.8%</td>
<td>82%</td>
<td>2.6%</td>
<td>0.0%</td>
<td>19%</td>
<td>5.4%</td>
<td>0.0%</td>
</tr>
<tr>
<td>Windows</td>
<td>0.0%</td>
<td>7.8%</td>
<td>78%</td>
<td>4.6%</td>
<td>6.7%</td>
<td>6.9%</td>
<td>2.4%</td>
</tr>
<tr>
<td>Doors</td>
<td>0.0%</td>
<td>0.0%</td>
<td>3.8%</td>
<td>94%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.5%</td>
</tr>
<tr>
<td>G tab</td>
<td>6.2%</td>
<td>0.0%</td>
<td>11%</td>
<td>0.0%</td>
<td>17%</td>
<td>8.8%</td>
<td>0.0%</td>
</tr>
<tr>
<td>G can</td>
<td>0.2%</td>
<td>10%</td>
<td>0.7%</td>
<td>0.0%</td>
<td>49%</td>
<td>75%</td>
<td>0.0%</td>
</tr>
<tr>
<td>G door</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>3.2%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>69%</td>
</tr>
</tbody>
</table>

Table 5. Results of the experiment with spatial context (without reference object) for the recognition of tables, cola cans, windows and doors. The columns correspond to the real objects, the lines to the recognized objects.

### 3.4 Recognition of composite objects

Most objects have a complex structure with several parts of different colours and complex shapes. The detection of some of these parts and their relationship with each other can provide important information on the object itself. The example considered here is a human (Fig. 10). In this case, skin colour recognition is able to detect the face, the arms and the hands, the rest of the body being hidden by clothes. Studying the relationships between these parts can help the robot to recognize these parts and decide whether a human being is present.

![Fig. 10. Examples of composite objects. The first one is a teacup having three main parts with different colours. The second one is a human where three parts are detected: the face and the arms.](image)

In this experiment, each detected feature is compared with each of the others. For all of these comparisons, the Bayesian network gives a result belonging to the set \{face, arm, hand\} for the feature. The final result is the one with the highest probability Ramel [2006].

The Bayesian network dedicated to this task is presented in Figure (11). The equation (22) gives the decomposition of the joint distribution. The nodes are described below.

\[ \text{Obj}_2: \] the part of the object that we wish to identify.
\[ \theta_1: \] the angle between the main axis of the reference part and a horizontal line.

\[ \theta_2: \] the angle between the great axis of the part that we wish to identify and a horizontal line.

\[ RPosX: \] the relative position on the \( x \)-axis of two parts.

\[ RPosY: \] the relative position on the \( y \)-axis of two parts.

\[ VPos: \] the absolute vertical position of the part to be identified (\( Obj_2 \)), in degrees.

\[ AS: \] the apparent size of \( Obj_2 \) in the image, in pixels. This is the average of the length and the width of the feature.

\[ h/w_1: \] the ratio between the height and the width, on the image, of the detected object \( Obj_i \), where \( i \in \{1, 2\} \).

\[ Col_i: \] the colour of \( Obj_i \), where \( i \in \{1, 2\} \), given by the average value of the \( (v) \) component of the HSV space.

---

**Identification**

As in the reference experiment, we use a database of examples to train the Bayesian network. From these examples, the network determines all the pa-
rameters of the distributions by using maximum likelihood parameter estimation.

Use

Finally, the use of these Bayesian networks determines the probability distribution.

\[
P(Obj_2 | Obj_1 \land h/w_1 \land Col_1 \land h/w_2 \land Col_2 \land RPosX \land RPosY \land Vpos \land theta_1 \land theta_2 \land AS \land D \land C)
\]  

Results

The results presented in Tables (6) and (7) correspond to the recognition of the face, the arms and the hands. In Table (7), the Bayesian network must differentiate the left and the right arms. The entire database comprises 273 examples. The training database consists of 30% of the total database, randomly chosen. Figure (12) gives several examples of scenes and the recognition of human body parts.

<table>
<thead>
<tr>
<th></th>
<th>Face</th>
<th>Hands</th>
<th>Arms</th>
<th>GC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Face</td>
<td>97.8%</td>
<td>2.3%</td>
<td>3.0%</td>
<td>7.1%</td>
</tr>
<tr>
<td>Hands</td>
<td>0.0%</td>
<td>90.2%</td>
<td>1.1%</td>
<td>7.0%</td>
</tr>
<tr>
<td>Arms</td>
<td>0.5%</td>
<td>0.0%</td>
<td>94.4%</td>
<td>17.0%</td>
</tr>
<tr>
<td>GC</td>
<td>1.7%</td>
<td>7.5%</td>
<td>1.5%</td>
<td>68.9%</td>
</tr>
</tbody>
</table>

Table 6. Results of recognizing face, arms and hands. The experiment uses a garbage model (GC).

<table>
<thead>
<tr>
<th></th>
<th>Face</th>
<th>Hands R</th>
<th>Hands L</th>
<th>Arms R</th>
<th>Arms L</th>
<th>GC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Face</td>
<td>97.9%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>3.5%</td>
<td>0.3%</td>
<td>7.0%</td>
</tr>
<tr>
<td>Hands R</td>
<td>0.0%</td>
<td>74.6%</td>
<td>0.6%</td>
<td>0.3%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>Hands L</td>
<td>0.0%</td>
<td>11.2%</td>
<td>91.7%</td>
<td>0.1%</td>
<td>1.0%</td>
<td>3.2%</td>
</tr>
<tr>
<td>Arms R</td>
<td>0.1%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>86.0%</td>
<td>21.0%</td>
<td>9.9%</td>
</tr>
<tr>
<td>Arms L</td>
<td>0.2%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>8.4%</td>
<td>69.2%</td>
<td>5.5%</td>
</tr>
<tr>
<td>GC</td>
<td>1.8%</td>
<td>14.2%</td>
<td>7.8%</td>
<td>1.7%</td>
<td>2.7%</td>
<td>74.5%</td>
</tr>
</tbody>
</table>

Table 7. Results of recognizing face, arms (right and left) and hands (right and left). The experiment uses a garbage model (GC).

We observe that the recognition of the parts is very good, but the differentiation of the left and the right is slightly worse. In fact, confusion between
left and right is the most frequent observation, but there is little confusion between the different parts of the body. This is because the database includes humans seen from both front and back, leading to confusion between left and right.
Fig. 12. The Bayesian network applies a red dot when it recognizes a face, a green dot when it recognizes a left arm, a blue dot when it recognizes a right arm, a yellow dot when it recognizes a left hand and a turquoise dot when it recognizes a right hand.
4 Conclusions and outlook

In this work, we first considered the incompleteness of the world and the impossibility of having perfect sensors. Thus, a system, artificial or alive, must cope with uncertainty. In this context we presented an approach that increases the ability to recognize objects by using context information, which does not depend on the structure of objects but on the common uses of the object, its topological relationship with other objects and its absolute position in space. To handle uncertainties, our approach uses the probabilistic method called a Bayesian network.

Our experiments demonstrate that the use of context information can strongly improve object recognition. The improvement of the distinction between two objects looking like each other, like the trash can and the cola can, is between 40% and 65%. Moreover, by using only the absolute position in space, the improvement of the recognition of objects like windows, doors or a table is better than 100% (the object is recognized in more than double the number of situations than when we do not use the context).

We have demonstrated that context is useful for analysing several parts of an object to recognize them (and by extension to recognize the object). In the case of a human being, the recognition of the face and limbs leads to gesture recognition. In the field of human–robot interaction, gesture recognition will be useful for obtaining new information and for learning by imitation or by demonstration.

4.1 Outlook

The usefulness of the context information has been demonstrated for several objects such as cola cans, windows, arms and faces. Nevertheless, this method can be extended to other features, for example, to recognize features such as eyes, mouth or nose to improve face detection, or elementary features such as corners and lines to recognize geometrical shapes. Thus, we can adapt this method to recognize more complex objects. Of course, we can use new variables, for example, not only the $v$ value of the $HSV$ space but also the $h$ and $s$ value and other variables depending on the shape. Moreover, we can use probability distributions other than the Gaussian.

To take more information into account, other modalities can be used: laser scanners to obtain 3D information, microphones for speech recognition, and mapping and localization for position and location information.

References


URL citeseer.nj.nec.com/kortenkamp96recognizing.html.


Bayesian Maps: probabilistic and hierarchical models for mobile robot navigation

Julien Diard\textsuperscript{1} and Pierre Bessière\textsuperscript{2}

\textsuperscript{1} Laboratoire de Psychologie et NeuroCognition CNRS UMR 5105, Université Pierre Mendès France, BSHM
\textsuperscript{2} CNRS - Grenoble Université

1 Introduction

Imagine yourself lying in your bed at night. Now try to answer these questions: Is your body parallel or not to the sofa that is two rooms away from your bedroom? What is the distance between your bed and the sofa? Except for some special cases (like rotating beds, people who actually sleep on their sofas, or tiny apartments), these questions are usually nontrivial, and answering them requires abstract thought. If pressed to answer quickly, so as to forbid the use of abstract geometry learned in high school, the reader will very probably give wrong answers.

However, if people had the same representations of their environment that roboticians usually provide to their robots, answering these questions would be very easy. The answers would come quickly, and they would certainly be correct. Indeed, robotic representations of space are usually based on large-scale, accurate, metric Cartesian maps. This enables judgment of parallelism and estimations of distances to be straightforward.

On the other hand, even though humans have difficulties with these questions, they usually have no trouble navigating from the sofa to the bed, or learning to do so in a new apartment. Robots have more difficulties in the same situation. In most robotic mapping approaches, the acquisition of a precise, and more importantly, accurate map of the environment is a prerequisite to solving navigation tasks. This is still a difficult and open issue in the general case.

Therefore, there appears to be a discrepancy in representations of space between the ones we usually provide to the robots we build and program, and the representations of space humans or animals use. Indeed, the nature, number, and possible interplay of the spatial representations involved in human or animal navigation processes are still an open question in the life sciences. There is also a discrepancy in the difficulty of navigation tasks currently solved by state-of-the-art robots and the navigation tasks solved very easily by humans or animals.
We believe studying the difference between robotics and life sciences models of navigation can be very fruitful, both for modelling better robots and understanding animal navigation better. That is the topic of this chapter.

We first offer a quick overview of navigation models, both in robotics and in biology. We will first focus, more precisely, on probabilistic approaches to navigation and mapping in robotics. These approaches include – but are far from limited to – Kalman filters [Leonard et al., 1992], Markov localization models [Thrun, 2000], (partially or fully) observable Markov decision processes [Kaelbling et al., 1998], and hidden Markov models [Rabiner and Juang, 1993]. We will here assume that the reader has some familiarity with these approaches. We will show how these methods differ from most models of human or animal navigation. Indeed, whereas robotics approaches mostly rely on large-scale monolithic representations of space, models of animal navigation, right from the start, assume hierarchies of representations. We thus then describe hierarchical approaches to robotic mapping.

Indeed, in this domain of probabilistic modelling for robotics, hierarchical solutions are currently flourishing. However, we will argue that the main philosophy used by all these approaches is to try to extract, from a very complex but intractable model, a hierarchy of smaller models. Of course, automatically selecting the relevant decomposition of a problem into subproblems is quite a challenge – this challenge being far from restricted to the domain of navigation for robots facing uncertainties.

We propose to pursue an alternative route. We investigate how, starting from a set of simple probabilistic models, one can combine them to build more complex models. The goal of this paper is therefore to present a new formalism for building models of the space in which a robot must navigate (the Bayesian Map model), and a method for combining such maps together in a hierarchical manner (the Abstraction operator). This formalism allows for a new representation of space, in which the final program is built upon many inter-related models, each of them deeply rooted in lower-level sensorimotor relationships.

For brevity, we will discuss neither the learning methods that can be included in Bayesian Maps [Simonin et al., 2005], nor other operators for merging Bayesian Maps (such as the Superposition operator [Diard et al., 2005]). The foundation of the present work was created in Diard’s PhD thesis [Diard, 2003].

The rest of this chapter is organized as follows. Section 2 gives a quick overview of the most prominent models of navigation and representation of large-scale space, first from a robotics point of view, then from a life sciences point of view. Section 3 offers a comparison of the main characteristics of the models, and an analysis of their strengths and weaknesses, and argues in favour of the need for hierarchical and modular probabilistic models of navigation. We then introduce our contribution to the domain, the Bayesian Map formalism (Section 4), and one of the operators we defined for combining Bayesian Maps, the abstraction operator (Section 5). Finally, we report in
Section 6 a series of robotic experiments in which we apply the Bayesian Map model and the abstraction operator on a Koala mobile robot, in a proof-of-concept experiment.

2 Navigation models in robotics and biology

We focus this brief review on existing models of navigation skills, in both robotics and life sciences. Because the literature in robotics concerning the representation of space is so large, we focus here on probabilistic approaches to mapping. In the life sciences, we describe some of the more prominent theoretical models of large-scale navigation in humans and animals, focusing on their hierarchical nature.

2.1 Probabilistic models of navigation and mapping

There is currently a wide variety of models in the domain of probabilistic mobile robot programming. These approaches include Kalman Filters (KF, [Leonard et al., 1992]), Markov Localization models (ML, [Thrun, 2000]), (Partially or fully) Observable Markov Decision Processes (POMDP, MDP, [Boutilier et al., 1999]), Hidden Markov Models (HMM, [Rabiner and Juang, 1993]), Bayesian Filters (BFs), and even Particle Filters (PFs). The literature covering these models is huge: for references that present several of them at once, giving unifying pictures, see [Murphy, 2002, Roweis and Ghahramani, 1999, Smyth et al., 1997]. Some of these papers define taxonomies of these approaches, by proposing some ordering that helps to classify them into families. One such taxonomy is presented in Fig. 1 (from [Diard et al., 2003]). It is based on a general-to-specific ordering: for example, it shows that Dynamic Bayesian Networks (DBNs) are a specialization of the Bayesian network formalism, tailored to take time series into account. In Fig. 1, subtrees that correspond to different specialization strategies can be identified. In the remainder of this section, we will focus on the Markov localization subtree, which corresponds to specializing DBNs using a four-variable model.

The ML model is basically an HMM with an additional action variable. It seems especially relevant in the robotic programming domain, because obviously robots can affect their states via motor commands. The stationary model of an HMM is basically the decomposition

$$ P(P^t, L^t, L^{t-1}) = P(P^{t-1})P(L^t | L^{t-1})P(P^t | L^t), $$

(1)

where $P^t$ is a perception variable, and $L^t$ and $L^{t-1}$ are state variables or, more precisely in our navigation case, location variables at time $t$ and $t - 1$. $P(L^t | L^{t-1})$ is commonly called the transition model, and $P(P^t | L^t)$ is referred to as the observation model. Starting from this structure, the action variable $A^t$ is used to refine the transition model $P(L^t | L^{t-1})$ into
Fig. 1. Some common probabilistic modelling formalisms and their general-to-specific partial ordering (adapted from [Diard et al., 2003]). The ML subtree, which results from specializing DBNs, is highlighted (dashed nodes).

\[ P(L^t \mid A^t L^{t-1}) \], which is called the action model. Thus, the ML model is sometimes referred to as the input–output HMM model. Because of the generality of the BRP formalism, the model for Markov Localization can be cast into a BRP program. This is shown Fig. 2.

![Program Description Table]

<table>
<thead>
<tr>
<th>Program</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Relevant Variables:</td>
</tr>
<tr>
<td></td>
<td>( P^t ): perception variable</td>
</tr>
<tr>
<td></td>
<td>( L^t ): discrete location variable at time ( t )</td>
</tr>
<tr>
<td></td>
<td>( L^{t-1} ): discrete location variable at time ( t-1 )</td>
</tr>
<tr>
<td></td>
<td>( A^t ): action variable</td>
</tr>
<tr>
<td></td>
<td>Decomposition:</td>
</tr>
<tr>
<td></td>
<td>( P(A^t L^t L^{t-1} L^t) = P(L^t \mid A^t L^{t-1})P(P^t \mid L^t)P(A^t)P(L^{t-1}) )</td>
</tr>
<tr>
<td></td>
<td>Parametric Forms:</td>
</tr>
<tr>
<td></td>
<td>usually, matrices or particles</td>
</tr>
<tr>
<td></td>
<td>Identification:</td>
</tr>
<tr>
<td></td>
<td>any</td>
</tr>
<tr>
<td></td>
<td>Question:</td>
</tr>
<tr>
<td></td>
<td>localization ( P(L^t \mid A^t P^t) )</td>
</tr>
</tbody>
</table>

The ML model is mostly used in the literature to answer the question \( P(L^t \mid A^t P^t) \), which estimates the state of the robot, given the latest motor commands and sensor readings. When this state represents the position of the robot in its environment, this amounts to localization. When this stationary ML model is replicated over time to estimate the positions \( L^{0:T} \) of the robot over a long series of time steps, it can be shown that an iterative localization
procedure exists that localizes the robot simply by updating the last position estimate in view of the latest motor commands $A^t$ and sensor readings $P^t$. This justifies, in this presentation, the focus on the stationary model.

The ML model has been successfully applied in a range of robotic applications, the most notable examples being the Rhino ([Thrun et al., 1998, Burgard et al., 1999]) and Minerva ([Thrun et al., 1999a,b]) robotic guides. The most common application of the ML model is the estimation of the position of a robot in an indoor environment, using a fine-grained metric grid as a representation. In other words, in the model of Fig. 2, the state variable is very frequently the pose of the robot, i.e. a pair of $x, y$ discrete Cartesian coordinates for the position, and an angle $\theta$ for the orientation of the robot. Assuming a grid cell size of 50 cm, an environment of 50 m $\times$ 50 m, and a 5$^\circ$ angle resolution entails a state space of 720,000 states.

Using some specialized techniques and assumptions, it is possible to make this memory-consuming model tractable.

For example, the forms of the probabilistic model can be implemented using sets of particles. These approximate the probability distributions involved in Fig. 2, which leads to an efficient position estimation. This specialization is called the Monte Carlo Markov Localization model (MCML, [Fox et al., 1999]).

Another possibility is to use a Kalman filter as a specialization of the ML model, in which variables are continuous. The action model $P(L^t \mid A^t \ L^{t-1})$ and the observation model $P(P^t \mid L^t)$ are both specified using Gaussian laws with means that are linear functions of the conditioning variables. With these hypotheses, it is possible to solve the inference problem analytically to answer the localization question. This leads to an extremely efficient algorithm that explains the popularity of Kalman filters.

### 2.2 Biologically inspired models

All the approaches mentioned in the preceding section are based on the classical view of robotic navigation, which is inherited from marine navigation. In this view, solving a navigation task basically amounts to answering sequentially the questions of Levitt and Lawton: “Where am I?”, “Where are other places with respect to me?”, and “How do I get to other places from here?” [Levitt and Lawton, 1990]. These are formulated somewhat similarly in the works of Leonard and Durrant-Whyte: “Where am I?”, “Where am I going?”, and “How should I get there?” [Leonard and Durrant-Whyte, 1991].

While they are a valid first decomposition of the navigation task into subtasks, these questions have usually led to models that require a global model of the environment, which allows the robot to localize itself (the first question), to infer spatial relationships between the current recognized location and other locations (the second questions), and to plan a sequence of actions to move within the environment (the third questions). These skills amount
to the first two phases of the “perceive, plan, act” classical model of robotic control.

Very early in their analysis, biomimetic models of navigation disputed this classical view of robotic navigation. Indeed, when studying living beings, the existence of such a unique and global representation that would be used to solve these three questions is very problematic. This seems obvious even for simple animals like bees and ants. For instance, the outdoor navigation capabilities of the desert ant *Cataglyphis*, which have been widely studied, rely on the use of the polarization patterns of the sky [Lambrinos et al., 2000]. It is clear that such a strategy is useless for navigating in a nest; this calls for another navigation strategy, and another internal model. The existence of a unique representation is also doubtful for humans. The navigation capabilities of humans are based on internal models of their environment (cognitive maps), but their nature, number and complexity are still largely debated (see for instance [Berthoz, 2000, Redish and Touretzky, 1997, Wang and Spelke, 2002], for entry points into the huge literature associated with this domain).

As a consequence, biomimetic approaches assume from the start the existence of multiple representations, most often articulated in a hierarchical manner. We now give a brief review of some theories from that domain, focusing on their hierarchical components.

**Works by Redish and Touretzky**

Works by Redish and Touretzky address the issue of the role of the hippocampus and parahippocampal populations in rodent navigation, focusing on the well-studied place cells and head direction cells. They proposed a conceptual model [Touretzky and Redish, 1996] and discussed its anatomical plausibility [Redish and Touretzky, 1997]. Their hierarchical conceptual model consists of four spatial representations (place code, local view, path integrator and head direction code), supplemented by two components called the reference frame selection subsystem and the goal subsystem.

Place codes are local representations tied to one or several landmarks or geometric features of the environment. When the environment of the animal becomes large or structured, several place codes may be used to describe this environment, each place code representing a part of the environment. For instance, Gothard et al. [Gothard et al., 1996] found different place codes for a rat navigating in an environment containing a goal and a starting box. They identified three independent place codes: one tied to the room, one to the goal, and one to the box. These effectively provide representations of sections of the environment: cells tuned to the box frame were only active when the rat was in or around the box, cells tuned to the goal only responded when the rat was near the goal, cells tuned to the room were active at other times (i.e. when the rat was not near the box or the goal).

The reference frame selection component is responsible for selecting the appropriate place code for navigating in the environment. In the above exam-
ple, this means that it is responsible for selecting, at any given time, which place code should be active.

This theory thus proposes an account of the low-level encoding of space in central nervous systems of animals using a two-layer hierarchy of models. The low-level layer consists of a series of place codes describing portions of the animal environment, under the hierarchical supervision of a larger-space model.

Computational models of the low-level component of this hierarchy, (i.e. place cells and head-direction cells) abound in the literature (e.g. [Hartley and Burgess, 2002]), whereas the reference frame selection component, to the best of our knowledge, has yet to be mathematically defined.

**Works by Jacobs and Schenk**

Jacobs and Schenk proposed a new theory of how the hippocampus encodes space [Jacobs and Schenk, 2003; Jacobs, 2003]. This theory is called the Parallel Map Theory (PMT), and it defines a hierarchy of navigation representations made of three components and two layers.

The bearing map is the first, low-level, component. This is a single map based on several directional cues such as intersecting gradients. It provides a large-scale two-dimensional reference frame, enabling large-scale navigation skills, simply using gradient ascent or descent.

The sketch maps are the second component of the low-level layer of the hierarchy. They encode small-scale fine-grained representations of the relationship of landmarks that are close to each other (positional cues). This creates local representations, which can be used for precise control of the position, and thus for solving precise, small-scale navigation tasks.

Finally, the integrated map is the third, high-level, component. This is constructed from the bearing map and several sketch maps. It consists of a unified map of large-scale environments, where the local sketch maps are cast into the large-scale reference frame of the bearing map. This provides the means to infer large-scale spatial relationship between the local, metric representations of the sketch maps, thus allowing computation of large-scale shortcuts and detours.

To the best of our knowledge, the papers by Jacobs and Schenk do not provide computational models of these different components. Instead, they mainly focus on the anatomical and phylogenetic plausibility of their conceptual model. This provides many experimental predictions concerning possible impairments resulting from lesions.

**Works by Wang and Spelke**

These authors dispute the idea that enduring, allocentric, large-scale representations of an environment should be the main theoretical tool used for investigating navigation in humans and animals. Indeed, the cognitive map
concept, introduced by Tolman in 1948, is still controversial [Tolman, 1948].
Instead, Wang and Spelke argue that many navigation capabilities in animals can be explained by dynamic, egocentric representations that cover a limited portion of the environment [Wang and Spelke, 2000, 2002]. Such representations can be studied in animals that are far simpler than humans, such as desert ants [Lambrinos et al., 2000].

Studies on these animals have identified three subsystems: a path integration system, a landmark-based navigation system, and a reorientation system. This last component is not hierarchically related to the other two, as it is mainly responsible for resetting the path integration system when the animal becomes disoriented. However, the first two components show a strong hierarchical relation. Indeed, it has been shown that the landmark-based strategy is hierarchically higher in the cognitive mechanisms of insects and rodents. It also appears that, in the sudden absence of landmarks after learning a path, animals rely on the path integration encoding as a “backup” [Stackman et al., 2002, Stackman and Herbert, 2002].

This model is somewhat different from the previous studies, as it focuses on defining a hierarchy of skills of navigation, instead of hierarchies of representations of space, as in the PMT or studies of the hippocampal and parahippocampal areas.

**Works by Kuipers, Franz, and Trullier**

The hierarchies of models proposed in the biomimetic robotic literature ([Kuipers, 1996, Trullier et al., 1997, Franz and Mallot, 2000, Kuipers, 2000, Victorino and Rives, 2004]) have several aspects: they are hierarchies of increasing navigation skills, but also of increasing scale of the represented environment, of increasing time scale of the associated movements, and of increasing complexity of representations. This last aspect means that topologic representations, which are simple, come at a lower level than global metric representations, which are arguably more complex to build and manipulate. This ordering stems from the general observation that animals that are able to use shortcuts and detours between two arbitrary encoded places (skills that require global metric models) are rather complex animals, like mammalians. These skills seem to be mostly absent from simpler animals, like invertebrates.

The resulting proposed hierarchies show a striking resemblance. We present the salient and common features of these hierarchies by summarizing the Spatial Semantic Hierarchy (SSH) proposed by Kuipers [Kuipers, 1996, 2000]. It is, to the best of our knowledge, the only biomimetic approach that has been applied to obtain a complete and integrated robotic control model.

The SSH essentially consists of four hierarchical levels: the control level, the causal level, the topological level, and the metric level.

The control level is a set of reactive behaviours, which are control laws deduced from differential equations. These behaviours describe how to move the robot to reach an extremum of some gradient measure. This extremum
Bayesian Maps 169

can be zero-dimensional (a point in the environment), in which case it is
called a locally distinctive state. The associated behaviour is called a hill-
climbing law. The extremum can also be one-dimensional (a line or curve in
the environment), in which case the behaviour is called a trajectory-following
law. Provided that any trajectory-following law guarantees arriving at a place
where a hill-climbing law can be applied, then alternating laws of both types
displace the robot in a repetitive fashion. This solves the problem of the
accumulation of odometry errors. The control level is also referred to as the
guidance level [Trullier et al., 1997, Franz and Mallot, 2000].

Given the control level, the environment can be structured and summa-
rized by the locations of locally distinctive states and the trajectories used
to go from one such state to another. This abstraction takes place in the
causal level, which is the second level of the hierarchy of representations.
Unlike the control level, it allows the robot to memorize relationships be-
tween places that are outside the current perceptive horizon (this is part of
the way-finding capabilities in other terminologies [Trullier et al., 1997, Franz
and Mallot, 2000]). To do so, Kuipers abstracts locally distinctive places as
views \( V \), the application of lower-level behaviours as actions \( A \), and defines
schemas as tuples \( \langle V, A, V' \rangle \) (expressed as first-order logic predicates). The
schemas have two meanings. The first is a procedural meaning: “when the
robot is at \( V \), it must apply action \( A \).” This aspect of a schema is equivalent
to the recognition-triggered response level of the other hierarchies [Trullier
et al., 1997, Franz and Mallot, 2000], or to the potential field approaches, or
to other goal-oriented methods. However, the second meaning of schemas is
a declarative one, where \( \langle V, A, V' \rangle \) stands for: “applying action \( A \) from view
\( V \) eventually brings the robot to view \( V' \).” This allows using the schemas for
prediction of future events, or in a planning process, for example.

The goal of the topological level is to create a globally consistent represen-
tation of the environment, as structured by places, paths and regions. These
are extracted from lower-level schemas by an abduction process that creates
the minimum number of places, paths and regions to maintain consistency
with the known schemas. Places are zero-dimensional parts of the environ-
ment that can be abstractions of lower-level views, or abstractions of regions
(for higher-level topological models). Paths are one-dimensional, oriented, and
can be built upon one or more schemas. Finally, regions are two-dimensional
subspaces, delimited by paths. It must be noted that, because the problem of
accumulation of odometry error was dealt with at the control level, building
a globally consistent topological representation (i.e. solving the global con-
ectivity problem) is much easier. To do so, Kuipers proposes an exploration
strategy, the rehearsal procedure, which, unfortunately, requires a bound on
the exploration time and is not well suited to dynamic environments. The
places and paths of the topological representation obtained can be used for
solving planning queries using classical graph-searching algorithms.

The last level is the metric level, in which the topological graph is cast
into a unique global reference frame. For reasons outlined above (and detailed
in [Kuipers, 2000]), this level is considered as a possibility, not a prerequisite for solving complex navigation tasks. If the sensors are not good enough to maintain a good estimation of the Cartesian coordinates of the position, for instance, it is still possible to use the topological model to act in the environment — although shortcuts and detours are not possible. Indeed, few robotics systems implementing biomimetic models include the metric level [Franz and Mallot, 2000, Trullier et al., 1997].

3 Theoretical comparison

3.1 Which mathematical formalism?

A major drawback of the biomimetic models presented previously is that they are seldom defined as computational models: they give conceptual frameworks for understanding animal navigation but lack complete mathematical definitions that would make simulations of these models possible. The notable exception is the SSH model, which not only defines layers in a hierarchy of space representations but also defines each of them mathematically.

However, the SSH model uses a variety of formalisms for expressing knowledge at different layers of the hierarchy: differential equations and their solutions for the control level, and first-order logic and deterministic algorithms for higher-level layers of the hierarchy. This makes it difficult to justify the consistency and correctness of the mechanisms for communication between the layers of the hierarchy theoretically. In some cases, it even limits and constrains the contents of the layers: for instance, the SSH model requires that the behaviours of the control level guarantee that the robot reaches the neighbourhood of a given locally distinctive state. In our view, this constraint is barely acceptable for any kind of realistic robotic scenario. Consider dynamic environments: how can we guarantee that a robot will reach a room if a door on the route can be closed?

We assume as a starting point for our analysis that the best formalism for expressing incomplete knowledge and dealing with uncertain information is the probabilistic formalism [Bessière et al., 1998, Jaynes, 2003]. This gives us a clear and rigorous mathematical foundation for our models. The probability distributions are our unique tool for the expression and manipulation of knowledge, and in particular, for communication between submodels. We will thus argue in favour of hierarchical probabilistic models.

3.2 Hierarchical probabilistic models

This idea is not a breakthrough: in the domain of probabilistic modelling for robotics, hierarchical solutions are currently flourishing. The more active domain in this regard is decision-theoretic planning: one can find variants of
MDPs that accommodate hierarchies or that select automatically the partition of the state space (see, for instance, [Hauskrecht et al., 1998, Lane and Kaelbling, 2001], or browse through the references in [Pineau and Thrun, 2002]). More exceptionally, one can find hierarchical POMDPs [Pineau and Thrun, 2002]. The current work can also be related to Thrun’s object mapping paradigm [Thrun, 2002], in particular concerning the aim of transferring some of the knowledge the programmer has about the task to the robot.

Some hierarchical approaches outside the MDP community include hierarchical HMMs and their variants (see [Murphy, 2002] and references therein), which, unfortunately, rely on the notion of the final state of the automaton. Another class of approaches relies on the extraction of a graph from a probabilistic model such as a Markov localization model [Thrun, 1998], or an MDP [Lane and Kaelbling, 2002]. Using such deterministic notions is inconvenient in a purely probabilistic approach, such as we are pursuing here.

Moreover, the main philosophy used by all the previous approaches is to try to extract, from a very complex but intractable model, a hierarchy of smaller models (structural decomposition, see [Pineau and Thrun, 2002]).

Again, this comes from the classical robotic approach, where the process of perception (in particular, localization) is assumed to be independent of the processes of planning and action. A model such as the ML model (Fig. 2) is only concerned with localization, not control; therefore, its action variable $A^t$ is actually only used as an input to the model. In this view, a pivotal representation is used between the perception and planning subproblems. It is classically assumed that the more precise this pivotal model, the better. Unfortunately, when creating integrated robotic applications, dealing with both the building of maps and their use is necessary. Some authors have realized that their global metric maps were too complex to be easily manipulated. Therefore, they have tried to degrade their maps, which were so difficult to obtain initially, for instance, by extracting graphs from their probabilistic models [Thrun, 1998]. This problem is also at the core of the robotic planning domain, where the given description of the environment is assumed to be an infinitely precise geometrical model. The difficulty is to discretize this intractable, continuous model into a finite model, typically, in the form of a graph [Latombe, 1991, Kavraki et al., 1996, Mazer et al., 1998, Svestka and Overmars, 1998].

3.3 Modular probabilistic models: towards Bayesian Maps

We propose to pursue an alternative route, investigating how, starting from a set of simple models, one can combine them to build more complex models. Such an incremental development approach allows us to depart from the classical “perceive, plan, act” loop, considering instead hierarchies built upon many inter-related models, each of them deeply rooted in lower-level sensory and motor relationships.
The Bayesian robotic programming methodology offers exactly the formal tool that can transfer information from one program to another in a theoretically rigorous fashion. Indeed, in Bayesian robotic programs, terms appearing in a description $c^1$ can be defined as a probabilistic question to another description $c^2$. This creates a link between the two descriptions, one being used as a resource by another. Depending on the way questions are used to link subprograms, several different operators can be created, each with specific semantics: for instance, in the framework of behaviour-based robotics, Lebeltel has defined behaviour combination, hierarchical behaviour composition, and behaviour sequencing and sensor model fusion operators. He has also applied these successfully to realize a complex watchman robot behaviour using a control architecture involving four hierarchical levels [Lebeltel et al., 2004].

Thus, we can solve a global robotic task problem by first decomposing it into subproblems, then writing a Bayesian robot program for each subproblem, and finally combining these subprograms. This method makes robot programming similar to structured computer programming. So far in our work, we have let the programmer do this analysis: relevant intermediary representations can be imagined, or copied from living beings. We propose to apply this strategy to the map-based navigation of mobile robots. The submodels can be submaps, in the spatial sense (i.e. covering a part of the global environment), or in the subtask sense (i.e. modelling knowledge necessary for solving part of the global navigation task), or even in less familiar senses (e.g. modelling partial knowledge from part of the sensorimotor apparatus).

Our approach is therefore based on a formalism for building models of the space in which the robot must navigate, called the Bayesian Map model, that allows us to build submodels that provide behaviours as resources. We also define operators for combining such maps in a hierarchical manner.

4 The Bayesian Map formalism: definition

4.1 Probabilistic definition

A Bayesian Map $c$ is a description that defines a joint distribution

$$P(P \ L^t \ L^{t'} \ A \ | \ c),$$

where:

- $P$ is a perception variable (the robot reads its values from physical sensors or lower-level variables);
- $L^t$ is a location variable at time $t$;
- $L^{t'}$ is a variable having the same domain as $L^t$, but at time $t'$ (without loss of generality, let us assume $t' > t$); and
- $A$ is an action variable (the robot writes commands to this variable).
For simplicity, we will assume here that all these variables have finite domains.

The choice of decomposition is not constrained: any probabilistic dependency structure can therefore be chosen here; see [Attias, 2003] for an example of how this can lead to interesting new models. Finally, the definition of forms and the learning mechanism (if any) are also not constrained.

For a Bayesian Map to be usable in practice, the description must be rich enough to generate behaviours. We describe as elementary behaviour any question of the form $P(A_{i} \mid X)$, where $A_{i}$ is a subset of $A$, and $X$ is a subset of the other variables of the map (i.e. of those not in $A_{i}$). A typical example consists of the probabilistic question $P(A \mid [P = p] [L' = l])$: compute the probability distribution over actions, given the current sensor readings $p$ and the goal $l$ to reach in the internal space of possible locations.

A behaviour can be not elementary, for example, if it is a sequence of elementary behaviours, or, in more general terms, if it is based on elementary behaviours and some other knowledge (which may be expressed in terms other than maps).

For a Bayesian Map to be interesting, we will also require that it generates several behaviours – otherwise, defining just a single behaviour instead of a map is enough. Such a map is therefore a resource, based on a location variable relevant enough to solve a class of tasks; this internal model of the world can be reified.

A “guide” one can use to “make sure” that a given map will generate useful behaviours is to check whether the map answers in a relevant manner the three questions $P(L^t \mid P)$ (localization), $P(L' \mid A L')$ (prediction) and $P(A \mid L' L')$ (control). By “relevant manner”, we mean that these distributions must be informative, in the sense that their entropy is “far enough” from its maximum (i.e. the distribution is different from a uniform distribution). This constraint is not formally well defined, but it seems intuitive to focus on these three questions. Indeed, the skills of localization, prediction and control are well identified in the literature as ways of generating behaviours. Checking that the answers to these questions are informative is a first step to evaluating the quality of a Bayesian Map with respect to solving a given task.

Figure 3 is a summary of the definition of the Bayesian Map formalism.

### 4.2 Generality of the Bayesian Map formalism

We now invite the reader to verify that the Markov localization model is indeed a special case of the Bayesian Map model by comparing Fig. 2 and Fig. 3. Recall that Kalman filters and particle filters are special cases of Markov localization, as they add hypotheses over the choice of dependency structure made by the Markov localization model. This implies that Kalman filters and particle filters are also special cases of Bayesian Maps.

Bayesian Maps can therefore accommodate many different forms, depending on the need or information at hand; for example, a Bayesian Map can be...
structured like a real-valued Kalman filter for tracking the angle and distance to some feature when it is available. If that feature is not present, or if the linearity hypotheses fail, we can use another Bayesian Map, which may not be a Kalman filter (for example, based on a symbolic variable).

Hierarchies of Bayesian Maps (via the abstraction operator) can thus be hierarchies of Markov localization models or hierarchies of Kalman filters, and so on. Moreover, heterogeneous hierarchies of these models can be imagined: ML over KFs, or even $n$ KFs and one ML model. In our view, this would be a potential alternative to the solution of Tomatis et al. [Tomatis et al., 2001, 2003].

### 5 Combining Bayesian Maps: definition of the abstraction operator and example

Having defined the Bayesian Map concept, we now turn to defining operators for combining Bayesian Maps. The one we present here is called the abstraction of maps. It is defined in Fig. 4 and discussed in the rest of this section.

As stressed above, in a Bayesian Map, the semantics of the location variable can be very diverse. The main idea behind the abstraction operator is to build a Bayesian Map $c$ containing locations that are other Bayesian Maps $c_1, c_2, \ldots, c_n$. The location variable of the abstract map will therefore take $n$ possible symbolic values, one for each underlying map $c_i$. Each of these maps will be “nested” in the higher-level abstract map, which justifies the use of the term “hierarchy” in our work. Recall that Bayesian Maps are designed for generating behaviours.

<table>
<thead>
<tr>
<th>Relevance</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relevant Variables:</td>
<td></td>
</tr>
<tr>
<td>$P$: perception variable</td>
<td></td>
</tr>
<tr>
<td>$L^t$: location variable at time $t$</td>
<td></td>
</tr>
<tr>
<td>$L^{t'}$: location variable at time $t'$, $t' &gt; t$</td>
<td></td>
</tr>
<tr>
<td>$A$: action variable</td>
<td></td>
</tr>
<tr>
<td>Decomposition:</td>
<td></td>
</tr>
<tr>
<td>Any</td>
<td></td>
</tr>
<tr>
<td>Parametric Forms:</td>
<td></td>
</tr>
<tr>
<td>Any</td>
<td></td>
</tr>
<tr>
<td>Identification:</td>
<td></td>
</tr>
<tr>
<td>Any</td>
<td></td>
</tr>
<tr>
<td>Question:</td>
<td></td>
</tr>
<tr>
<td>elementary behaviours: $P(A_i \mid X)$, with $A_i \subseteq A$</td>
<td></td>
</tr>
<tr>
<td>$X \subseteq \left( {P, L^t, L^{t'}, A} \setminus A_i \right)$</td>
<td></td>
</tr>
</tbody>
</table>

**Fig. 3.** The Bayesian Map model definition expressed in the BRP formalism.
Let us denote \(a_1, a_2, \ldots, a_k\), the \(k\) behaviours defined in the \(n\) underlying maps, with \(k \geq n\). In the abstract map, these behaviours can be used for linking the locations \(c_i\). The action variable of the abstract map will therefore take \(k\) possible symbolic values, one for each behaviour of the underlying maps. To build an abstract map having \(n\) locations, the programmer will require \(n\) previously defined lower-level maps, which generate \(k\) behaviours. The numbers \(n\) and \(k\) are therefore small, and so the abstract map deals with a small internal space, having retained from each underlying map only a symbol, and having “forgotten” all their details. This justifies the use of the name “abstraction” for this operator. However, this “summary mechanism” has yet to be described: that is what the perception variable \(P\) of the abstract map will be used for, as it will be the list of all the variables appearing in the underlying maps: \(P = P_1, L'_1, L''_1, A_1, \ldots, P_n, L'_n, L''_n, A_n\).

Given the four variables of the abstract map, we define its joint distribution with the following decomposition:

\[
P(P \mid L^t, L''^t) = P(L^t)P(L''^t)P(A \mid L^t, L''^t) \prod_{i=1}^{n} P(P_i, L'_i, L''_i \mid A_i) \mid L^t) \tag{2}
\]

**Fig. 4.** The abstraction operator definition expressed as a Bayesian Map.
In this decomposition, $P(L^t_i)$ and $P(L^t_i')$ are defined as uniform distributions. All the terms of the form $P(L_t L_t' A_i | [L_t = c])$ are defined as follows: when $c \neq c_i$, the probabilistic dependency between the variables $P_t$, $L_t$, $L_t'$, $A_i$ of the map $c_i$ is supposed unknown, and therefore defined by a uniform distribution. When $c = c_i$, however, this dependency is exactly what the map $c_i$ defines. Therefore this term is a question to the description $c_i$ but a question that includes the whole subdescription by asking for the joint distribution it defines. Because the last term, $P(A | L^t L^t')$, only includes symbolic variables that have a small number of values, it makes sense to define it as a table, which can be easily programmed a priori or learned experimentally.

The abstract Bayesian Map is now fully defined, and, given the underlying maps, can be built automatically. The last step is to verify that it generates useful behaviours. We will examine the guiding questions of localization, prediction and control.

The localization question leads to the following inference (derivation omitted): $P(L_t | P) \propto \prod_{i=1}^n P(L_t^i L_t'^i A_i | L^t_i)$. The interpretation of this result will be explained with an example in Section 6. The derivations for solving the prediction $P(L_t' | A L^t)$ and control $P(A | L^t L^t')$ questions are also straightforward, given Fig. 4.

Recall that the final goal of any Bayesian Map is to provide behaviours. In the abstract map, this is done by answering a question like $P(A | [L^t' = c_i] [P = p])$: what is the probability distribution over lower-level behaviours, knowing all values $p$ of the variables of the lower level, and knowing that we want to “go to map $c_i$” (more formally, “reach some location recognized as the lower-level map $c_i$”)? Answering this question thus allows selection of the most relevant underlying behaviour to reach a given high-level goal. The computation is as follows:

$$P(A | L_t' P) \propto \sum_{L^t_i} \left( \prod_{i=1}^n P(L_t^i L_t'^i A_i | L^t_i) \right) P(A | L^t L^t').$$

This computation includes the localization question, by weighing the probabilities given by the control model $P(A | L^t L^t')$. In other words, the distribution over the action variable $A$ includes all localization uncertainties. Each underlying model is used, even when the robot is located at a physical location that this model is not made for. As a direct consequence, there is no need to decide what map the robot is in, nor to switch from map to map: the computation considers all possibilities and weighs them according to their (localization) probabilities. Therefore the underlying maps are not necessarily “mutually exclusive” in a geographical sense.

### 6 Experimental validation

We report here an experiment made on the well-known Koala mobile robot platform (K-team company). To keep as much control as possible over our
experiments and the different effects we observe, we simplify the sensorimotor system and its environment. We only use the 16 proximeters \( P_x = P_{x0} \land \ldots \land P_{x15} \) of our robot, and we keep two degrees of freedom of motor control, via the rotation and translation speeds \( V_{rot} \) and \( V_{trans} \). The environment we use is a \( 5 \times 5 \) m area made of movable planks (a typical configuration is shown in Fig. 5). The goal of this experiment is to solve a navigation task: we want the robot to go and hide in any corner, as if the empty space in the middle of the area were dangerous.

The first programming step is to analyse this task into subtasks. Three particular situations are relevant for solving the task: the robot can either be near a wall, and it should follow it in order to reach the nearest corner, or the robot can be in a corner, and it should stop, or finally it could be in empty space, and should therefore go straight, so as to leave the exposed area as quickly as possible.

### 6.1 Low-level Bayesian Maps

Given this analysis, the second programming step is to define one Bayesian Map for each of the three situations. They all use the same perception variable \( P = P_x \) and the same action variable \( A = V_{rot} \land V_{trans} \).

The first map, \( c_{wall} \), describes how to navigate in the presence of a single wall, using a location variable \( L^t = \theta \land Dist \); the phenomenon “wall” is summarized by an angle \( \theta \) and a distance \( Dist \). Therefore, \( c_{wall} \) defines

\[
P(x,\theta,Dist|\theta',Dist',V_{rot},V_{trans},c_{wall}) = \prod_{i=1}^{12} \frac{P(x_i|\theta,Dist)}{P(\theta',Dist') \cdot P(V_{rot}|\theta,Dist,\theta',Dist') \cdot P(V_{trans}|\theta,Dist,\theta',Dist')}
\]

\( P(\theta,Dist) \) and \( P(\theta',Dist') \) are uniform probability distributions. Each term of the form \( P(x_i|\theta,Dist) \) is a set of bell-shaped probability distributions that were identified experimentally in a supervised learning phase: we physically put the robot in all 36 possible situations with respect to the wall and recorded proximeter values so as to compute experimental means and standard deviations. Finally, the two control terms \( P(V_{rot}|\theta,Dist,\theta',Dist') \) and \( P(V_{trans}|\theta,Dist,\theta',Dist') \) were programmed “by hand”: given the current angle and distance, and the angle and distance to be reached, what should the motor commands be?  

---

*Bell-shape probability distributions approximate Gaussian probability distributions on finite discretized ranges.*
This map successfully solves navigation tasks like “follow wall right”, “follow wall left”, “go away from wall”, “stop”, using behaviours of the same name. For example, “follow wall right” is defined by the probabilistic question $P(Vrot \ Vtrans \ | \ Px \ [L' = (90, 1)])$: compute the probability distribution on motor variables knowing the sensory input and knowing that the location to reach is $\theta = 90^\circ$, $Dist = 1$ (wall on the right at medium distance).

This map is an instance where a Kalman filter-based Bayesian Map could have been used instead, for example, if we had required a more accurate angle and distance to the wall using continuous variables. The coarse-grained set of values we used were sufficient for our experiments.

The two other Bayesian Maps we define are the following. 1) $c_{\text{corner}}$ describes how to navigate in a corner, using a symbolic location variable that can take four values: FrontLeft, FrontRight, RearLeft and RearRight. This is enough for solving tasks like “quit corner and follow right”, “away from both walls”, “stop”. 2) $c_{\text{emptyspace}}$ describes how to navigate in empty space, i.e. when the sensors do not see anything. The behaviours defined here are “straight ahead” and “stop”. For brevity, these two Bayesian Maps are not described further here; the interested reader can find details in Diard’s PhD thesis [Diard, 2003].

6.2 Abstract Bayesian Map

Given these three maps, the third and final programming step is to apply the abstraction operator to them. We obtain a map $c$ with a location variable $L = \{c_{\text{wall}}, c_{\text{corner}}, c_{\text{emptyspace}}\}$. The action variable lists the behaviours defined by the low-level maps: $A = \{\text{follow wall right, go away from wall,} \ldots\}$. The rest of the abstract map is in accordance with the schema of Fig. 4.

We now discuss the localization question. Let us assume that the robot is in empty space: all its sensors read zero. Let us also assume that the robot is currently applying the “straight ahead” behaviour, which sets $Vrot$ and $Vtrans$ near 0 (no rotation) and 40 (fast forward movement), respectively, using sharp bell-shaped distributions.

Let us consider the probability of being in location $c_{\text{emptyspace}}$ (with $w$ standing for wall, $c$ for corner and $e$ for emptyspace).

$$ P([L = c_{\text{emptyspace}}] \ | \ P)$$

$$ \propto \left( P(P_w \ L_w \ L_w' \ A_w \ | \ [L = c_{\text{emptyspace}}]) \right) \cdot \left( P(P_c \ L_c \ L_c' \ A_c \ | \ [L = c_{\text{emptyspace}}]) \right) \cdot \left( P(P_e \ L_e \ L_e' \ A_e \ | \ [L = c_{\text{emptyspace}}]) \right) .$$

Of the three terms of the product, two have uniform distributions, and one is the joint distribution given by $c_{\text{emptyspace}}$. This joint distribution gives a very high probability for the current situation, because describing the phenomenon “going straight ahead in empty space” basically amounts to favouring sensory
readings of 0 and motor commands near 0 and 40 for $V_{rot}$ and $V_{trans}$, respectively. The situation is quite the opposite for $P([L^t = c_{wall}] \mid P)$: for example, $c_{wall}$ does not favour this sensory situation at all. Indeed, the phenomenon “I am near a wall” is closely related to the sensors actually sensing something. The probability of seeing nothing on the sensors knowing that the robot is near a wall is very low: $P([L^t = c_{wall}] \mid P)$ will be very low. The reasoning is similar for $P([L^t = c_{corner}] \mid P)$.

This computation can thus be interpreted as the recognition of the most pertinent underlying map for a given sensorimotor situation. Alternatively, it can be seen as a measure of the coherence of the values of the variables of each underlying map, or even as a Bayesian comparison of the relevance of models, as assessed by the numerical value of the joint distributions of each lower-level model. Because these distributions include (lower-level) location and action variables, the maps are recognized not only from sensory patterns but also from what the robot is currently doing.

The localization question can therefore be used to assess the “validity zones” of the underlying maps, i.e. the places in the environment where the hypotheses of each model hold. Experimentally, we make the robot navigate in the environment, and we ask at each time step the localization question. We can visually summarize the answer, for example, by drawing values for $L^t$, and reporting the drawn value on a Cartesian map of the environment. A simplified but readable result is shown in Fig. 5. As can be seen, the robot correctly recognizes each situation for which it has a model. Note that the resulting zones are not contiguous in the environment: for example, all the corners of the environment are associated with the same symbol, namely, $c_{corner}$. This effect is known as perceptual aliasing. However, this very simple representation is sufficient for solving the task that was given to the robot: the behaviour “go hide in any corner” is indeed generated by the abstract map.

Using the abstract Bayesian Map we have programmed in this way, the robot can solve the task of reaching corners. A typical trajectory for the robot, starting from the middle of the arena, is to start by going straight ahead. As soon as a couple of forward sensors sense something, the “empty space” situation is no longer relevant, and the robot applies the best model it has, depending on the correlation between what the sensors see: if it looks like a wall and continues to do so as the robot moves, then the probability for the “wall” model is high; on the other hand, if it instead feels like a corner, then the corner model wins the probabilistic competition. Suppose the robot is near a wall and starts to follow it until a corner is reached. In our first version, the corner model was designed too independently of the wall model: the validity zone of the $c_{corner}$ map was too small and seldom visited by the robot as it passed the corner using the “follow wall right” behaviour, defined by $c_{wall}$. The robot would then miss the first corner and stop at another one. This shows that the decomposition of the task gives independent subtasks only as a first approximation. We solved the problem by modifying the “corner” model, so that it would recognize a corner on a typical “follow wall right” trajectory.
Fig. 5. 2D projection of the estimated “validity zones” of the maps $c_{\text{wall}}$, $c_{\text{corner}}$ and $c_{\text{emptyspace}}$. The bottom part of the figure is a screenshot of the localization module of the abstract map: it shows the “comparison” and competition between the underlying models. The winner is marked by the central dot: in this case, the robot was near a wall.

7 Conclusion

We have presented the Bayesian Map formalism: it is a generalization of most probabilistic models of space found in the literature. Indeed, it drops the usual constraints on the choice of decomposition, forms, or implementation of the probability distributions. We have also presented the abstraction operator, for building hierarchies of Bayesian Maps.

The experiments we presented are of course to be regarded only as “proofs of concept”. Their simplicity also served didactic purposes. However, these experiments, in our view, are a successful preliminary step towards applying our formalism. Part of the current work is of course aimed at enriching these experiments, in particular with respect to the scaling up capacity of the formalism.

Moreover, because each map of the hierarchy is a full probabilistic model, it is potentially very rich. Possible computations based on these maps include questions like the prediction question $P(L' | L)$, which can form the basis of planning processes. Hierarchies of Bayesian Maps can therefore be considered as model-based approaches rather than as purely reactive approaches. Exploiting such knowledge by integrating a planning process in our Bayesian Map formalism is also part of the ongoing work.
Acknowledgements

This work has been supported by the BIBA European project (IST-2001-32115). The authors wish to thank Emmanuel Mazer, Alain Berthoz and Panagiota Panagiotaki for many discussions on this work, and Carla Koike for invaluable comments on drafts of this chapter.

References


Bayesian approach to action selection and attention focusing

Carla Cavalcante Koike\textsuperscript{1}, Pierre Bessière\textsuperscript{2}, and Emmanuel Mazer\textsuperscript{3}

\textsuperscript{1} Computer Science Department - Universidade de Brasília
\textsuperscript{2} CNRS - GRAVIR Laboratory
\textsuperscript{3} PROBAYES SAS

1 The ultimate question for autonomous sensory–motor systems

What similarities can be found between an animal and an autonomous mobile robot? Both can control their motor capabilities based on information acquired through dedicated channels. For an animal, motor capabilities are muscles and joints, and filtered information from the environment is acquired through sensors: eyes, nose, ears, skin, and several others. For a mobile robot, motor capabilities are mostly end effectors and mechanical motors, and information about the surroundings consists of data coming from sensors such as proximeters, laser range sensors and bumpers.

In this work, we assume that an autonomous sensory–motor system, like the two examples discussed above, is continually answering the ultimate question: $P(M|z_{0:t} \wedge m_{0:t-1} \pi)$. This question can be put into words as: \textit{What can I do next, knowing what I have seen and what I have done up to now?}

Mathematically, the expression gives the probability distribution over the values of the motor control variables $M$ at time instant $t$, knowing the values of the observed variables $Z$ from time instant 0 to time instant $t$, as well as the values of all motor controls exerted from time instant 0 to time instant $t-1$, under the previous knowledge about the robot, its task and the environment. All this previous knowledge is assembled and summarized by the variable $\pi$.

This article proposes a framework for programming autonomous robots in such a way that they are able to answer the ultimate question. Our solution also features Bayesian-based action selection and selective perception. The framework is founded on a succession of incremental hypotheses and assumptions, defined within the strict mathematical framework presented in the chapter \textit{Basic concepts of Bayesian programming}.

Sections 2 to 6 present individual solutions to the ultimate question, as a succession of incremental capabilities and cumulative hypotheses and simplifications. Each individual solution to the ultimate question is discussed and
compared with the proposals from the previous sections. Cognitive implications of the proposed solutions are also presented.

Evidence of the practicability of the proposed concepts for controlling a robot is given in Section 7. The method applied to develop this experiment is an additional contribution.

2 Internal state variables to reduce complexity

The simplest possible solution to the ultimate question

\[ P(M^t|z^{0:t}, m^{0:t-1}) \]

is a table providing the probability distribution over the motor variables \( M \) at time instant \( t \). This table is manually filled with probability values for all the different values of observed and motor variables, with one dimension for each variable. As the time horizon extends, the dimensions of the table increase very quickly: each additional time step adds as many dimensions as the total number of variables.

This solution is very simple and requires only one assumption (that variables, including time, are assigned discrete values), but it is not feasible. The memory requirements are unrealistic for any practical implementation, and an incredibly large number of parameters must be adjusted, even though their manual programming is rather difficult.

This section deals with the addition of state variables as a means of simplifying the utilization of sensor signals and the definition of motor commands, uncoupling observation and motor command variables.

2.1 Bayesian program

We present in this section a Bayesian program for solving the ultimate question at a given time instant \( j \): variables at past time instants and their relationships are not considered here. Our goal in applying such a simplification is to show the utility of state variables when solving the ultimate question. We postpone dealing with problems associated with time dependence to the next section.

In addition to the set of observation variables \( Z \) and the set of motor variables \( M \), this program includes state variables \( S \). State variables are chosen as the minimum set of relevant variables related to the robot, to the specified task and to the most significant features in the robot’s environment. All these variables are considered at time instant \( j \).

Additionally, \( Z^j \) (the set of observation variables at time instant \( j \)) is considered to be composed of \( N_z \) observation variables, so that:

\[ Z^j = \{1Z^j, 2Z^j, \ldots N_zZ^j\}, \forall j. \]
The first term in this joint distribution (the decomposition in Figure 1) shows a priori information about state variables.

The second line in the decomposition equation is the product of terms related to the observation variables. These terms are called sensor models, and they establish the relationships between sensor observations and the environment state. The product of terms indicates that sensor observations are considered independent of each other if the state variables are known.

The last term in the equation is the definition of how motor variables change with state values.

It can be noticed that state variables dissociate observation variables and motor commands. Observations are linked to states in the sensor model, and motor commands are functions of states in the motor model.

This program can be used to estimate state variables from sensor measures and to find the motor command probability distributions given the sensor measures: in other words, to solve a simplified version of the ultimate question.

Fig. 1. Bayesian Program illustrating utilization of state variables.

2.2 Analysis and discussion

State variables and sensor fusion are used regularly in mobile robotics for several reasons. State variables are able to uncouple sensor and motor systems, thus allowing more sophisticated reasoning and consequent complex behaviours. Mapping of a robot’s surroundings, for example, is achieved by means of state variables describing places or locations in this map.
3 Uncoupling dependencies between time instants without forgetting the past

When answering the ultimate question, it is necessary to take into account past time information, which could rapidly lead to an explosion of memory and computation resource requirements.

In this section, it is proposed to apply a Bayes filter to answer the ultimate question, as the Bayes filter structure allows partial uncoupling of dependencies between time instants. We do not want to lose all information about the past but to limit time dependence to a reasonable depth. This limit is imposed here by applying the Markov assumption. Dependence in time is also reduced by assuming that models are stationary in time.

Bayes filters are employed widely in autonomous robotics, as in Markov Localization and in input/output Hidden Markov Models (HMMs); the filter presented here is augmented with a motor model in the joint distribution.

3.1 Bayesian program

The variables involved in this program are the set of state variables $S$, the set of observation variables $Z$ and the set of motor command variables $M$, for all time instants from zero to an arbitrary time instant $t$, as given by the

Fig. 2. Bayesian Program for a Bayes Filter extended with a motor model.
right side of the Bayesian program for a Bayes filter extended with a motor model.

The Bayes filter joint distribution is shown in the decomposition of Figure 2, and it involves all relevant variables.

The first term inside the product, \( P(S_j | S_{j-1} M_{j-1} \pi_f) \), is called the dynamic model, and it expresses our knowledge about how the state variables evolve over time. The state variables at time instant \( j \) depend on the state variables at the previous time step \( S_{j-1} \) and on how the robot moved in the previous time step \( M_{j-1} \). As the state variables describe world features relevant to the robot’s task, the dynamic model is responsible for identifying changes in world features as time passes and according to the robot’s actions in the environment.

The second term inside the product, \( P(Z_j | S_j \pi_f) \), is the sensor model, and it establishes the relation between sensor observations and the environment state, as described in the previous section.

The sensor model is written based on sensor supplier information. A sensor measure is related to a phenomenon, which is associated with a state variable. If the phenomenon and its associated state variable are known, the sensor measure can be calculated.

The third term inside the product, \( P(M_j | S_j M_{j-1} \pi_f) \), is the motor model, and it defines how robot activity (defined by motor commands) depends on the environment and the specified task (described by the state variables). The motor commands at time instant \( j \) depend not only on the state variables at time instant \( j \) but also on the motor commands at the previous time instant \( M_{j-1} \).

Outside the product, \( P(M^0 S^0 Z^0 | \pi_f) \) is a term for the initial conditions, i.e. the distribution over relevant variables at time instant \( j = 0 \).

We can also see that the first-order Markov assumption is present in both the dynamic and motor models: time dependence has a depth of one time step. The product of terms is a consequence of the stationarity assumption: models do not change with time. They are defined over different variables at each time instant, but the relationships between these variables remain the same for all time steps.

The Bayes filter described in the above joint distribution allows recursive utilization. The mathematical proof of this property of recursiveness is shown in Koike [2005].

Recursive calculation in the above Bayes filter consists of asking three questions, sequentially, of the filter joint distribution. The first question is called prediction, and in this part of the calculation loop, the information about the past is updated using the dynamic model.

The second question is called estimation, and it aims at updating the prediction question result with evidence given by observations.

Once state variable estimates are computed, it is possible to apply these estimations to decide which motor commands are most relevant for the present estimated situation.
The sequence of questions is better visualized in the diagram shown in Figure 3. The dotted line marks the transition from time step $t - 1$ to time step $t$. Prediction is the first question asked at each time step, and it employs state information from the previous time step, as well as motor variable values. The estimation question is then asked of the joint distribution of the filter: the observation values $z^t$ are taken into account to update the state prediction. Next, the motor question is asked, and from the answer to that question, the values of motor commands $m^t$ are decided.

The following equation shows how prediction is calculated in the Bayes filter. It can be seen that the past is summarized by the answer to the estimation question in the previous time step. A summation over state variables at the past time step is also realized, as no decision is taken in relation to values of these variables.

$$P(S^t|m^{0:t-1}, z^{0:t-1}, \pi_f)$$

\[\propto \sum_{S^{t-1}} \left[ P(S^t | S^{t-1}, m^{t-1}, \pi_f) \times P(m^{t-1} | S^{t-1}, m^{t-2}, \pi_f) \times P(S^{t-1} | z^{0:t-1}, m^{0:t-2}, \pi_f) \right]. \tag{2} \]

### 3.2 Analysis and discussion

The first-order Markov assumption is also applied in several approaches in robotics, such as Markov localization and POMDP.

Markov localization, also called an input–output hidden Markov model, is a specialization of dynamic Bayesian networks widely applied in robotics [Fox et al., 1998b, Thrun, 2000]. It has the additional influence of an action variable in the dynamic model but no motor model in the joint distribution.
Bayesian approach to action selection and attention focusing

Partially Observable Markov Decision Processes (POMDP), are an extension of Markov localization, as one additional constraint is added in relation to action decisions [Diard et al., 2003]. In contrast to Markov localization, where no assumption is included about how actions are decided, POMDP proposes to assign a reward value to each possible action to compare the adequacy of actions. This reward value is associated with executing a given action in a given state situation. A policy is produced that prescribes the choice of action for any possible state [Cassandra et al., 1994, Simmons and Koenig, 1995, Kaelbling et al., 1998]. Finding the policy involves maximizing reward in a finite time horizon, and it is very hard to calculate.

Our proposal for action decision can be seen as a simpler alternative to POMDP. While POMDP explicitly considers future time when evaluating a sequence of actions to maximize reward, in our filter decisions, we take only the present time into account. Another difference is that the reward function in POMDP can consider other variables as well as state variables, while in our motor model, only state variables influence motor command decisions.

4 Exploiting the existence of environment areas of interest

The Bayes Filter solution proposed in the previous section removed part of the time dependence by applying stationarity and the first-order Markov assumptions. However, it still presents huge time complexity because of the summation over the state space in the prediction question (equation 2).

In this section, we aim to reduce this complexity by exploiting the existence of areas of interest in the environment that are independent of one another. Each area of interest can provide an expert opinion regarding motor variables, and these opinions are then combined to obtain only one probability distribution for the motor command variables. We propose to impose constraints in the dynamic and sensor model in relation to these areas of interest, resulting in the decomposition of the state and observation variable spaces into independent subsets.

4.1 Bayesian program

In this section, conditional independence in the state and observation spaces is exploited to reduce the cost of executing summations over the state space. While conditional independence can exist in the state and observation spaces, it cannot be applied to the motor command variables. This creates a problem when defining the joint distribution, as the definition of the motor model as

\[
\prod_{i=1}^{N_i} P(M^j|S_i^j, M^{j-1}, \pi_{lf})
\]
Variables. Each state subset \( S \) details, see Pradalier et al. [2003a,b] fusion, and it does not add new knowledge or new assumptions. For more observation space decomposition.

This technique was originally named fusion with diagnosis. In our work, the \( \text{Diagnosis variable} \) is semantically nearer to a coherence variable, and we take the liberty of calling it \textit{coherence-based fusion}.

is not correct according to the rules detailed in the chapter \textit{Basic concepts of Bayesian programming}: variable \( M^j \) appears more than once on the left. However, a product of motor models can be achieved using intermediate variables \( \lambda_i \), called coherence variables. The motor model definition becomes:

\[
P(M^j | \pi_{lf}) \prod_{i=1}^{N_i} P(\lambda_i^j | M^j S_i^j M^{j-1} \pi_{lf}).
\]

Expressing a model using coherence variables is called coherence-based fusion, and it does not add new knowledge or new assumptions. For more details, see Pradalier et al. [2003a,b]\footnote{This technique was originally named fusion with diagnosis. In our work, the \textit{diagnosis variable} is semantically nearer to a coherence variable, and we take the liberty of calling it \textit{coherence-based fusion}.}. Examining the joint distribution shown in the decomposition of Figure 4, the first line inside the time product is the dynamic model, which is decomposed as the product of \( N_i \) terms, each one related to a subset \( S_i \) of the state variables. Each state subset \( S_i, i = 1, \ldots, N_i \) is composed of \( N_x \) state variables.
ables, so that $\sum_{i=1}^{N_s} N_s_i = N_s$, and it is possible to define a dynamic model for each subset.

The second line in the joint distribution is the sensor model. It is also decomposed as a product of $N_s$ terms, and each term uses the same state subsets defined above.

In the third line, the motor command model is now defined using coherence-based fusion.

The condition to allow the joint distribution shown is that the state, observation and coherence variables spaces can be split into $N_s$ mutually exclusive subsets, so that:

$$S^j = \bigcup_{i=1}^{N_s} [S^j_i], S^j_k \cap S^j_m = \emptyset, \forall k, m, \text{ with } 1 \leq k, m \leq N_s \text{ and } k \neq m; \quad (3)$$

$$Z^j = \bigcup_{i=1}^{N_s} [Z^j_i], Z^j_k \cap Z^j_m = \emptyset, \forall k, m, \text{ with } 1 \leq k, m \leq N_s \text{ and } k \neq m; \quad (4)$$

$$\lambda^j = \bigcup_{i=1}^{N_s} [\lambda^j_i], \lambda^j_k \cap \lambda^j_m = \emptyset, \forall k, m, \text{ with } 1 \leq k, m \leq N_s \text{ and } k \neq m. \quad (5)$$

In the last line of the joint distribution, outside the time products, are the initial condition distribution.

The joint distribution in equation 4 defines a filter called a global filter, and it is defined over all relevant variables. Following the partition of the variable space into the mutually exclusive subsets shown in equations 3, 4 and 5, it is possible to derive the existence of elementary filters, one for each variable subset. The demonstration of the equivalence between utilization of global or elementary filters is available in Koike [2005].

The Bayesian program for a general elementary filter can be seen in Figure 5. In the joint distribution, inside the time product, we can see that the terms are now related to only one subset of the variables. The first line contains the dynamic model on the state variables subset $S_i$. In the second line, the sensor model is related to the observation variable subset $Z_i$, and the state variables subset $S_i$.

The third line shows the motor model relating the state variables subset $S_i$ and the motor variables $M$. It is defined here using coherence-based fusion.

In the last line of the elementary filter joint distribution, we can see the probability distribution for the initial conditions of variables relevant to this elementary filter.

Instead of asking questions of a global joint distribution, we now wish to use elementary filters as much as possible. From Figure 6, it can be seen...
that the basic utilization of an elementary filter \( i \) is similar to the utilization shown in the previous section: prediction, estimation and motor questions can be easily identified.

It is demonstrated in Koike [2005] that asking prediction, estimation and motor question of the global filter is analogous to asking prediction, estimation and motor questions of each elementary filter.

The motor question deserves more attention, because it is necessary to combine motor command proposals from each filter to obtain global motor commands with the coherence variable \( \lambda_i^0 \) made equal to one, as only coherent values of motor commands are desired.

In Figure 6, a hexagon with letter F is included to illustrate the need to combine motor command answers. The front diagram is related to elementary filter 1, but fusion is executed using information from all the other filters: in the figure, additional elementary filters are shown as diagrams in the back.

### 4.2 Analysis and discussion

The reduction of time complexity when employing Bayesian modelling approaches is the subject of several research projects.
Solutions proposed in the literature for computing state-related questions (called **computation of belief state**) are mostly compact representations of the state space (as discussed by Boyen [2002]) or approximate inference algorithms (Monte Carlo-based algorithms such as particle filters), described in Thrun [2002b].

The factored Markov decision process (factored MDP) and its counterpart factored partially observable Markov decision process (factored POMDP) both exploit problem-specific structures to represent large and complex dynamic systems compactly [Guestrin, 2003, Poupart, 2005].

The main idea of factored representation in MDP is to split the state space into subsets, disjoint or weakly coupled in relation-state dynamics. Then, algorithms for computing control policy are developed that cope with the state space separation [Parr, 1998, Lane and Kaelbling, 2001, Guestrin, 2003].

Our proposal is an attempt to reduce this complexity by imposing some constraints on dependencies in the dynamic and observation models. These constraints must be verified in the early stages of creating the robot task model.

Our approach is very similar to the compact representation proposed by Boyen [2002], which groups state variable subsets with weak dependencies with other subsets in a cluster. The compact representation is then built using a cluster-structured filter. It is shown that the accumulated error between this compact representation utilization and the complete system diminishes with time according to the choice of clusters: a technique called **contraction**.

---

**Fig. 6. Utilization of elementary filters.**
analysis, based on a stochastic contraction phenomenon, is proposed to analyse different possible choices of clusters so that a good complexity reduction is achieved and the error is made reasonable. Another technique called Projection Analysis is proposed to analyse whether removed dependencies, even if weak, could affect a posteriori estimation over longer time intervals.

We argue that the choice of state and observation variables is decisive for defining dependencies in dynamic and observation models. More specifically, referential choice is crucial when dealing with dependencies. For example, state variables defined as the distance from robot to significant features in the environment can fulfil the necessary constraints.

Compared with the Bayes filter proposed in the previous section, programming becomes easier inside each elementary filter. The independence condition between state and observation variables is enough to ensure that programming dynamic and sensor models in each elementary filter can be isolated from all other elementary filters.

Even if motor command variables are common to all elementary filters, motor models are written individually for each elementary filter. Knowledge about other elementary filters is not required, and whenever variable subset values have no relevant information about motor commands, the uniform distribution is applied.

5 Building behaviour from a collection of motor patterns

The utilization of elementary filters proposed in the previous section helps to reduce the computation burden and also simplifies writing motor models. However, the choice of motor commands can become very complex as the number of elementary filters increases.

Motor commands can be grouped in collections of simple and independent motor patterns: follow an object, go straight ahead, and so on. Each pattern is called here a basic behaviour, and in this section, we propose the addition of a behaviour selection mechanism to the Bayes filter of the previous section.

Selection of the most pertinent behaviour is made by employing fusion of elementary filter proposals, which are the results of local strategies of behaviour choice in the elementary filters, according to their own subsets of variables.

5.1 Bayesian program

To coordinate the execution of basic motor patterns, we propose to add a set of behaviour variables, denoted $B^t$, common to all elementary filters. The semantics of these variables are related to the patterns found in the motor commands: each variable corresponds to a group of motor patterns, one value for each pattern. One motor pattern corresponds to a set of motor commands applied simultaneously on motor actuators, and it can depend on state variables.
Bayesian approach to action selection and attention focusing

Fig. 7. Bayesian program of an elementary filter, extended for behaviour coordination.

Behaviour variables are common to all elementary filters, and coherence-based fusion is applied to describe the behaviour selection model.

The joint distribution for a general elementary filter is as shown in Figure 7.

Inside the time product, the first line consists of the dynamic model, unchanged from the dynamic model in the previous section. The second line contains the sensor model, also unchanged.

The third line shows the behaviour model, in the coherence-based fusion form $P(\beta^t_i | B^t_i S^t_i B^{t-1}_t \pi_i)$.

Behaviour at the present time depends on the behaviour selected at the previous time and on the current states.

The relation between selected behaviours at the previous and present times aims to ensure continuity (called persistence) in the execution of a behaviour, even if states show a reduction of environment stimuli. Dependence on the present state means favouring reactivity to environment changes.
The motor command model, $P(x_i|\lambda^tS_i^tB^tM^{t-1} \pi_i)$ in the fourth line of the joint distribution now includes the dependence on the behaviour variable.

It is interesting to consider the role of the behaviour variable in the motor model. Behaviour variables take control of switching between motor models: each value corresponds to a motor pattern defined by a specific motor model. Considering only one behaviour variable with $n_b$ possible values, we have the following motor model:

$$P(M^t|S^t_iB^tM^{t-1} \pi_i) = \begin{cases} 
  P(M^t|S^t_iM^{t-1} [B^t = b_1] \pi_i); \\
  P(M^t|S^t_iM^{t-1} [B^t = b_2] \pi_i); \\
  \ldots \\
  P(M^t|S^t_iM^{t-1} [B^t = b_{n_b}] \pi_i).
\end{cases}$$  \hfill (6)

Each term in the above equation corresponds to one motor pattern:

- $P(M^t|S^t_iM^{t-1} [B^t = b_1] \pi_i)$ is the motor model of behaviour $b_1$;
- $P(M^t|S^t_iM^{t-1} [B^t = b_2] \pi_i)$ is the motor model of behaviour $b_2$, and so on.

The last line of the joint distribution shows the initial conditions of all relevant variables.

Compared with the utilization in the previous section, one more question is asked of each elementary filter, called the behaviour selection question.

The Bayes filter utilization is summarized in Figure 8. Fusion of elementary filter answers is executed twice: first for behaviour selection, then for motor commands. State variable estimation is done after behaviour selection, which ensures the recursive calculation for the next time step.

5.2 Analysis and discussion

To decide which motor pattern to apply, all environment features must be taken into account, and for that reason, the behaviour selection variable is common to all elementary filters. To benefit from the advantages of elementary filters, the behaviour model in each elementary filter includes dependence only on that filter’s state variables, and utilization of the coherence-based fusion form when describing the behaviour selection model allows posterior fusion of elementary filter proposals.

The behaviour selection model in each elementary filter implements the selection strategy of this filter, based on its own state variables. Consequently, knowledge about state variables of other filters is not available. Taking into account the posterior fusion of elementary filter proposals, proscriptive programming emerges as an intuitive way to build the behaviour selection model. The benefits of programming proscriptively using probability distributions, especially when they are combined by a fusion method, were discussed by Lebeltel [1999, page 178] and Pradalier et al. [2003a].
When choosing parametric forms for the behaviour selection model, closed-form expressions can be employed to describe the probability distributions. When this is not possible, a probability table can be employed. It can be the hardest form to tune, because of the large number of parameters, and this approach is discussed here.

To fill a probability table for the behaviour selection model while avoiding manual definition of all probability values, levels are chosen for classifying behaviours for a given situation: highly recommended, recommended, not relevant, unwise, strictly prohibited. Each situation is defined by associating a probability value for each possible behaviour selected in the previous time step with the possible values of the state variables at the present time. The strictly prohibited probability value must be as small as necessary to prevent its selection, even if another filter recommends it. These levels are also easily related to the knowledge applied to build the behaviour model. Instead of having a very large number of probability values to adjust (approximately equal to the amount of memory necessary to store the behaviour model in memory, as shown above), only five parameters must be tuned: the probability values for each of the above levels.

6 Focusing attention on relevant environment features

Until now, no details have been given about how observations are acquired from the environment. All observations are assumed to be available at the moment when the behaviour selection question is asked.
Observations are defined as the results of processing sensor raw data. Sensors like vision often provide a huge amount of data, from which several features can be extracted by applying diverse processing algorithms. To have all the observations required for the Bayes filter, it can be necessary to execute different processing algorithms over sensor raw data, which is very time consuming.

In this section, we propose to reduce the overhead necessary to calculate all observations by applying the concept of attention, where only the most relevant perceptual observations are calculated and applied. Relevance of an observation is defined in relation to the pertinence of the associated feature to the behaviour selected for robot execution.

6.1 Bayesian program

To choose more relevant observations, a set of attention selection variables $C^t$ is defined. Each variable in this set corresponds to one physical sensor or incoming flow of data. For a given variable, each value indicates a different processing method to be applied to the raw data and the observation that results.

Attention variables $C^t$ are common to all elementary filters, and the attention model is defined by applying coherence-based fusion. A coherence variable set $\alpha^t$ is then necessary, one variable in each elementary filter for each variable in $C^t$.

In the joint distribution shown in Figure 9, the first term shows the dynamic model, which is unchanged from the Bayes filter in the previous section.

The observation model $P(Z^t_j|S^t_j C^t \pi_i)$ in the second line of the time product now includes the attention variable.

The attention variable is necessary in the sensor model to indicate valid observations. If observations are valid, the sensor model applies. If an observation is not available, a uniform distribution is used. In other words, it is only possible to establish a relation between sensor observations and state variables if those observations are available. Attention variables then control switching between a real sensor model and a uniform distribution.

The third line inside the product presents the behaviour selection model, which is unchanged from the previous version.

The next line introduces the attention selection model $P(\alpha^t_i|C^t S^t_i B^t \pi_i)$, where the attention variable depends on the present time states but also, and mainly, on present behaviour.

The last line inside the time product shows the motor command model, unchanged from the previous section’s Bayes filter, and the last line gives the initial conditions for all relevant variables.

Utilization of this Bayes filter is similar to that in the previous section, with the addition of the attention selection question. This question is asked of each elementary filter after prediction: the answers are then combined, resulting
in the global attention selection distribution; attention variables values are drawn from this global distribution.

Utilization of the Bayes filter is illustrated in Figure 10. A new symbol (similar to a funnel) is employed in this figure to indicate that not all observations $Z_i^t$ are available.

It is important to analyse in detail the relation between attention and behaviour. As attention aims to focus perception into features relevant to the present robot action, it would be necessary to know which behaviour is under execution before deciding attention variable values. However, it is not possible to infer the behaviour to be executed if no clue from perception is already available.
This problem is solved in a rather natural way by Bayesian inference. Considering the answer to the attention selection question in the elementary filter, we have:

\[
P(C^t B^t | z_{t-1}^{0:t-1} c^{0:t-1} b^{0:t-1} m^{0:t-1} \lambda_i^{0:t-1} \beta_i^{0:t-1} \alpha_i^{0:t-1} \pi_i) \\
\propto \sum_{B^t} \prod_{i=1}^{N_i} \{P(C^t B^t | z_{t-1}^{0:t-1} c^{0:t-1} b^{0:t-1} m^{0:t-1} \lambda_i^{0:t-1} \beta_i^{0:t-1} \alpha_i^{0:t-1} \pi_i)\}. \quad (7)
\]

In the left part of the question, it can be seen that not only the attention variables \(C^t\) but also the behaviour variables \(B^t\) are present. As the behaviour variables are not known, they are inferred in the attention question.

The answers to the attention question from the elementary filters are combined using the coherence-based fusion method, and the global attention expression is:

\[
P(C^t | z_{t-1}^{0:t-1} c^{0:t-1} b^{0:t-1} m^{0:t-1} \lambda_i^{0:t-1} \beta_i^{0:t-1} \alpha_i^{0:t-1} \pi_i) \\
\propto \sum_{B^t} \prod_{i=1}^{N_i} \{P(C^t B^t | z_{t-1}^{0:t-1} c^{0:t-1} b^{0:t-1} m^{0:t-1} \lambda_i^{0:t-1} \beta_i^{0:t-1} \alpha_i^{0:t-1} \pi_i)\}. \quad (8)
\]

At the moment of combining these proposals, only the attention variable is relevant. Consequently, it is desired to find the global distribution for the
Bayesian approach to action selection and attention focusing 203

attention selection variables only. To find this global distribution, the answers from the elementary filters are combined (by means of their product distributions), which means that both the attention and behaviour distributions are combined. As no observation is still available, this global distribution over behaviour variables is actually a prediction of behaviour variable values.

A summation over behaviour variables is also executed. This sum is necessary to weight the attention model for all possible values of behaviour variables.

Once the global distribution over the attention selection variables is found, values for the attention variables are drawn, and the respective processing methods are applied to the sensor raw data. Corresponding observation variables are then assigned values resulting from the data processing, and valid observations are available.

These observations are then employed in the right part of the behaviour questions asked of the elementary filters. The answers are combined, and the result is no longer the prediction of possible behaviour but the global distribution over behaviour variables. From this global distribution, the values of behaviour variables are drawn.

6.2 Analysis and discussion

Attention deployment has been the object of research and applications in artificial systems, in different scientific domains, with different aims and concerns. More specifically for robotic applications, attention models of brain visual processing and active/purposive computer vision have several aspects in common. Above all, both methodologies were developed to deal with complexity in visual systems and to optimize the utility of the acquired information.

Typical applications of these methodologies (some examples are Mihaylova et al. [2002], Fox et al. [1998a], Soyer et al. [2003], S.Vijayakumar et al. [2001], to cite only a few) aim to model an attention system, or to add an attention model to a more complex behaviour system in simulation, or even to verify attention mechanisms in simple robotic tasks. Active sensing, which consists of choosing which sensors to use and where to point them, is mostly employed for plan-based navigation, and sensors, including vision, are used to identify landmarks or environment features.

Our approach aims to point out the importance of the reciprocal influence of attention selection and behaviour coordination. To decide correctly what to do, it is necessary to have new information about the robot’s surroundings; on the other hand, to select the most useful information, we must know what the robot is going to do.

The Bayes filter recursive loop solves this dilemma: prediction of behaviour allows us to decide on attention, and information acquired after focusing attention is used to estimate and draw a definite behaviour to be executed.

Utilization of elementary filters helps to reduce computational effort but generates another problem: mathematically, it is possible to have behaviour
prediction executed locally (in every filter) or globally (during attention fusion).

There are two reasons for preferring the global prediction form for the attention question. The first is that the calculations are lighter, because only one summation over the behaviour variable space is required. The second reason is that attention results from a global prediction of behaviour, taking into account the combination of predictions from all elementary filters.

7 Robot and simulation experiments

This section presents and discusses the results of practical utilization of the presented solutions to the ultimate question. Initially, a simulation arena was created to test and illustrate a possible application of the frameworks developed in Sections 3 to 6. The solution proposed in Section 5 was also implemented to control the behaviour of a real mobile robot.

In the simulation, the robot is inside an arena, without obstacles. It can move freely, although it cannot go through the walls. Our robot is not alone: a bad predator is there, and its goal is to capture the robot. A prey and a nest are also present. Specified behaviour for the simulated robot consists of escaping from the predator, chasing the prey and eventually going to the nest, because when it is in the nest, also called home, the robot is safe from the predator.

Figure 11 illustrates the simulation arena running the filter presented in Section 6. Simulation results demonstrate the feasibility of programming methodologies for this framework and enabled their further exploration.

![Simulation arena. The robot is black, the prey is green, home is blue and the predator is red.](image)

Robot experiments were developed using the BIBA robot, shown in Figure 12. Our robot wanders around in an indoor, office-like environment, avoiding
obstacles. Whenever it perceives a predator, it stays motionless if the predator is far away, or it escapes in the opposite direction from the predator if it is close. When a prey is seen, the robot chases it, and when the prey is close enough, the robot captures it\(^5\).

As security in relation to obstacles is essential in a real application, the framework was implemented to produce the desired motor commands for the robot. A low-level Bayesian program, employing prescriptive programming, is responsible for executing the desired motor commands while respecting obstacle avoidance constraints.

The robot's resulting behaviour was considered to be compatible with the specification. Desired characteristics, such as reactivity to predator appearance and persistence in chasing and obeying, were observed in several situations.

---

7.1 Programming methodology

The theoretical structure of the filters presented in Section 5 seems intricate, but implementation is not particularly difficult. This section describes general guidelines employed when programming both simulations and real robot experiments.

One of the basic assumptions of the Bayesian robot programming approach is that the description and utilization phases are independent. When writing

\(^5\) A video of this behaviour may be found at http://www.bayesian-programming.org/videoB1Ch8-1.html
terms in the joint distribution decomposition, only previous knowledge about context and the situation being modelled is employed: it is important not to predict utilization. The same program description can answer several questions regarding any of its relevant variables.

In spite of the number of variables involved and the joint distribution’s size (54 variables and 36 terms), the programming task is simplified by the independence of the elementary filters. Assumptions necessary to have disassociated elementary filters (see equations 3 and 4) are hard to fulfil, but once these conditions are met, the consequent independence for programming each filter is very useful. Each term in the joint distribution is then programmed employing the knowledge provided by the dependent variables.

For terms involving common variables (behaviour and motor commands, for example), programming can become a delicate task. Because the robot’s resulting behaviour comes from global distributions over common variables, it appears that it might be necessary to balance the probability values within each elementary filter so that the global distributions correspond to the specified behaviour.

In fact, the elementary filter’s independence is also essential when dealing with common variables, mainly because coherence-based fusion of common variables provides a satisfying combination of elementary proposals. Terms involving common variables in each elementary filter must therefore consider only local information, as state variables. Situations where local information is enough to recommend or forbid a specific value for a common variable are easily programmed: recommended values have high probability values, and forbidden ones have very low probability values.

In other situations, local information is not sufficient. It is important neither to prohibit a behaviour (as another filter could indicate that it is interesting) nor to recommend it (as it is not possible to know whether it is really indicated). Medium probability values are the most appropriate in these cases. Section 5.2 describes how this technique can be applied when programming the behaviour model. In Koike [2005], the methodology outlined here is described in full detail.

The method proposed was used in programming both the simulation and real robot experiments, and it is considered to be effective in reducing complexity of programming.

8 Discussion

The framework proposed here presents several interesting aspects, especially from the cognitive viewpoint. There is no space here to develop these aspects fully, but some of them are discussed briefly in this section.
Making a decision about internal states

The Bayes filter mechanisms presented in Sections 3 to 6 show a compromise between (i) the need to take into account the history of sensory inputs, motor decisions and internal variables to solve the ultimate question, and (ii) the impossibility of memorizing and processing the great quantity of information involved. Making a decision about internal states would mean using the decided values instead of the probability distributions; this would be equivalent to losing all past information and destroying the Bayes filter recursive estimation utility.

Sensory–motor systems, however, impose a decision on motor commands at each time step, and each of these decisions is a commitment. This motor commitment “simplifies” the world: perception is oriented, fewer acting options are opened for the future, and reasoning is more tractable. Regarding behaviour and the attention variable, the decision about also means a commitment that extends its influence over the state variables at the moment of estimation.

Switching versus weighting

The local sensory–motor modules employed in this section are suggested as one possible mathematical implementation of the idea of motor synergies as proposed by Bernstein [Bernstein, 1967]. The introduction of the behaviour variable $B$ greatly simplifies the intrinsic complexity of the motor models. In fact, $B$ acts as a selector between the synergies, and each of these synergies becomes much simpler and less complex to specify than the whole motor model.

For efficiency reasons, we choose in this work to make a decision on $B$ to decide which synergy to apply at each time step. It is also possible to avoid this decision by making the $B$ variable a state variable, known only by a probability distribution. This alternative solution would imply marginalizing on variable $B$ in all questions asked: the probability on $B^t$ would be obtained as a weighted sum of the different dynamical models, where the weights are the probabilities estimated from the past internal states updated by sensor models. We think that this process could provide some mathematical insights into the discussion of switching versus weighting, and we propose that this matter should be investigated further.

Perception and attention

From a cognitive point of view, the central nervous system does not allocate uniform effort to all the possible sensory data processing. It rather focuses its attention on some specific sensors according to its present internal state and to the present selected synergy.
Vision begins in the eyes: an image is projected on the retina, and from there, specialized cells capture, convert and transfer information deep into the brain. Segregation of information at the lowest levels makes it possible to decrease the amount of information to be transmitted and processed by parts of the brain. The visual world is then analysed by decomposition in predefined categories: colour, form, motion. Recomposition is realized in relation to aims: advanced areas of visual processing can actually change the properties of the first layers to adapt to the present interest [Berthoz, 2002].

The first levels of perception (sensory data processing) are then adjusted according to this early attention selection, and perception is tuned to the stimulus necessary for the previewed behaviour. Incoming sensor data are understood according to the needs of the expected action.

In our proposal, attention is modelled as an early decision in the behaviour selection process. Actually, the decision as to which features are to be perceived is based on anticipation of which behaviour will be selected. Equations 7 and 8 illustrate this interesting property, as variable \( C \) is obtained as a weighted sum on all the possible values of \( B \) variable.

9 Conclusions

In this chapter, we propose a succession of assumptions and simplifications related to controlling autonomous sensory-motor systems. Our main contribution is a framework for robot programming based on strict mathematical definitions and featuring Bayesian-based action selection and selective perception.

The proposed framework was applied in a real robot application, using a programming methodology to define \( a \) priori all free parameters, and the resulting behaviour is acceptable in relation to the desired behaviour specification. This application also makes it possible to examine the filter working details in a realistic situation.

Contributions of the framework proposed in this chapter are mainly:

- the addition of the motor command in the Bayes filter recursive loop of calculations;
- the inclusion of an action selection mechanism in the Bayes filter;
- the association of a Bayesian selective perception with action selection; and
- the proposal of a programming method.

Among these perspectives, we can cite two in particular. Mapping and planning are considered essential for navigation in autonomous robotics according to several studies [Thrun, 2002a]. In relation to the framework proposed in this thesis, the utilization of maps, especially Bayesian maps as described in the chapter Bayesian maps: probabilistic and hierarchical models for
Bayesian approach to action selection and attention focusing is under study. However, because of the large number of state variables required for maps, which are usually highly connected models, it can be particularly difficult to respect the constraints described in equations 3 and 4. Can coherence-based fusion help in this situation, as with the attention, behaviour and motor variables presented in this work? Are Bayesian maps and their operators still valid in a system where the map representation is one of the elementary filters? Is planning feasible in this context?

Another point we would like to study further is related to active sensing. Selective perception as included in the proposed framework does not consider that motor variables may be necessary to control perception data: to move the head of the robot to search for an object, for example. Several studies include active perception in maps built into Kalman or particle filters, and the combination of active and selective perception is an interesting cognitive aspect to be investigated.

10 Acknowledgements

This work was sponsored by the European project BIBA, and C. Koike was financially supported by CAPES/Brazilian Government.

References


Part III

Industrial applications
BCAD: a Bayesian CAD system for geometric problems specification and resolution

Kamel Mekhnacha\textsuperscript{1} and Pierre Bessière\textsuperscript{2}

\textsuperscript{1} PROBAYES  
\textsuperscript{2} CNRS - Grenoble Université

We present “BCAD”, a Bayesian CAD modeller for geometric problem definition and resolution. This modeller provides tools for (i) modelling geometric uncertainties and constraints, and (ii) solving inverse geometric problems while taking into account the propagation of these uncertainties. The proposed method may be seen as a generalization of constraint-based approaches in which we explicitly model geometric uncertainties. Using our methodology, a geometric constraint is expressed as a probability distribution on the system parameters and the sensor measurements instead of as a simple equality or inequality. To solve geometric problems in this framework, we propose the Monte Carlo Simultaneous Estimation and Maximization (MCSEM) algorithm as a resolution technique able to adapt to problem complexity. Using three examples, we show how to apply our approach using the BCAD system.

1 Introduction

The use of geometric models in robotics and CAD systems necessarily requires a more-or-less realistic model of the environment. However, the validity of calculations with these models depends on their degree of fidelity to the real environment and the capacity of these systems to represent and take into account possible differences between the models and reality when solving a given problem.

This chapter presents a new methodology based on Bayesian formalism to represent and handle geometric uncertainties in robotics and CAD systems. The proposed approach may be seen as a generalization of constraint-based approaches. This generalization consists of explicitly taking into account the uncertainties in models. A constraint on a relative pose between two frames or on a shape parameter is represented by a probability distribution on the parameters instead of a simple equality or inequality. In this framework, modelling information given by the programmer and the measurements obtained using sensors are represented and used in a homogeneous way. For a given
problem, all the information that we include on the geometric model and on the responses of the sensors is used optimally by applying Bayesian reasoning.

The principle of the proposed method is to infer, for a given problem, the \textit{a posteriori} marginal distribution of the unknown parameters using the probability calculus. The original geometric problem is reduced to an optimization problem over this distribution to find a solution with maximum probability. In the general case, this marginal probability may contain an integral on a large dimension space.

The resolution method used to solve this integration/optimization problem is based on an adaptive genetic algorithm. The problem of integral numerical estimation is approached using a stochastic Monte Carlo method. The accuracy of this estimation is controlled by the optimization process to reduce computation time. This approach is called the Monte Carlo Simultaneous Estimation and Maximization algorithm (MCSEM).

Many robotic applications are instances of inverse geometric problems in the presence of uncertainties, for which our method is well suited. The simplicity of our specification method and the robustness of our resolution method make our approach applicable to such applications [Mekhnacha et al., 2000, 2001], including:

- kinematics inversion under geometric uncertainties using possibly redundant mechanisms;
- robot and sensor calibration;
- parts' pose and shape calibration using sensor measurements;
- robotic work-cell design to obtain a configuration that can accomplish a given task with maximum accuracy.

Extensive experimentation on the approach was made possible thanks to the design and the implementation of the BCAD modeller. Three examples of this experimentation are presented in this chapter.

This chapter is organized as follows. We first report related work in Section 2. In Section 3 we present our specification methodology and show how to formulate an optimization problem. In Section 4 we describe our numerical resolution method. Section 5 is an overview of the implementation of our modeller. We present three examples to illustrate our approach in Section 6 and give some conclusions and perspectives in Section 7.

This chapter summarizes the work initiated in the Ph.D. thesis of Kamel Mekhnacha [Mekhnacha, 1999].

2 Related work

The representation and handling of geometric uncertainties is a central issue for robotics and mechanical assembly. Since the precursor work of Taylor [Taylor, 1976], in which geometric uncertainties were taken into account in
the robot manipulator’s planning process, numerous approaches have been proposed to model these uncertainties explicitly.

Methods modelling the environment using “certainty grids” [Moravec, 1988, Coué et al., 2003, Yguel et al., 2006] and those using uncertain models of motion [Lozano-Pérez, 1987, Alami and Simeon, 1994] have been used extensively, especially in mobile robotics.

Gaussian models to represent geometric uncertainties and to approximate their propagation have been proposed in manipulator programming [Puget, 1989] as well as in assembly [Sanderson, 1997]. Kalman filtering is a Bayesian recurrent implementation of these models. This technique has been used widely in robotics and vision [Zhang and Augeras, 1992], and particularly in data fusion [Bar-Shalom and Fortmann, 1988]. Gaussian model-based methods have the advantage of economy in the computation that they require. However, they are only applicable when a linearization of the model is possible. Another limitation of these methods is their inability to take inequality constraints into account.

Geometric constraint-based approaches [Taylor, 1976, Owen, 1996] using constraint solvers have been used in robotic task-level programming systems. Most of these methods do not represent uncertainties explicitly. They handle uncertainties using a least-squares criterion when the solved constraint systems are overdetermined. Where uncertainties are explicitly taken into account (as in Taylor’s system), they are described solely as inequality constraints on possible variations.

Bayesian networks (BN) [Jensen, 2001] are modern tools for general-purpose probabilistic model specification and resolution. However, these methods are seldom applicable in complex problems involving a large number of variables. Moreover, the proposed algorithms are not applicable for continuous cases.

3 Specification of probabilistic geometric constraints

In this section, we describe our methodology by giving some concepts and definitions necessary for probabilistic geometric constraint specification. We further show how to derive an objective function to maximize from the original geometric problem.

3.1 Probabilistic kinematic graph

A geometric problem is described as a “probabilistic kinematic graph”, which we define as a directed graph having a set of $n$ frames $S = \{S_1, \cdots, S_n\}$ as vertices and a set of $m$ edges $A = \{A_{i,j_1}, \cdots, A_{i,m}\}$, where $A_{i,j_k}$ denotes an edge between the parent vertex $S_{i}$ and its child $S_{j_k}$, and represents a probabilistic constraint on the corresponding relative pose. We call these edges “probabilistic kinematic links”. A given edge may describe:
• a modelling constraint (a piece of knowledge) on the relative pose of the parent frame and its child;
• a sensor measurement of the pose of a given entity; or
• a constraint that we wish to satisfy to solve the problem (an objective value with a given precision, for example).

Each edge $A_{i_{e_{jk}}}$ is labelled as follows.

1. A probability distribution $P(Q_{i_{e_{jk}}})$, where $Q_{i_{e_{jk}}}$ is the relative pose vector (six-vector) $Q_{i_{e_{jk}}} = (t_x t_y t_z r_x r_y r_z)^T$. The first three parameters of this six-vector represent the translation, while the remaining three represent the rotation.

2. Possible equality/inequality constraints ($E_k(Q_{i_{e_{jk}}}) = 0, C_k(Q_{i_{e_{jk}}}) \leq 0$).
   These constraints represent possible geometric relationships between the two geometric entities attached to these two frames. Their shapes depend on the type of the geometric relationship. We implement several relationships between geometric entities in this work, such as points, polygonal faces, edges, spheres and cylinders. The details on equality/inequality constraints induced by these relationships can be found in [Mekhnacha, 1999].

3. A “status” six-vector describing, for each parameter of $Q_{i_{e_{jk}}}$, its role (nature) in the problem. A status can take one of the three following values.
   • Unknown (denoted by X) for parameters representing the unknown variables of the problem. Their values must be found to solve the problem.
   • Free (denoted by L) for parameters with values that are only known with a probability distribution. This allows us to express uncertainties on the model.
   • Fixed (denoted by F) for parameters having known fixed scalar values that cannot be changed.

Fig. 1. Example of a cycle in the kinematic graph.
In the general case, the kinematic graph may contain a set of cycles. The presence of a cycle represents the existence of more than one path between two vertices (frames) of the graph. To ensure the geometric coherence of the model, the computation of the relative pose between these two frames must give the same value for all paths. For each cycle containing \( k \) edges (see Fig. 1), we have:

\[
T_{S_iS_i} = T_{S_{i+1}S_{i+2}} * \ldots * T_{S_{k}S_{k-1}} S_{k} * T_{S_1S_2} * \ldots * T_{S_{i-1}S_i} S_{i} = I_4,
\]

where \( T_{ij} \) is the \( 4 \times 4 \) homogeneous matrix corresponding to the pose vector \( Q_{ij} \), \( I_4 \) is the \( 4 \times 4 \) identity matrix and \( d_{ij} \in \{-1, 1\} \) is the direction in which the edge \( A_{ij} \) has been used.

We call these additional equality constraints the “cycle-closing constraints”. They are global constraints involving, for each cycle, all the parameters it contains. The minimal number of cycles allowing coverage of a connected graph having \( n \) vertices and \( m \) edges is \( p = m - n + 1 \) [Gondran and Minoux, 1990]. Consequently, we obtain \( p \) cycle-closing constraints for a given problem.

Recent versions of BCAD extend the notion of “kinematic graph” to include shape parameters in the model. Examples of such shape parameters are as follows.

- The radius of a sphere (see example 6.3).
- The position of a vertex in a mesh primitive.

3.2 Objective function

Given a probabilistic kinematic graph, we are interested in constructing a marginal distribution over the unknown parameters (parameters having the unknown status) of the problem. Maximizing this distribution will provide a solution to the problem.

We define the following sets of propositions.

- A set of \( p \) propositions \( \{K_i\}_{i=1}^p \) such as: \( K_i \equiv \text{“cycle } c_i \text{ is closed”} \).
- A set of \( m \) propositions \( \{H_k\}_{k=1}^m \) such as: \( H_k \equiv \text{“} C_k(Q_{i_k j_k}) \leq 0 \text{ and } E_k(Q_{i_k j_k}) = 0” \).

If we denote the unknown parameters of the problem by \( X \), a solution to a problem is a value of \( X \) that maximizes the marginal distribution

\[
P(X|\mathcal{H}_1 \cdots \mathcal{H}_m, \mathcal{K}_1 \cdots \mathcal{K}_p).
\]

If we denote by \( L' \) the concatenation of the parameters having status L and by \( X \) the concatenation of the parameters having status X, we can write using the probability calculus:
\[ P(X|H_1 \cdots H_mK_1 \cdots K_p) \propto \int dL' \ P(XL'H_1 \cdots H_mK_1 \cdots K_p) \]
\[ = P(X) \int dL' \ P(L')P(H_1 \cdots H_mK_1 \cdots K_p|XL'). \]

To use the global equality constraints (eq. 1), we take for each cycle \( c_i, \ i = 1 \cdots p \) a pose vector that we rename \( O_i \) (Fig. 1). This pose vector is chosen so that it contains no parameters having the X status. Equation 1 allows us to compute the value of \( O_i \) using the values of all the other pose vectors pertaining to \( c_i \):

\[ O_i = Q_{S_1s_k} \\
= F_i \left( Q_{S_1s_2}, Q_{S_2s_3}, \ldots, Q_{S_{k-1}s_k} \right) \\
= \text{vect} \left( \left( \text{mat} \left( Q_{S_1s_2} \right) \right)^{d_{s_1s_2}} \ast \ldots \ast \left( \text{mat} \left( Q_{S_{k-1}s_k} \right) \right)^{d_{s_{k-1}s_k}} \right), \]

where:
- \( \text{vect} \) is the function allowing the derivation of a pose vector from the corresponding \( 4 \times 4 \) homogeneous matrix;
- \( \text{mat} \) is the function allowing the derivation of a \( 4 \times 4 \) homogeneous matrix from the corresponding pose vector; and
- \( d_{ij} \in \{1, -1\} \) denotes the direction in which the edge \( A_{ij} \) has been used.

Using this equality constraint cancels the integrals over the parameters of \( L' \) that pertain to \( O_i \), because the integrand takes a non-null value only for the points that respect eq. 1.

For each edge \( A_{ij} \), if we denote by \( L_{ij} \) the set of parameters having status \( L \) and by \( X_{ij} \) the parameters having status \( X \), we can write, using appropriate independence assumptions, the following general form:

\[ P(X|H_1 \cdots H_mK_1 \cdots K_p) \propto P(X)I(X), \]

where

\[ I(X) = \int dL \\
P(L_{i_1j_1})P(H_1|X_{i_1j_1}L_{i_1j_1}) \\
\vdots \\
P(L_{i_{m-p}j_{m-p}})P(H_{m-p}|X_{i_{m-p}j_{m-p}}L_{i_{m-p}j_{m-p}}) \\
P_{O_1}(F_1(X, L))P(H_{m-p+1}|F_1(X, L)) \\
\vdots \\
P_{O_p}(F_P(X, L))P(H_m|F_P(X, L)). \quad (2) \]

For each cycle \( c_i, \ i = 1 \cdots p \), \( P_{O_i} \) denotes the distribution over \( O_i \), while \( L \subset L' \) is the concatenation of \( L_{i_1j_1}, \ldots, L_{i_{m-p}j_{m-p}} \).
The distribution $P(X)$ is called the \textit{a priori} distribution over the unknown parameters $X$ (before incorporating the constraints), while the distribution $P(X|\mathcal{H}_1 \cdots \mathcal{H}_m K_1 \cdots K_p)$ is called the \textit{a posteriori} distribution over $X$ (after incorporating the constraints).

For each $A_{ik,jh}, k = 1, \cdots, m - p$, marginalizing (by integration) over the free parameters $L_{ik,jh}$ allows us to take into account the propagation of the uncertainties expressed using the distribution $P(L_{ik,jh})$ \textit{corrected} using the local constraints $\mathcal{H}_k$.

Maximizing the \textit{a posteriori} distribution $P(X|\mathcal{H}_1 \cdots \mathcal{H}_m K_1 \cdots K_p)$ provides the “Maximum \textit{A Posteriori}” (MAP) solution of the problem.

The Bayesian program corresponding to this problem is schematized in Fig. 2.

\begin{verbatim}
Fig. 2. The Bayesian program corresponding to a probabilistic geometric problem.
\end{verbatim}

4 Resolution method

We showed in the previous section how to express a geometric problem as an integration/optimization problem:

$$X^* = \max_X \left[ P(X|\mathcal{H}_1 \cdots \mathcal{H}_m K_1 \cdots K_p) \right].$$

In this section, we will present the practical numerical methods that we used to solve these two problems.
4.1 Numerical integration method

Integral calculus is a central issue in Bayesian inference. Unfortunately, analytic methods for integral evaluation seem very limited in real-world applications, where integrands may have complex shapes and integration spaces may have very high dimensionality.

Domain subdivision-based methods (such as trapezoidal or Simpson’s methods) are often used for numerical integration in low-dimensional spaces. However, these techniques are poorly adapted for high-dimensional cases.

Monte Carlo methods for numerical estimation

Monte Carlo (MC) methods are powerful stochastic simulation techniques that may be applied to solve optimization and numerical integration problems in large dimensional spaces. Since their introduction in the physics literature in the 1950s, Monte Carlo methods have been at the centre of the recent Bayesian revolution in applied statistics and related fields, including econometrics [Geweke, 1996] and biometrics. Their application in other fields such as image synthesis [Keller, 1996] and mobile robotics [Dellaert et al., 1999] is more recent.

Principles

In this section, we will generally assume that \( X \) is a \( k \)-dimensional vector with real components.

The aim of Monte Carlo methods for numerical integration is to approximate efficiently the \( k \)-dimensional (where \( k \) can be very large) integral:

\[
I = \int P(X)g(X) \, d^k X.
\]  

(3)

Assuming that we cannot visit every location \( x \) in the state (integration) space, the simplest solution that we can imagine to estimate the integral (3) is to sample the integration space uniformly and then estimate \( I \) by \( \hat{I} \):

\[
\hat{I} = \frac{1}{N} \sum_{i} P(x^{(i)})g(x^{(i)}),
\]

where \( \{x^{(i)}\}_{i=1}^{N} \) are randomly drawn in the integration space.

Because high-dimensional probability distributions are often concentrated on a small region \( T \) of the state (integration) space \( X \), known as its “typical set” [MacKay, 1996], the number \( N \) of points drawn uniformly for the state (integration) space \( X \) must be sufficiently large to cover the region \( T \) containing most of the probability mass of \( P(X) \).

Instead of exploring the integration space uniformly, Monte Carlo methods try to use the information provided by the distribution \( P(X) \) to explore this
space more efficiently. The main idea of these techniques is to approximate the integral (3) by estimating the expectation of the function $g(X)$ under the distribution $P(X)$

$$I = \int P(X)g(X) \, d^k X = \langle g(X) \rangle.$$  

Clearly, if we are able to obtain a set of samples $\{x^{(i)}\}_{i=1}^N$ ($k$-vectors) from the distribution $P(X)$, we can use these samples to find the estimator

$$\hat{I} = \frac{1}{N} \sum_{i} g(x^{(i)}). \tag{4}$$

As the number of samples $N$ increases, the variance of the estimator $\hat{I}$ will decrease as $\sigma^2/N$ where $\sigma^2$ is the variance of $g$:

$$\sigma^2 = \int P(X)(g(X) - \langle g(X) \rangle)^2 \, d^k X.$$  

Using Monte Carlo methods for our application

Using a Monte Carlo method to estimate the integral (2) requires the following steps.

1. Sample a set of $N$ points $\{L^{(i)}\}_{i=1}^N$ from the prior distribution $P(L)$ such that the sampled points respect local equality/inequality constraints (i.e. the variables $\{H_i\}_{i=1}^{m-p}$ have the value true).
2. Estimate the integral $I(X)$ using the set $\{L^{(i)}\}_{i=1}^N$ of points as follows:

$$\hat{I}(X) = \frac{1}{N} \sum_{i=1}^{N} P_{O_1}(F_1(X, L^{(i)})|F_1(X, L^{(i)})) \cdot \cdots \cdot P_{O_p}(F_p(X, L^{(i)})|F_p(X, L^{(i)})).$$

Points sampling

The set of $N$ points used to estimate the integral may be sampled in various ways. Because parameters pertaining to different pose vectors are independent, we can decompose the “state vector” $L$ into $m - p$ components $\{L_{ikjk}\}_{k=1}^{m-p}$ and apply a local sampling algorithm [Geweke, 1996, Neal, 1993]. Using a local sampling algorithm, updating the state vector $L$

$$L^{(t)} = (L_{t_1j_1}^{(t)}, L_{t_2j_2}^{(t)}, \cdots, L_{t_{ikjk}}^{(t)}, \cdots, L_{t_{m-p}j_{m-p}}^{(t)}$$

only requires updating one component $L_{ikjk}$.
\[ L^{(t+1)} = (L_{i_1j_1}^{(t)}, L_{i_2j_2}^{(t)}, \ldots, L_{i_kj_k}^{(t+1)}, \ldots, L_{i_m-j_m-p}^{(t)}). \]

\( N \) iterations of this procedure give us the set \( \{L^{(i)}\}_{i=1}^{N} \), which will be used to estimate the integral.

To update a component \( L_{i_kj_k} \) (a set of parameters pertaining to the same pose vector \( Q_{i_kj_k} \)), we must take into account possible dependencies between these parameters. Consequently, we face two problems.

- **Candidate point sampling**
  A candidate \( L_{i_kj_k}^c \) is drawn from the distribution \( P(L_{i_kj_k}) \). Direct sampling methods from simple distributions such as uniform distributions and Gaussians are available. If we do not have a direct sampling method from \( P(L_{i_kj_k}) \) at our disposal, an indirect sampling method must be used. In this work, we chose a Metropolis sampling algorithm [Geweke, 1996, Neal, 1993].

- **Candidate validity checking**
  Suppose that we have a geometric relationship between two geometric entities \( E_i \) and \( E_j \). A geometrical calculus depending on the type of this relationship allows checking of the constraint \( C_k(Q_{i_kj_k}) \leq 0 \). If this constraint is respected (i.e. \( P(H_k|X_{i_kj_k}L_{i_kj_k}) = 1 \)), the candidate \( L_{i_kj_k}^c \) is accepted, otherwise it is rejected. Figure 3 shows a face-on-face relationship example.

![Fig. 3. The candidate point is rejected because it does not respect the face-on-face constraint.](image)

**Optimization of computation time**

Using a local sampling method to update the state vector \( L \) allows a reduction in the computation time of the estimates of integrals. If, for a given point \( L^{(t)} \), we denote the values of functions \( F_i(X) \) by \( F_i^{(t)}(X) \), \( i = 1 \cdots p \), then the values of \( F_i(X) \) in the next step \( F_i^{(t+1)}(X) \) are obtained by partly updating \( F_i^{(t)}(X) \).
4.2 Optimization method: the “Monte Carlo Simultaneous Estimation and Maximization” (MCSEM) algorithm

The optimization method to be chosen for our application must satisfy a set of criteria in relation to the shape and nature of the function to optimize. The method must:

1. be global, because the function to optimize is often multimodal;
2. allow multiprecision computation of the objective function (its estimation with high accuracy may require long computation times); and
3. allow parallel implementation to improve efficiency.

The resolution method used in BCAD to solve the double integration/optimization problem is based on an adaptive genetic algorithm. The accuracy of integral numerical estimation is controlled by the optimization process to reduce computation time.

In the following, we first present the general principles of these algorithms. Then, we discuss the practical problems that we faced when using standard genetic algorithms in our application and give the required improvements.

Principles of genetic algorithms

Genetic Algorithms (GAs) are stochastic optimization techniques inspired by the biological evolution of species. Since their introduction by Holland [Holland, 1975] in the 1970s, these techniques have been used for numerous global optimization problems, thanks to their ease of implementation and their relative independence of application fields. They are widely used in a large variety of domains including artificial intelligence [Grefenstette, 1988] and robotics [Mazer et al., 1998].

Biological and mathematical motivations of genetic algorithms and their principles are not discussed here. We only discuss the practical problems that we face when using standard genetic algorithms in Bayesian inference. We give the required improvements and the corresponding algorithms.

In the following, we will use $G(X)$ to denote the objective function $P(X|H_1 \cdots H_m K_1 \cdots K_p)$.

Narrowness of the objective function – constraint relaxation

In our applications, the objective function $G(X)$ may have a narrow support (the region where the value is not null) for very constrained problems. The initialization of the population with random individuals from the search space may give null values of the function $G(X)$ for most individuals. This will make the evolution of the algorithm very slow, and its behaviour will be similar to random exploration.

To deal with this problem, a concept inspired from classical simulated annealing algorithms consists of introducing a notion of “temperature”. The
principle is first to widen the support of the function by changing the original function to obtain non-null values even for configurations that are not permitted. To do so, we introduce an additional parameter that we call $T$ (for temperature) for the objective function $G(X)$. Our goal is to obtain another function $G^T(X)$ that is smoother and has wider support, with

$$\lim_{T \to 0} G^T(X) = G(X).$$

To widen the support of $G(X)$, all elementary terms (distributions) of $G(X)$ are widened, namely:

- distributions $P_{Oi}(F_i(X,L))$, where $i = 1 \cdots p$;
- inequality constraints $P(\mathcal{H}_{m-p+j}|F_j(X,L))$, where $j = 1 \cdots p$.

Some examples follow.

- For a Gaussian distribution (Fig. 4):
  $$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$
  $$f^T(x) = \frac{1}{\sqrt{2\pi}(1+T)} e^{-\frac{1}{2} \frac{(x-\mu)^2}{(1+T)^2}}.$$

- For an inequality constraint over the interval $[a, b]$ (Fig. 4):
  $$f(x) = \begin{cases} 
  1 & \text{if } a \leq x \leq b \\
  0 & \text{else}
  \end{cases}$$
  $$f^T(x) = \begin{cases} 
  1 & \text{if } a \leq x \leq b \\
  e^{-\frac{(x-a)^2}{2(1+T)}} & \text{if } x < a \\
  e^{-\frac{(x-b)^2}{2(1+T)}} & \text{otherwise}
  \end{cases}$$

**Fig. 4.** A Gaussian distribution and an inequality constraint at different temperature values.

In the general case, inequality constraints may be more complex. Figure 5 shows the case of a *point-on-face* inequality constraint for a square face.
Fig. 5. The distribution corresponding to inequality constraints induced by a point-on-face relationship for a square face at different values of temperature. The left figure shows the original constraints \((T = 0)\), while the middle and the right ones show these constraints relaxed at \((T = 50)\) and \((T = 100)\) respectively.

Accuracy of the estimates – multiprecision computing

The second problem that we must face is that only an approximation \(\hat{G}(X)\) of \(G(X)\) is available, of unknown accuracy. Using a large number of points to obtain sufficient accuracy may be very expensive in computation time, so that use of a large number of points in the whole optimization process is inappropriate.

Because the accuracy of the estimate \(\hat{G}(X)\) of the objective function depends on the number of points \(N\) used for the estimation, we introduce \(N\) as an additional parameter to define a new function \(\hat{G}_N(X)\).

Suppose that we initialize and run for some cycles a genetic algorithm with \(\hat{G}(X)\) as evaluation function. The population of this GA is a good initialization for another GA having \(\hat{G}_N(X)\) as evaluation function, where \(N_2 > N_1\).

General optimization algorithm

In the following, we label the evaluation function (the objective function) by the temperature \(T\) and the number \(N\) of points used for estimation. It will be denoted by \(G^T_N(X)\).

Our optimization algorithm may be described by the following three phases:

1. initialization and initial temperature determination;
2. reduction of temperature to recreate the original objective function; and
3. augmentation of the number of points to increase the accuracy of the estimates.

Initialization

The population of the GA is initialized at random from the search space. To minimize computing time in this initialization phase, we use a small number
$N_0$ of points to estimate integrals. We propose the following algorithm as an automatic initialization procedure for the initial temperature $T_0$, able to adapt to the complexity of the problem. This initialization phase is summarized by the following algorithm and schematized in Fig. 6 for a 1D objective function example.

**INITIALIZATION()**
BEGIN
FOR each population[i] DO
    REPEAT
        population[i] = random($S$)
        value[i] = $E_{T_0}^i$(population[i])
        if (value[i] == 0.0)
            $T = T + \Delta T$
        UNTIL (value[i] > 0.0)
    END FOR
END FOR
Reevaluate_population()
END

where $\Delta T$ is a small increment value.

**Temperature reduction**

To obtain the original objective function ($T = 0.0$), a possible scheduling procedure consists of multiplying the temperature, after running the GA for a given number of cycles $n_{c1}$, by a factor $\alpha$ ($0 < \alpha < 1$). A small value for $\alpha$ may cause the divergence of the algorithm, while a value too close to 1.0 may considerably increase the computation time. In this work, the value of $\alpha$ was fixed experimentally to 0.8. The “temperature reduction” phase is summarized by the following algorithm and schematized in Fig. 7.

**TEMPERATURE_REDUCTION()**
BEGIN
    WHILE ($T > T_\epsilon$) DO
        FOR i=1 TO $n_{c1}$ DO
            Run_GA()
        END FOR
        $T = T * \alpha$
        Reevaluate_population()
    END WHILE
    $T = 0.0$
    Reevaluate_population()
END

where $T_\epsilon$ is a small threshold value.

**Augmenting the number of points**

At the end of the temperature reduction phase, the population may contain several possible solutions for the problem. To decide between these solutions, we must increase the accuracy of the estimates. One approach is to multiply
$N$, after running the GA for a given number of cycles $nc_2$, by a factor $\beta$ ($\beta > 1$) so that the variance of the estimate is divided by $\beta$:

$$Var(G_{\beta N}^0(X)) = \frac{1}{\beta} Var(G_N^0(X)).$$

The “increasing number of points” phase is summarized by the following algorithm and schematized in Fig. 8.

**NUMBER_OF_POINTS_INCREASING()**

BEGIN

WHILE ($N < N_{max}$) DO

FOR $i = 1$ TO $nc_2$ DO

Run_GA()

END FOR

$N = N \times \beta$

Reevaluate_population()

END WHILE

END

where $N_{max}$ is the number of points that allows convergence of the estimates $E_N^0(X)$ for all individuals of the population.

**Fig. 6.** The initialization phase of the MCSEM algorithm. The theoretical distribution to maximize is shown in black, and the estimated one using Monte Carlo numerical integration is shown in grey. From left to right, the $T$ (temperature) parameter is increased starting from zero (i.e. the initial distribution).

BCAD also provides an “anytime” version of the MCSEM algorithm. In this version, the user is allowed to fix the maximum number of evaluations of the objective function or the maximum time to be used to maximize it.

A more generic implementation of the MCSEM algorithm has been successfully integrated in a general-purpose Bayesian inference engine [Mekhnacha et al., 2006].

**5 Overview of the implementation**

In this section, we present an overview of the implementation of the CAD modeller that follows the principles presented above.
5.1 Specification language

A workcell is constructed by evaluating a script file. This script contains a set of instructions used:

• to create geometric entities;
• to create parts; and
• to describe probabilistic constraints between parts.

After evaluation of the script, a graphic model of the cell is constructed.

Geometric entities creation

Geometric entities creation uses a specialized method for each entity. When creating an entity, a frame attached to it is also created automatically. The following methods are used.

• \texttt{New\_Vertex}(x, y)
• \texttt{New\_Edge}(vertex1, vertex2)
• \texttt{New\_Face}(list\_of\_vertices)
• \texttt{New\_Sphere}(centre, radius)
• *New_Cylinder*(centre, radius, direction, length)
• *New_Mesh*(list_of_vertices)

BCAD allows the user to add a distribution on each parameter of a given geometric entity using the following method.

• *Add_Distrib_On_Param*(geom_entity, param_number)

For example, using this method, we can describe a probabilistic mesh by adding a Gaussian distribution to all its vertex positions.

In the same way, BCAD allows the user to add “Shape Degrees Of Freedom”. This functionality extends the “Unknown” status of a given parameter (in the probabilistic kinematic graph) to shape parameters. It uses the following method.

• *Add_Shape_DOF_On_Param*(geom_entity, param_number)

Using this method, we can define the radius of a sphere as a DOF, as in the geometric problem example defined in Subsection 6.3.

**Parts creation**

A part is a set (possibly empty, when only the attached frame is modelled) of geometric entities. This set of entities is given as a parameter when creating the part. An additional graphic object can be added to give a realistic graphic representation. We use the following method.

• *New_Part*(list_of_geom_entities, add_graph_obj)

**Probabilistic kinematic links description**

Creating a probabilistic kinematic link between two frames or two geometric entities uses the following instructions to create the probabilistic kinematic link and to use it to attach entities.

• *New_Link*(status_vector, distribution)
• *Attach*(parent_item, child_item, link)

If *parent_item* and *child_item* are geometric entities (instead of simple frames), the corresponding equality and inequality constraints are automatically added by the system.

**5.2 Graphics system and geometric uncertainties visualization**

The use of graphic support has considerable interest for 3D geometric workcell modelling and for appreciation of the calculated solutions for a given problem. Moreover, it may in our case allow a *visualization* of geometric uncertainties and make their perception easier.
A workcell is constructed by evaluating a script containing a set of instructions, as described above. Besides the construction of the internal representation of the workcell, the evaluation of the script constructs a graphic model corresponding to this workcell and passes it as a parameter to the invoked 3D viewer (see Fig. 9).

In the original version of BCAD, the 3D-visualization system is based on the Quickdraw3D graphic library developed and proposed by Apple for MacOS and Windows 95/98/NT platforms. A more recent version of BCAD uses the OpenGL and QT multiplatform libraries (see Fig. 9).

Geometric uncertainties visualization

Because the relative poses of parts are described as probability distributions instead of single scalar values, we are interested in developing a graphic representation that takes account of this probabilistic aspect of poses on a display screen.

The proposed method is to simulate uncertainties in the poses of parts. The principle is to use a Monte Carlo simulation by sampling the values of the parameters of the poses in the workcell from probability distributions over these parameters. Instead of displaying a part in a fixed pose in the graphic scene, the part is displayed, with a given frequency, in the poses obtained by this sampling. If the frequency of sampling is high enough, this will give a good visual perception of the geometric uncertainties in the model of the workcell.

This visualization of uncertainties allows a more concrete perception of their propagation in a given configuration. In particular, it allows graphic comparison of two different solutions for a given geometric problem.

**Fig. 9.** Screen copies of BCAD: on the left the Quickdraw3D version and on the right the OpenGL/QT one.
6 Examples

This section presents three problem specifications and resolutions using BCAD.

6.1 A kinematics inversion example

In this section, we describe how to use our CAD modeller for concrete problems. We present in detail a kinematics inversion problem under geometric uncertainties.

Problem description

Using two Stäubli Rx90 robot arms with six revolute joints, we are interested in placing two prismatic parts one against the other. The only constraint is that a face of the first part will be in a face-on-face relationship with a face of the second.

The two arms are modelled as a set of parts attached to each other using probabilistic kinematic links. We assume that the more significant uncertainties are on the zero positions of the joints. The two parts are also attached to arms’ end effectors using probabilistic kinematic links. The added constraint that we wish to satisfy to solve the problem is represented by a link between the two faces to place in the face-on-face relationship. For this link, we use three Gaussians on the three constrained parameters $t_z$, $r_x$ and $r_y$ with zeros as mean values and 0.5 mm, 0.01 rad and 0.01 rad respectively as standard deviations. Figure 10 shows the two arms and the corresponding kinematic graph.
We suppose in this example that the zero position uncertainties of the arm on the right of Fig. 10 (Right_Arm) are five times larger than the ones of the arm on the left (Left_Arm) (for each joint, we suppose a Gaussian distribution on the zero position with 0.01 rad as the standard deviation for Left_Arm and with 0.05 rad for Right_Arm). Our aim is to comment qualitatively on the solution obtained and to show the importance of taking uncertainties propagation into account when choosing a solution.

Results
Figure 11 shows the solution obtained by the system. This solution gives maximal precision for the required face-on-face relationship because:

1. the Right_Arm (the less accurate one) is coiled to minimize the propagation of the uncertainties of its zero positions; and
2. rotation axes are perpendicular to the common normal of the two faces.

![Fig. 11. The solution obtained by the system.](image)

Table 1 summarizes the problem complexity and the system performance for this problem using a PowerPC G3/400 machine.

Discussion
This example shows how the proposed method takes geometric uncertainties into account in a general and homogeneous way. No assumptions have been made, either about the uncertainty models (shapes of the distributions used) or about the linearity of the model or the possibility of its being linearized. It also shows how possible redundancy of the system relating to the required task is used to find the most accurate solution.
Table 1. Some parameters summarizing the problem complexity and the system performance for this kinematics inversion problem.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integration space dimension</td>
<td>50</td>
</tr>
<tr>
<td>Optimization space dimension</td>
<td>12</td>
</tr>
<tr>
<td>Number of cycles</td>
<td>1</td>
</tr>
<tr>
<td>Number of frames</td>
<td>28</td>
</tr>
<tr>
<td>Number of inequality constraints</td>
<td>16</td>
</tr>
<tr>
<td>Computation time (seconds)</td>
<td>13</td>
</tr>
</tbody>
</table>

6.2 A calibration example

In this section, we present a calibration problem.

Problem description

![Fig. 12. A parallelepiped pose and dimensions calibration problem using contact relationships (left) and the set of contacts to be used for calibration (right).](image)

The purpose of this example is to calibrate the pose and the size of a 3D part. More precisely, we are interested in identifying the parameters of the pose of a parallelepiped on a table and the three dimensions of this parallelepiped (see Fig. 12).

The experimental protocol is as follows. For each measurement, a six-DOF arm is moved to a configuration that allows obtaining a contact between a touch sensor mounted on the end effector of the arm and a face of the parallelepiped. A set of $N$ contacts between the touch sensor and the faces will give the set of $N$ measurements (configurations that allow contact) that we will use for calibration (see Fig. 12). The geometric model of the arm is the same used for Left_Arm in the previous example.
We suppose that the parallelepiped lies on a table. Consequently, we should identify only the $x$ and $y$ position parameters and the $\alpha$ orientation parameter. For the size of the parallelepiped, we should identify the parameters $sx$, $sy$ and $sz$ representing the distances between each pair of parallel faces. We used for this example a set of 10 contacts. For each face (except for the inferior face, which lies on the table), two measurements were taken. Figure 13 (left) shows the contact points and the corresponding faces that must be joined to solve this calibration problem. It also gives the kinematic graph corresponding to this problem (Fig. 13 (right)).

![Fig. 13. Contact points and the parallelepiped faces that must be joined to solve the calibration problem (left), and the corresponding kinematic graph (right).](image)

**Results**

The *a priori* distribution $P(X)$ on the search space $X = (x, y, \alpha, sx, sy, sz)^T$ expresses our prior knowledge on the parameters to be identified. For this example, we have assumed a uniform distribution $P(X)$ to express the fact that no initial estimation of these parameters is available.

We summarize the problem complexity and the system performance for this problem using a PowerPC G3/400 machine in Table 2.

The simulated contacts were taken at non-null distances between the touch sensor and the parallelepiped faces. Table 3 gives error values for the 10 measurements. We emphasize that all these contact errors have positive values because the touch sensor cannot overlap the parallelepiped.

Table 4 gives simulation values of the parameters to calibrate and the values obtained after calibration.
Table 2. Some parameters summarizing the problem complexity and the system performance for this calibration problem.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integration space dimension</td>
<td>30</td>
</tr>
<tr>
<td>Optimization space dimension</td>
<td>6</td>
</tr>
<tr>
<td>Number of cycles</td>
<td>10</td>
</tr>
<tr>
<td>Number of frames</td>
<td>77</td>
</tr>
<tr>
<td>Number of inequality constraints</td>
<td>40</td>
</tr>
<tr>
<td>Computation time (seconds)</td>
<td>23</td>
</tr>
</tbody>
</table>

Table 3. Error values used when simulating contacts.

<table>
<thead>
<tr>
<th>Contact 1</th>
<th>Contact 2</th>
<th>Contact 3</th>
<th>Contact 4</th>
<th>Contact 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulated errors (mm)</td>
<td>0.677</td>
<td>0.567</td>
<td>0.303</td>
<td>0.792</td>
</tr>
<tr>
<td>Contact 6</td>
<td>Contact 7</td>
<td>Contact 8</td>
<td>Contact 9</td>
<td>Contact 10</td>
</tr>
<tr>
<td>Simulated errors (mm)</td>
<td>0.791</td>
<td>0.883</td>
<td>0.858</td>
<td>0.383</td>
</tr>
</tbody>
</table>

Table 4. Initial values (simulation values) of the parameters to calibrate and calibration results.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$ (mm)</td>
<td>900.000</td>
</tr>
<tr>
<td>$y$ (mm)</td>
<td>-900.000</td>
</tr>
<tr>
<td>$\alpha$ (rad)</td>
<td>0.7854</td>
</tr>
<tr>
<td>$s_x$ (mm)</td>
<td>300.000</td>
</tr>
<tr>
<td>$s_y$ (mm)</td>
<td>300.000</td>
</tr>
<tr>
<td>$s_z$ (mm)</td>
<td>300.000</td>
</tr>
<tr>
<td>Simulation values</td>
<td>900.000</td>
</tr>
<tr>
<td>Calibration results</td>
<td>900.195</td>
</tr>
</tbody>
</table>

Discussion

This example presents an application of our method to parameter identification problems. In particular, we show that using this method allows the following.

- We can take into account prior information on the parameters to estimate.
- We can take into account, for each measurement (contact), its accuracy by propagating the uncertainties of the arm model. This allows an implicit weighting of these measurements (more accurate measurements are given more importance in the calibration process).
- We can take into account prior information on the measurement tool used.

In this particular example, where measurements are contact relationships, we have expressed the non-overlap phenomenon using a non-symmetrical distribution

$$P(t_z) = \begin{cases} 
\frac{2}{\sqrt{2\pi}\sigma_c} e^{-\frac{1}{2} \frac{t_z^2}{\sigma_c^2}} & \text{if } t_z \geq 0 \\
0 & \text{else}
\end{cases}$$

where $\sigma_c$ was 0.5 mm.
6.3 Online positioning for computer-assisted surgery

This section describes an application for online positioning in computer-assisted surgery systems.

**Problem description**

Accurate positioning is a central issue in computer-assisted surgery, especially in total hip replacements for people suffering from severe arthritic conditions. Using such a system aims at avoiding improper alignment, which can result in decreased mobility of the joint and a risk of dislocation.

More precisely, this application aims to locate the rotational centre of the hip joint by using the movements of the femur with a rigid body attached. The surgeon moves the patient’s thigh within the free range of motion of the hip joint and the 3D position of the attached rigid body is tracked using a vision system. The set of 3D position points obtained is then used to estimate the rotational centre of the hip.

The experimental data set used for this application contains 171 3D points and was provided by the Aesculap company, which specializes in hip and knee implants.

The problem is modelled as a set of point-on-sphere constraints. The task is to estimate four parameters: The centre \( c_x, c_y, c_z \) (the rotational centre of the hip), and the radius \( R \) (the femur’s length) of the sphere. We assume Gaussian errors on the 3D points provided by the vision system. We also assume a Gaussian error on the position of the rigid body itself. Figure 14 shows the set of 3D points used and the corresponding kinematic graph.

![Fig. 14. The rotational centre of the hip joint estimation problem and the corresponding kinematic graph.](image-url)
Results

Figure 15 shows the solution obtained by the system.

Table 5 summarizes the problem complexity and the system performance for this problem using a PowerPC G4/1.33 GHz laptop machine.

Table 5. Some parameters summarizing the problem complexity and the system performance for this problem.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integration space dimension</td>
<td>3</td>
</tr>
<tr>
<td>Optimization space dimension</td>
<td>4</td>
</tr>
<tr>
<td>Number of cycles</td>
<td>171</td>
</tr>
<tr>
<td>Number of frames</td>
<td>173</td>
</tr>
<tr>
<td>Number of inequality constraints</td>
<td>0</td>
</tr>
<tr>
<td>Computation time (seconds)</td>
<td>15</td>
</tr>
</tbody>
</table>

Discussion

This example shows how to use BCAD to model and solve a positioning problem expressed as a set of probabilistic point-on-sphere constraints. In particular, it demonstrates the applicability of the proposed resolution algorithm when using a large number of measurements (constraints).

The measurement points are not simulated but provided by a real tracking system, making it difficult to evaluate objectively the quality of the calculated estimate.

For the data set shown, the prior probability distribution on the four estimated parameters was assumed uniform. However for poor quality data sets\(^3\),

\(^3\) Poor quality data sets are those containing points concentrated in a small 3D region. Larger ranges of hip joint motion made by the surgeon allow better quality of the estimate.
using a non-uniform prior distribution (on some parameters) may greatly improve the quality of the estimate by guiding it to more probable regions and avoiding local minima.

For example, giving a Gaussian prior (even flat) around the statistical mean of human femur lengths (sphere radius parameter) may result in an important improvement of both the radius and the centre estimates.

7 Conclusion and future research

We have presented a generic approach to geometric problem specification and resolution using a Bayesian framework. We have shown how a given problem is first represented as a probabilistic kinematic graph and then expressed as an integration/optimization problem. The MCSEM algorithm has been proposed as an appropriate numerical technique for applying this methodology. For generality, no assumptions have been made on the shapes of distributions or on the amplitudes of uncertainties.

Numerous geometric problems have been specified and solved using our BCAD system. Three examples have been presented in this chapter.

Experimental results from our system have demonstrated the effectiveness, the robustness, the expressiveness, and the homogeneity of representation of our approach. However, additional studies are required to improve the expressiveness of BCAD and the performance of its resolution module.

To improve expressiveness, future work will aim at allowing the use of high-level sensors such as vision-based systems. It will also aim at integrating projective geometry functions to allow definition of probabilistic constraints in the image plane. We are also considering extending our system to take collision detection into account by automatically generating additional probabilistic constraints corresponding to potential collisions.

For the resolution module, we think that numerical integration can be avoided in special cases in which the integrand is a product of generalized normals (Dirac delta functions and Gaussians) and when the model is linear or can be linearized with small enough error). We also think that the optimization algorithm may be improved by using a local derivative-based method after the convergence of our genetic algorithm.

References


J. C. Owen. Constraints on simple geometry in two and three dimensions. 


3D human hip volume reconstruction with incomplete multimodal medical images

Application to computer-assisted surgery for total hip replacement (THR)

Miriam Amavizca¹, Christian Laugier¹, Emmanuel Mazer², Juan Manuel Ahuactzin², and Francois Leitner³

¹ French National Institute for Research in Computer Science and Control (INRIA Rhone-Alpes)
² ProBayes S.A.S.
³ AESCULAP B-BRAUN

1 Introduction

This work is within the context of Computer-assisted Orthopaedic Surgery (CAOS), in particular Total Hip Replacement (THR) surgery.

“Computer-assisted Surgery (CAS) has the aim of assisting surgeons in their therapeutic efforts to be as exact and minimally invasive as possible” [Corbillon, 2002]. CAS is an interdisciplinary research area; it uses many sources of information, devices, computer techniques and clinics. In the past, medical imagery was only used for diagnosis and pathology localization. Today, image processing and computer-assisted surgery systems help surgeons to improve their perception and action capabilities. “Medical image processing makes possible the acquisition of a numerical model of reality. In surgery, this corresponds to a replica of the patient’s anatomy” [Corbillon, 2002].

In conventional medical imagery, the patient’s anatomy is obtained from multiple two-dimensional (2D) medical images. In contrast to conventional medical imagery, image processing allows a three-dimensional (3D) visualization of the patient from a numerical model. The 3D representation adds visibility to the organs and makes them more perceptible. Surgical systems intervene to obtain the 3D representation of the organs and to use the model in the different stages of the patient’s medical treatment: preoperative (planning), operative (control) and post-operative (validation). In this way, the aim of CAS is to obtain more precise post-operative results according to the preoperative planning than can be achieved with conventional techniques [Corbillon, 2002].

CAS has applications in orthopaedic surgery and surgery of deformable tissues (such as vascular and heart surgery). The improvement of surgical tech-
niques in orthopaedics is particularly important in prosthesis implantation: an error in the component’s placement could lead to dislocation or abnormal wear and hence to prosthesis failure. In contrast with deformable tissues, bones can be studied by non-invasive imagery, grasped by manual tools and manipulated without significant deformation.

In THR, a prosthesis is used for the treatment of hip or femur diseases or trauma. It is used when the surface of the articular joint of the hip and femur is damaged with the consequence that mobility is inadequate. THR replaces the damaged joint by two artificial articular surfaces, the cotyloidean prosthesis in the pelvis and the stem in the femur. In the conventional process, surgeons follow three steps.

- **The preoperative step** allows the selection of the best prosthesis type for the patient and the planning of the optimal position with radiography. “Optimal position” means the best position for restoring the patient’s mobility.
- In the **operative step**, the surgeon prepares the bones for the implantation, by reaming the cotyloidean surface, resecting the femoral head and reaming the intramedullary cavity of the femur. The prostheses are then implanted. These processes are of course according to the planned position from the preoperative step.
- In the **post-operative step**, the surgeon determines the surgery quality by evaluating the final position of the prosthesis components and other measures.

A THR surgery system supports preoperative planning and operative assistance in the prosthesis implantation. All this is achieved while taking into account the patient’s morphology. Most such systems require magnetic resonance images (MRI), computer tomography images (CT) or scanner images to reconstruct the 3D volumes of the hip and femur.

The procedural steps of a computer-assisted surgery system are different from those of the conventional process.

- In the preoperative step, medical images of the organ are acquired to construct a 3D model of the structures. The surgery is planned using software that helps the surgeon to select the best position and type of the prosthesis using the 3D model. The position of the prosthesis is evaluated by a collision checker taking into account the different hip movements. The 3D model allows the surgeons to practice using a simulation of the surgical process and to define the trajectories of the surgical tools.
- During surgery, the surgical systems facilitate and improve the prosthesis implantation. At the beginning of the surgery, the surgeons collect some points from the hip to determine the real position of the patient on the surgical table, matching them with the preoperative 3D model of the patient. The aim is to minimize the distance between the operative points and the preoperative model. The system then guides the surgeon to place the prosthesis in the planned position.
- Finally, in the post-operative step, the system evaluates the final position of the prosthesis compared with the planned position.
THR surgery using 3D imagery systems has several advantages over the conventional technique using 2D imagery. These systems help in diagnosis and treatment. A 3D volume of the patient’s hip obtained from MRI/CT/scanner medical images allows the surgeon to analyse, select and validate the best prosthesis position for the patient. During surgery, the systems allow the surgeon to improve the surgical tool’s position in 3D, the surgical movements and the accuracy of the prosthesis positioning.

1.1 Medical problems and difficulties

In spite of the advantages of computer-assisted surgery systems using 3D imagery, this technique is not very developed. Today’s surgery systems have difficulties linked to the accessibility of medical images.

- Medical imaging takes too much time.
- MRI/CT/scanner imaging machines are not readily accessible in all clinics.
- It is hard to obtain an automatic image segmentation for the 3D reconstruction.
- Some patients cannot be exposed to MRI/CT/scanner imaging.

Thus, most surgeons still use the conventional techniques. The problems with the 2D conventional technique are as follows.

- The preoperative planning and the operative guidance in THR’s are made using anteroposterior hip radiography of the patient.
- The surgery is performed in a 3D space while the surgeon must implant the prosthesis according to a 2D-planned position.
- The surgeon must consider the patient’s movements while implanting the prosthesis during the surgery, and must be guided by experience and the surgical tools furnished by the prosthesis fabricators.
- The surgical tools furnished by the prosthesis fabricators use landmarks that will match the positional axes of the patient (anteroposterior, medial, distal), but they do not take the patient’s movements into account.

The conventional 2D technique can lead to bad prosthesis positioning. Hassan [Hassan, 1998] remarks that "more than 42% of THRs performed with the conventional tools are out of the “safe position” of Lewinnek. This includes operations by experienced surgeons" [Lewinnek, 1978]. Some of these problems have been solved by systems like those of Mollard [Mollard and Leitner, 2002] and Kiefer [Kiefer, 2003]. These systems do not use MRI/CT/scanner images but use the Orthopilot® system for tracking the surgical tools and the patient’s hip. The system guides the surgeon during the implantation of the prosthesis. Nevertheless, there are problems with (i) the planning, (ii) the prosthesis position validation (made using 2D data), and (iii) the impossibility of treating deformed hips.
1.2 The scientific problem

We propose a methodology for obtaining a 3D patient’s hip model from 2D data and partial 3D data. The 2D data are obtained from one anteroposterior radiographic image, and the partial 3D data are obtained from echographic images. All procedures are non-invasive, low cost and fast. The patient’s personalized model is obtained by a “deformation” of a generic hip mesh. Control points are obtained from radiographic and echographic data (Fig. 1). They correspond to a selection of characteristic hip points defined in Section 2.1 and described in Section 2.3.

This work’s contribution is a methodology for obtaining a 3D hip volume with radiographic data and partial echographic data.

The proposed method for obtaining the 3D hip model from 2D data and partial 3D data has three steps (Fig. 2):

- (i) radiographic (2D) data acquisition and echographic (3D) data acquisition corresponding to characteristic points;
- (ii) 3D radiographic data inferred from a probabilistic atlas and a Bayesian model; and
- (iii) 3D model construction by deforming a generic hip mesh.

These steps impose some challenges linked to the lack of 3D radiographic data and to the difficulty of acquiring the echographic data. To solve the problem of 3D radiographic data acquisition from the 2D data, we use morphing and Bayesian techniques. Bayesian techniques are used to obtain a probabilistic atlas of the hip proportions, that is, the probability distribution of the relations between the hip’s dimensions.
1.3 Organization of this chapter

In this chapter, we present a methodology for obtaining a 3D hip model using a patient’s radiographic and echographic data. First, we present a selection of hip characteristic points to be used in the construction of a 3D model; these characteristic points were obtained from our research in the literature. Second, we present our methodology for obtaining the 3D human hip model using a probabilistic atlas and a Bayesian model. This process is divided into three steps: (i) the acquisition of patient information, (ii) the construction of a probabilistic atlas and a probabilistic geometric model, and (iii) a mesh deformation method. Finally, we present the experimentation, results and conclusions.

The context of this work, an analysis of related work and current THR surgery systems are described in Amavizca-Ruiz [2005].

2 Reconstruction of a 3D human hip model using a probabilistic atlas

2.1 Towards a generic hip model: the probabilistic atlas

In this section, we describe the more generic characteristics of the hip: points, distances and angles. Most of these characteristics are used for classification in anthropometry to study the dimensions of the human body. This research allowed us to obtain data that could be incorporated into the hip generic model.
2.2 Particular hip characteristics

Hip and femur characteristics have been studied by several authors. These characteristics allow us to determine, for example, the sex, race and age of the subject to whom they belong. The classification is made by using indexes and dimensions. Detailed descriptions (including indexes and authors) of the hip and femur characteristics that we found to be the most used in anthropometry are listed in Amavizca-Ruiz [2005].

The next section presents the results of this research.

Discussion

Our research in anthropology showed us that the construction of a 3D hip generic volume is an open problem. Specialists do not have enough information to specify values or indexes to classify the hip by sex or race. A classification by sex independent of race is possible only using some morphological features. Actual classification values generally belong to a specific population; furthermore, in some cases, these values are obtained from different indexes.

This problem also arises in femur sex classification. However, the femur does not vary as much as the hip. That is why Viceconti [Viceconti, 1996a] proposed a standardized femur model.

The more generic values of the hip are represented by a set of points with characteristic average values [Amavizca-Ruiz, 2005]. However, there are no statistics about the relations between the different characteristic points that would tell us, for example, the relation of the hip’s length to its width.

This lack of generic classification led us to select potentially exploitable hip characteristic points, distances and angles with the aim of obtaining a generic hip model for the construction of a 3D hip model. These characteristics are described in the following section and they will be used for the construction of a hip probabilistic atlas.

2.3 The characteristic hip points

The characteristic hip points that were selected from the hip’s anthropometric bibliography are presented below. These lists show the point indexes and their descriptions. The point index $P_{n=1,...,70}$ is presented as pairs because of the hip’s symmetry. We have two points corresponding to the same description for each side of the hip (left and right). For example, points $P_1$ and $P_{29}$ correspond to the superior iliac crests of the hip. $P_1$ is over the right iliac crest and $P_{29}$ is over the left iliac crest.

Points were selected according to the following rules.

- The points are characteristic and easy to recognize in the images.
- The points are obtained from anteroposterior radiography, an echography or by external palpation.
The set of characteristic hip points in an anterior view (identifiable from radiography) (Fig. 3a) and in a profile view (identifiable by echography) (Fig. 3b) are as follows.

### Anterior view

<table>
<thead>
<tr>
<th>Points</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$, $P_{30}$</td>
<td>Superior point over the iliac crest</td>
</tr>
<tr>
<td>$P_2$, $P_{31}$</td>
<td>Most lateral point over the iliac crest</td>
</tr>
<tr>
<td>$P_3$, $P_{32}$</td>
<td>Anterior superior spine</td>
</tr>
<tr>
<td>$P_4$, $P_{33}$</td>
<td>Anterior inferior spine</td>
</tr>
<tr>
<td>$P_5$, $P_{34}$</td>
<td>Superior point over the acetabular border</td>
</tr>
<tr>
<td>$P_6$, $P_{35}$</td>
<td>Inferior point over the acetabular border</td>
</tr>
<tr>
<td>$P_7$, $P_{36}$</td>
<td>Inferior point of the ischial notch</td>
</tr>
<tr>
<td>$P_8$, $P_{37}$</td>
<td>Superior point over the articular surface</td>
</tr>
<tr>
<td>$P_9$, $P_{38}$</td>
<td>Inferior point over the articular surface</td>
</tr>
<tr>
<td>$P_{10}$, $P_{39}$</td>
<td>Point over the curved line to obtain the pelvic interior diameter</td>
</tr>
<tr>
<td>$P_{11}$, $P_{40}$</td>
<td>Acetabular fossa</td>
</tr>
<tr>
<td>$P_{12}$, $P_{41}$</td>
<td>Acetabular drop</td>
</tr>
<tr>
<td>$P_{13}$, $P_{42}$</td>
<td>Medial point over the border of the obturator foramen</td>
</tr>
<tr>
<td>$P_{14}$, $P_{43}$</td>
<td>Superior point in the border of the ischial tuberosity</td>
</tr>
<tr>
<td>$P_{15}$, $P_{44}$</td>
<td>Superior point of the symphysisal surface</td>
</tr>
<tr>
<td>$P_{16}$, $P_{45}$</td>
<td>Inferior point of the symphysisal surface</td>
</tr>
<tr>
<td>$P_{17}$, $P_{46}$</td>
<td>Medial point over the border of the obturator foramen</td>
</tr>
<tr>
<td>$P_{18}$, $P_{47}$</td>
<td>Posterior and superior iliac spine</td>
</tr>
<tr>
<td>$P_{19}$, $P_{48}$</td>
<td>Posterior and inferior iliac spine</td>
</tr>
<tr>
<td>$P_{20}$, $P_{49}$</td>
<td>Big sciatic notch</td>
</tr>
<tr>
<td>$P_{21}$, $P_{50}$</td>
<td>Ischial spine</td>
</tr>
<tr>
<td>$P_{22}$, $P_{51}$</td>
<td>Inferior point of the greater sciatic notch</td>
</tr>
<tr>
<td>$P_{23}$, $P_{52}$</td>
<td>Superior point of the acetabular fossa</td>
</tr>
<tr>
<td>$P_{24}$, $P_{53}$</td>
<td>Superior point of the acetabulum</td>
</tr>
<tr>
<td>$P_{25}$, $P_{54}$</td>
<td>Inferior point of the acetabular fossa</td>
</tr>
<tr>
<td>$P_{26}$, $P_{55}$</td>
<td>Inferior point over the acetabular border</td>
</tr>
<tr>
<td>$P_{27}$, $P_{56}$</td>
<td>Point over the iliopubic notch</td>
</tr>
<tr>
<td>$P_{28}$, $P_{57}$</td>
<td>Point over the posterior fibres of the gluteus minimus over the line between the anterior superior spine and the posterior superior iliac spine</td>
</tr>
<tr>
<td>$P_{29}$, $P_{58}$</td>
<td>Inferior point in the border of obturator foramen</td>
</tr>
<tr>
<td>$P_{30}$, $P_{59}$</td>
<td>Point over the posterior fibres of the gluteus medius</td>
</tr>
</tbody>
</table>

### Profile view

<table>
<thead>
<tr>
<th>Points</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_{31}$, $P_{40}$</td>
<td>Posterior and superior iliac spine</td>
</tr>
<tr>
<td>$P_{32}$, $P_{41}$</td>
<td>Posterior and inferior iliac spine</td>
</tr>
<tr>
<td>$P_{33}$, $P_{42}$</td>
<td>Big sciatic notch</td>
</tr>
<tr>
<td>$P_{34}$, $P_{43}$</td>
<td>Ischial spine</td>
</tr>
<tr>
<td>$P_{35}$, $P_{44}$</td>
<td>Inferior point of the greater sciatic notch</td>
</tr>
<tr>
<td>$P_{36}$, $P_{45}$</td>
<td>Superior point of the acetabular fossa</td>
</tr>
<tr>
<td>$P_{37}$, $P_{46}$</td>
<td>Superior point of the acetabulum</td>
</tr>
<tr>
<td>$P_{38}$, $P_{47}$</td>
<td>Inferior point of the acetabular fossa</td>
</tr>
<tr>
<td>$P_{39}$, $P_{48}$</td>
<td>Inferior point over the acetabular border</td>
</tr>
<tr>
<td>$P_{40}$, $P_{49}$</td>
<td>Posterior point over the acetabular border</td>
</tr>
<tr>
<td>$P_{41}$, $P_{50}$</td>
<td>Superior point in the border of the ischial tuberosity</td>
</tr>
<tr>
<td>$P_{42}$, $P_{51}$</td>
<td>Posterior and inferior iliac spine</td>
</tr>
<tr>
<td>$P_{43}$, $P_{52}$</td>
<td>Inferior point over the acetabular border</td>
</tr>
<tr>
<td>$P_{44}$, $P_{53}$</td>
<td>Point over the iliopubic notch</td>
</tr>
<tr>
<td>$P_{45}$, $P_{54}$</td>
<td>Point over the posterior fibres of the gluteus minimus over the line between the anterior superior spine and the posterior superior iliac spine</td>
</tr>
<tr>
<td>$P_{46}$, $P_{55}$</td>
<td>Inferior point in the border of obturator foramen</td>
</tr>
<tr>
<td>$P_{47}$, $P_{56}$</td>
<td>Point over the posterior fibres of the gluteus medius</td>
</tr>
<tr>
<td>$P_{48}$, $P_{57}$</td>
<td>Point over the line between the most lateral point of the iliac crest and the posterior superior iliac spine and the posterior origin fibres of the gluteus medius</td>
</tr>
<tr>
<td>$P_{49}$, $P_{58}$</td>
<td>Posterior point over the acetabular border</td>
</tr>
</tbody>
</table>

### 2.4 Characteristic distances and angles

Distances are obtained from the characteristic points (Fig. 3c), and the selected characteristic angles are shown in (Figs. 3b and d).

### 2.5 Towards a Bayesian model of the hip

The topological description of the hip is based on three main types of characteristics: (i) points, (ii) distances and (iii) angles. However, it is necessary to remark that the hip centre, the distances and the angles can be calculated using the characteristic points of Section 2.3. We propose then to represent the hip topology as a set containing these points.

**Definition 1.** The hip atlas of a patient, which we call the atlas, is defined as the set of values of the characteristic points of the hip atlas $\{p_i = (x_i, y_i, z_i) \mid i = 1, 2, \ldots, 70\}$. 


Fig. 3. Hip characteristic points obtained from radiographic (anterior view) (a) and ecographic points (profile views) (b). To reduce the information in the image, we have represented the points by their numbers. For example, point $P_1$ is represented by the number 1 in the image. Characteristic distances (c) and angles (d) obtained from radiographic points.

By convention, the atlas points of a patient are defined in the coordinate system of the parallelepiped enclosing the hip with its origin at the parallelepiped’s centre. Note that in this way, the atlas centre is well defined regardless of the dimensions of the patient (See Fig. 9b). We will show in Section 4.4 that the atlas centre is obtained using the pairs of points $(P_3, P_{35})$, $(P_1, P_7)$ and $(P_2, P_{30})$. Certainly, these points allow us to calculate the parallelepiped dimensions and then the centre.

In this section, we show that there is no hip generic model and that the measured data has a certain variability. In other words, there are an infinite
number of possible values for \textit{atlas}. Nevertheless, the possible values are limited. This limit will be given, for example, by those people having the biggest and smallest hips in the world. In addition, some values are more frequent than others are; that is, it is possible to measure the value \textit{atlas} quantitatively. Using probabilities, we can express the uncertainty of \textit{atlas} in quantitative terms by the \textit{atlas} event probability, denoted \( P(\text{atlas}) \). Thus, there is a random variable \textit{Atlas} that represents all possible values of the characteristic points and its probability distribution \( P(\text{Atlas}) \). Then, \( P(\text{atlas}) = P(\text{Atlas} = \text{atlas}) \) represents the probability of the characteristic point values denoted by \textit{atlas}.

We propose the construction of a \textit{probabilistic atlas} of the hip characteristic points, or in other words, the construction of \( P(\text{Atlas}) \). Then, the hip will be seen as a structure connecting the characteristic points described by the probability distribution.

\textbf{Definition 2.} The probabilistic atlas is defined as the hip generic model represented by a probability distribution \( P(\text{Atlas}) \). This distribution characterizes the dimensional uncertainty of the hip characteristic points. More particularly, we define the probabilistic atlas as a model based on a multivariable normal distribution statistical analysis [Jaynes, 2003].

\[
P(\text{Atlas}) = \text{Multivariate Gaussian Distribution}
\]

Thanks to the \textit{probabilistic atlas} and the \textit{Bayesian model} that describe the estimation of the patient’s characteristic points (using radiography and some echographic images), it is possible to estimate the patient atlas. When the patient atlas has been estimated, it is possible to construct an estimation of the 3D hip model. In Section 4, we present a general description of this procedure.

3 Bayesian modelling

The concept and basis for probabilistic modelling, the concept of a Bayesian program, and the identification and modelling of a probabilistic geometric model have been presented in Chapters 4 and III. In the next section, we present the process for 3D hip reconstruction using the probabilistic approach.

4 The 3D reconstruction process

4.1 Main steps of the 3D reconstruction

The patient’s 3D hip reconstruction is made from the hip probabilistic atlas and the partial knowledge of a subset of the atlas points of the patient’s hip. This partial data is obtained from two different sources: radiography and
Certainly, the knowledge of the patient’s characteristic points from these two sources is partial, because an X-ray is two-dimensional information, and echography allows us to obtain only some characteristic points. Additionally, the recognition of characteristic points by echography is a very difficult task that even an experienced surgeon can perform with a variable degree of uncertainty. There are therefore unknown data and uncertain data. This is why we have used a Bayesian model that includes the patient’s data and the probabilistic atlas data: this model will allow us to obtain the unknown data, that is, to infer the patient’s atlas. Finally, the deformation of an initial mesh according to the preceding estimation will allow us to obtain a personalized 3D model. To do this, we apply the following five steps (Fig. 2).

- Echographic (3D) data acquisition
- Radiographic (2D) data acquisition
- Data matching
- Patient’s atlas creation by the use of the probabilistic atlas and the Bayesian model of the data
- Mesh deformation respecting the data and morphologic constraints

These steps are briefly described in the following sections.

4.2 Acquisition of echographic (3D) information

Three-dimensional information is obtained from the patient’s echographic images containing the characteristic points. The echographic images are obtained with an Orthopilot® system that locates infrared diodes to position the images according to the hip’s position. The position of the echographic images is represented by a homogeneous matrix in the patient’s rigid body coordinate system. The method for obtaining the echographic data is described in Section 5.1.

4.3 Acquisition of radiographic (2D) information

The radiographic information is obtained from anteroposterior radiography of the patient’s hip. The aim is to recognize and select the hip characteristic points in the image (see Fig. 3a). This information takes into account the patient’s position (when the radiographic image was taken) represented by three rotation angles: frontal, transversal and sagittal. The acquisition method for obtaining the radiographic data is described in Section 5.1.

4.4 Data matching

The patient data is in several coordinate systems. Consequently, it is necessary to perform matching according to the patient’s atlas centre, which corresponds to the centre of the parallelepiped enclosing the hip. We describe this process in Section 5.5.
4.5 Creation of the patient’s probabilistic atlas from the Bayesian model

The patient’s atlas is obtained with a Bayesian model that integrates the patient’s information and the probabilistic atlas. The probabilistic atlas is a probability distribution that captures the uncertainty of the proportions between the hip characteristic points and represents the hip’s generic and geometric model. We constructed our probabilistic atlas from the data for 52 hips, which was furnished by Kepple [Kepple, 1998]. The Bayesian system is described in Section 5.

4.6 Deformation method

In the final step, the initial hip mesh is deformed according to the patient’s inferred atlas. The deformed mesh then represents the actual 3D model of the patient. The deformation algorithm that we use is a derivation of Kshirsagar’s algorithm [Kshirsagar, 2000]. This deformation is made using the “control points” of the mesh, which are displaced to satisfy the patient’s atlas characteristics and the geometric characteristics of the human hip.

In the following section, we present in detail the patient’s 3D hip model reconstruction steps.

5 The Bayesian model of the system

5.1 Introduction

In our system, we want to obtain the patient’s atlas from radiographic (2D) and echographic (3D) data. This incomplete and uncertain data must allow us to infer the patient’s atlas by inferring the more probable values from the Bayesian model. To solve this problem, the system is represented as a probabilistic kinematic model [Mekhnacha, 1999]. The graph of the kinematic model allows us to describe the relations associated with the patient’s data (see Fig. 4) and then to identify the relations between the data and the uncertainties.

An echographic image of the hip contains a point $P_i$ of the atlas. The 3D localization of the image is made with a set of diodes placed in a rigid body. With this mechanism, each echographic image can be positioned in the space (e.g. by the determination of the corresponding geometric transformation). If we suppose that there is no uncertainty in the echographic localization and the point localizations, the points can be represented in the coordinate system of the “rigid body”. The echographic points, which are in their coordinate system $E_i$, are represented in the coordinate system of the “rigid body” by the transformation $T_{E_i}^{RB}$,

$$P_i^{RB} = T_{E_i}^{RB} P_i^{E_i}$$
Fig. 4. A representation of the different coordinate systems of the data. The radiographic image \( R \), an echographic image \( E \), the “rigid body” \( RB \) and the atlas centre \( C \).

On the other hand, a point \( P_C^i \), with respect to the atlas centre, can be represented in the coordinate system of the rigid body by:

\[
P_R^{RB,i} = T_{RB}^{-1} P_C^i.
\]

The matrix \( T_{RB}^{RF} \) represents the echographic position according to the rigid body on the patient. The matrix value \( T_{RB}^{CF} \) can be calculated from a subset of points of the patient’s atlas. The calculation of this transformation is described in Section 5.5.

Similarly to the echographic images, a radiographic image contains a set of atlas point projections, which are identified by the surgeon on the radiographic image. Certainly, the projected points have only two components because the radiographic image is two-dimensional. A 3D point \( P_i \) of the atlas is therefore represented by a 2D point in the radiographic image \( P_{yz}^R \). The notation “\( yz \)” of \( P_{yz}^R \) is because the radiography plane coincides with the \( yz \) plane of the patient’s atlas coordinate system. Certainly, \( P_{yz}^R \) takes into account the patient’s position when the image was taken, so \( P_{yz}^R = projection_{yz}(x_i^R, y_i^R, z_i^R) \), with \([x_i^R, y_i^R, z_i^R]^T = T_C^R P_C^i \), where \( T_C^R \) is the transformation of the atlas centre frame to the radiography frame. This position is represented by rotation angles in the frontal, transversal and sagittal planes of the patient.

At this point, we identify the following geometric objects (nodes) and their relations (arcs or variables).

The geometric objects are:

- \( C \) the centre of the patient’s atlas coordinate system;
- \( RB \) the rigid body coordinate system;
- \( R \) the radiographic coordinate system; and
- \( \{P_1, P_2, ..., P_n\} \), the patient’s atlas points.

Their relations are:
Fig. 5. A view of some echographic images $E_3, E_4, E_{16}, E_{30}$ and $E_{31}$ (a). One of these images is shown in (b) and is searched to find the characteristic point $P_{31}$. For each echographic image, the estimation $\overline{P}_{E_i}$ (white point) of the characteristic point $P_i$ is manually selected by the surgeon.

- $T_C^R$, the transformation matrix from the atlas centre coordinate system to the radiographic coordinate system;
- $T_{E_i}^R$, the transformation matrix from the echographic $i$ coordinate system to the rigid body coordinate system;
- $T_C^{RB}$, the transformation matrix from the atlas centre coordinate system to the rigid body coordinate system;
- $P_C^i$, the atlas point $i$ of the patient represented in the atlas centre coordinate system;
- $P_R^{E_i}$, the patient’s atlas point $i$ projected in the radiographic;
- $P_{E_i}^i$, the patient’s atlas point $i$ represented in the coordinate system of the echography $E_i$; and
- $P_{RB}^i$, the patient’s atlas point $i$ represented in the coordinate system of the rigid body.

As we mentioned before, the acquisition of the echographic images (Fig. 5) is made with a system that localizes infrared diodes in 3D and allows the positioning of the echographic sensor in the space. This means that the system furnishes an estimate $\overline{T}_{E_i}^R$ of the matrix $T_{E_i}^{RB}$. Notice that it is not possible to obtain the real matrix $T_{E_i}^{RB}$ because of several factors linked to the system, especially the patient’s movements. Certainly, the rigid body is fixed to the patient with a waistband, generating uncertainty in the positioning of each acquisition. Similarly, the point $P_{E_i}^i$ can only be obtained with uncertainty because the point is manually selected by the surgeon in the echographic image (see Fig. 5). The surgeon therefore obtains an approximation of $P_{E_i}^i$ noted $\overline{P}_{E_i}^i$.

In the following sections, we define the probabilistic kinematic graph, the geometric problem and the Bayesian program of our system.
5.2 The probabilistic kinematic graph

The probabilistic kinematic graph is composed from the set
\[ S = \{ C, RB, R, P_1, ..., P_n, E_1, E_2, ... E_m \}, \]
which represents the different coordinate systems, and the atlas points.

The arc set is:
\[ A = \{ T^{RB}_C, T^{RB}_{E_1}, ..., T^{RB}_{E_n}, T^R_C, P^C_1, ..., P^C_n, P^R_1, ..., P^R_k, P^E_1, ..., P^E_m \}. \]

The probabilistic kinematic graph is presented in Fig. 6.

![Fig. 6. The probabilistic kinematic graph representing the different geometric entities of the system (nodes) and the relations between them (arcs). The nodes that correspond to a coordinate system are labelled with a square, and the nodes that correspond to a point are labelled with a circle. The arc between two coordinate systems represents a transformation matrix, and the arc between a point and a coordinate system represents the point’s position according to the coordinate system.](image)

5.3 The probabilistic geometric problem

In this section, we identify the probabilistic geometric problem of our system. This allows us to formulate the Bayesian program.

To make the exposition simpler, we define different sets of the atlas points, a set of transformation matrices and two sets of errors.

- \( I_{Atlas} \subset \{ 1, 2, ..., 70 \} \) The set of indexes of the atlas.
- \( I_{E'} \subset I_{Atlas} \) The set of indexes of the atlas points that are necessary to calculate the atlas centre.
- \( I_E \subset I_{Atlas} \) t.q. \( I_{E'} \subset I_E \) The indexes of the echographic points, which include the indexes of the points that are necessary to calculate the atlas centre.
- \( I_R \subset I_{Atlas} \) The indexes of the radiographic points.
We define the following quantities.
\( \text{Atlas}^C = \{ P^C_i \mid i \in I_{\text{Atlas}} \} \) The set of atlas points in the coordinate system of the atlas centre.
\( \text{Atlas}_R = \{ P^R_y \mid i \in I_R \} \) The set of radiographic points in the radiographic coordinate system \( R \).
\( \text{Atlas}_E = \{ P^E_i \mid i \in I_E \} \) The set of echographic points in the echographic coordinate system \( E_i \).
\( \text{Atlas}_{E}^E = \{ P^E_i \mid i \in I_E \} \) The set of estimated echographic points in the echographic coordinate system \( E_i \).
\( \text{Atlas}_{RB}^E = \{ P_{RB}^E \mid i \in I_E \} \) The set of echographic points in the rigid body coordinate system \( RB \).
\( \mathbf{T}_{RB}^E = \{ \mathbf{T}_{RB}^E \mid i \in I_E \} \) The set of transformation matrices from the echographic coordinate system \( E_i \) to the rigid body \( RB \) coordinate system.
\( e_R = \{ \varepsilon_R \mid i \in I_R \} \) The error between the radiographic points and the atlas points. This value corresponds to the equality constraint of the closing cycle generated by the atlas points and the projected points in the radiographic image.
\( e_E = \{ \varepsilon_E \mid i \in I_E \} \) The error between the echographic points and the atlas points. This value corresponds to the equality constraint in closing cycles generated by the atlas points and the echographic points.

The probabilistic geometric problem consists of finding the following distribution.
\[
P(\text{Atlas}^C \mid \mathbf{T}_{RB}^E, \text{Atlas}_E, \text{Atlas}_R, e_R, e_E)
\]

We therefore define the status of the variables as:
Unknown \( \text{Atlas}^C \)
Fixed \( \mathbf{T}_{RB}^E, \text{Atlas}_E, \text{Atlas}_R, e_R, e_E \)
Free \( \mathbf{T}_C, \mathbf{T}_{RB}^E, \text{Atlas}_E, \mathbf{T}_{RB}^R, \text{Atlas}_E, \mathbf{T}_C, \text{Atlas}_R, e_E \)

where \( \text{Atlas}^C \) is the unknown of the problem; \( \mathbf{T}_{RB}^E, \text{Atlas}_E, \text{Atlas}_R, e_R, e_E \) are the known parameters with constraints \((e_R = 0)\) and \((e_E = 0)\); and \( \text{Atlas}_{RB}^R, \mathbf{T}_{RB}^R, \text{Atlas}_E, \mathbf{T}_{RB}^E, \mathbf{T}_C \) are the unknown parameters that will take values that follow a probability distribution.

5.4 The Bayesian program of the system

To simplify the definition of the Bayesian program, we define two subsets of \( \text{Atlas}^C \).
The set of radiographic points in the coordinate system of the atlas centre \(C\):

\[
\text{Atlas}_R^C = \{P^R_i | i \in I_R\}
\]

The set of echographic points in the coordinate system of the atlas centre \(C\):

\[
\text{Atlas}_E^C = \{P^C_i | i \in I_E\}
\]

The resulting Bayesian program is shown in Fig. 7, and its graphic representation is shown in Fig. 8.

Thus, to answer the question of our Bayesian program, we must first find the distribution of the probabilistic geometric problem, as follows.

\[
P(\text{Atlas}_R^C | \overrightarrow{T}_E^R, \overrightarrow{\text{Atlas}_E} | \overrightarrow{\text{Atlas}_R} | \overrightarrow{\text{Atlas}_E}, \alpha_E) = \frac{1}{Z} \int_{\overrightarrow{T}_C^R, \overrightarrow{\text{Atlas}_E}^R, \overrightarrow{\text{Atlas}_R}^R} P(\text{Atlas}_R^C | \overrightarrow{T}_C^R, \overrightarrow{\text{Atlas}_E}^R, \overrightarrow{\text{Atlas}_R}^R, \overrightarrow{\text{Atlas}_E}^R) \prod P(\overrightarrow{\text{Atlas}_R} | \overrightarrow{\text{Atlas}_E} | \overrightarrow{\text{Atlas}_R} | \overrightarrow{\text{Atlas}_E}, \alpha_E) \prod P(\overrightarrow{\text{Atlas}_E} | \overrightarrow{\text{Atlas}_R} | \overrightarrow{\text{Atlas}_E} | \overrightarrow{\text{Atlas}_R}, \alpha_E) \prod P(\overrightarrow{\text{Atlas}_R} | \overrightarrow{\text{Atlas}_E} | \overrightarrow{\text{Atlas}_R} | \overrightarrow{\text{Atlas}_E}, \alpha_E)
\]

The equation denominator in (1) is a normalization constant, consequently, we obtain the following.

\[
P(\text{Atlas}_R^C | \overrightarrow{T}_E^R, \overrightarrow{\text{Atlas}_E} | \overrightarrow{\text{Atlas}_R} | \overrightarrow{\text{Atlas}_E}, \alpha_E) \propto \int_{\overrightarrow{T}_C^R, \overrightarrow{\text{Atlas}_E}^R, \overrightarrow{\text{Atlas}_R}^R, \overrightarrow{\text{Atlas}_E}^R} P(\text{Atlas}_R^C | \overrightarrow{T}_C^R, \overrightarrow{\text{Atlas}_E}^R, \overrightarrow{\text{Atlas}_R}^R, \overrightarrow{\text{Atlas}_E}^R) \prod P(\overrightarrow{\text{Atlas}_R} | \overrightarrow{\text{Atlas}_E} | \overrightarrow{\text{Atlas}_R} | \overrightarrow{\text{Atlas}_E}, \alpha_E) \prod P(\overrightarrow{\text{Atlas}_E} | \overrightarrow{\text{Atlas}_R} | \overrightarrow{\text{Atlas}_E} | \overrightarrow{\text{Atlas}_R}, \alpha_E) \prod P(\overrightarrow{\text{Atlas}_R} | \overrightarrow{\text{Atlas}_E} | \overrightarrow{\text{Atlas}_R} | \overrightarrow{\text{Atlas}_E}, \alpha_E)
\]

We know the values \(\text{atlas}_R, \overrightarrow{T}_E^R\) and \(\overrightarrow{\text{Atlas}_E}\) of the variables \(\text{Atlas}_R, \overrightarrow{T}_E^R\) and \(\overrightarrow{\text{Atlas}_E}\) (e.g. \(\text{Atlas}_R = \text{atlas}_R, \overrightarrow{T}_E^R = \overrightarrow{\text{Atlas}_E}\)). We can therefore introduce the (constant) values \(P(\text{Atlas}_R = \text{atlas}_R), P(\overrightarrow{T}_E^R = \overrightarrow{\text{Atlas}_E})\) and \(P(\overrightarrow{\text{Atlas}_E} = \overrightarrow{\text{Atlas}_E})\) in the normalization constant obtaining a second constant \(Z\)', and obtain the following.

\[
P(\text{Atlas}_R^C | \overrightarrow{T}_E^R, \overrightarrow{\text{Atlas}_E} | \overrightarrow{\text{Atlas}_R} | \overrightarrow{\text{Atlas}_E}, \alpha_E) \propto \frac{1}{Z} \int_{\overrightarrow{T}_C^R, \overrightarrow{\text{Atlas}_E}^R, \overrightarrow{\text{Atlas}_R}^R, \overrightarrow{\text{Atlas}_E}^R} P(\text{Atlas}_R^C | \overrightarrow{T}_C^R, \overrightarrow{\text{Atlas}_E}^R, \overrightarrow{\text{Atlas}_R}^R, \overrightarrow{\text{Atlas}_E}^R) \prod P(\overrightarrow{\text{Atlas}_R} | \overrightarrow{\text{Atlas}_E} | \overrightarrow{\text{Atlas}_R} | \overrightarrow{\text{Atlas}_E}, \alpha_E) \prod P(\overrightarrow{\text{Atlas}_E} | \overrightarrow{\text{Atlas}_R} | \overrightarrow{\text{Atlas}_E} | \overrightarrow{\text{Atlas}_R}, \alpha_E) \prod P(\overrightarrow{\text{Atlas}_R} | \overrightarrow{\text{Atlas}_E} | \overrightarrow{\text{Atlas}_R} | \overrightarrow{\text{Atlas}_E}, \alpha_E)
\]

The distributions \(P(\overrightarrow{\text{Atlas}_R} | \overrightarrow{T}_E^R, \overrightarrow{\text{Atlas}_E})\) and \(P(\overrightarrow{T}_E^R | \overrightarrow{\text{Atlas}_R} | \overrightarrow{\text{Atlas}_E})\) are Diracs, so we can eliminate the sums in \(\overrightarrow{\text{Atlas}_R}^R\) and \(\overrightarrow{T}_C^R\) and the corresponding distributions. We obtain the following final expression.
3D human hip volume reconstruction using Bayesian techniques

Relevant Variables:

\[ \begin{align*}
\text{Atlas}^C &= \{ P^C_i \mid i \in I_{\text{Atlas}} \} \\
T^R_E &= \{ P^R_{i} \mid i \in I_{\text{R}} \} \\
T^R_E &= \{ P^R_{i} \mid i \in I_E \} \\
\text{Atlas}^E &= \{ P^E_i \mid i \in I_E \} \\
\text{Atlas}^R_E &= \{ P^R_{i} \mid i \in I_E \} \\
e_R &= \{ \varepsilon_R \mid i \in I_R \} \\
e_E &= \{ \varepsilon_E \mid i \in I_E \}
\end{align*} \]

Decomposition:

\[
P(\text{Atlas}^C, T^R_E, \text{Atlas}^R_e, \text{Atlas}^E, \text{Atlas}^R_e, T^R_E, \varepsilon_R, \varepsilon_E) =
\]

\[
P(\text{Atlas}^C) P(T^R_E) P(\text{Atlas}^E) P(\varepsilon_R) P(\varepsilon_E)
\]

Parametric Forms:

\[
P(\text{Atlas}^C) = \text{Normal}(\text{Atlas}^C, \text{atlas}_u, \Delta_1)
\]

\[
P(T^R_E) = \text{Normal}(T^R_E, \mu, \Delta_2)
\]

\[
P(\text{Atlas}^E) = \text{Unknown}
\]

\[
P(T^R_E) = \text{Unknown}
\]

\[
P(\text{Atlas}^R_E) = \text{Normal}(\text{Atlas}^R_E, \text{Atlas}^E, \Delta_3)
\]

\[
P(\varepsilon_R) = \text{Dirac}
\]

\[
P(\varepsilon_E) = \text{Dirac}
\]

Identification:

\[
\begin{align*}
\text{atlas}_u, \Delta_1, \Delta_2, \Delta_3, & \text{ Obtained by learning;} \\
\Delta_4, \Delta_5, \Delta_6 & \text{ A priori data; (See the text for more details).}
\end{align*}
\]

Question:

\[
\text{Best}(P(\text{Atlas}^C) \mid T^R_E = T^E_R \mid \text{Atlas}^E = \text{Atlas}_E) \ [\text{Atlas}^E = \text{Atlas}_R] \ [\varepsilon_R = \mathbf{0}] \ [\varepsilon_E = \mathbf{0})]
\]

Fig. 7. The Bayesian program for computing the patient’s atlas from echographic and radiographic data.

\[
P(\text{Atlas}^C, T^R_E, \text{Atlas}^E, \varepsilon_R, \varepsilon_E) \propto 
\]

\[
\frac{1}{Z^R} \int_{T^R_E \mid T^R_E = T^E_R} \left[ P(\text{Atlas}^C) P(T^R_E) P(\text{Atlas}^E) P(\varepsilon_R) P(\varepsilon_E) \right] dT^R_E
\]
Fig. 8. The graphic representation or Bayesian network of the program shown in Fig. 7. The dotted nodes correspond to the available information; the grey nodes correspond to the sought information.

with: $\text{Atlas}^{RB}_{E} = f_1(T^{RB}_{E}, P^{E}_{i})$ and $T^{RB}_{C} = f_2(P^{RB}_{i})$.

This final expression allows us to find the answer to the question of our Bayesian program: to infer the patient atlas from the available data taking into account the uncertainty generated at the moment of data acquisition. The most probable patient’s atlas is given by $\text{atlas}^*$, which is the value of $\text{Atlas}^C$ that maximizes (3).

5.5 Parametric forms and identification

Definition of $P(\text{Atlas}^C)$

Let $\{\text{Atlas}^C_1, \text{Atlas}^C_2, ..., \text{Atlas}^C_n\}$ be a partition of $\text{Atlas}^C$. $P(\text{Atlas}^C)$ represents the probabilistic atlas where:

$$P(\text{Atlas}^C) = \prod_{i=1}^{n} P(\text{Atlas}^C_i) \text{ with } P(\text{Atlas}^C_i) = \text{Normal}(\text{Atlas}^C_i, \text{atlas}^i_{\mu}, \Delta^i).$$

Each $\text{Atlas}^C_i$ is a subset of characteristic points of the atlas representing a region of the hip. This division by regions was made for two reasons: (i) the natural growth of the hip occurs as more or less independent regions, and (ii) this approach allows us to simplify the calculation. Using a single multivariate normal distribution for all the atlas points results in a much more difficult equation to optimize (see (3)).

The probabilistic atlas represents the generic hip model and consists of a probability distribution that captures the uncertainty of the proportions between the hip’s characteristic points. The probabilistic atlas is a model

\[ \text{We remark that } \text{Atlas}_R^C, \text{Atlas}_E^C \subset \text{Atlas}^C. \]
based on a statistical analysis of multivariate normal distributions. In our case, \textit{atlas}_\mu and \Delta_1 were obtained from the analysis of 52 files of data furnished by Kepple [Kepple, 1998].

\textit{Definition of }P(T^R_C)

\[ P(T^R_C) \] is a multivariate normal distribution centred in \( \mu \) and with a covariance matrix \( \Delta_2 \):

\[ P(T^R_C) = \text{Normal}(T^R_C, \mu, \Delta_2). \]

The values of \( \mu \) and the covariance matrix \( \Delta_2 \) are given \textit{a priori}. If we consider that the patient’s position at the time of radiographic image acquisition may vary \( \pm 3^\circ \) of orientation in the rotation angles \( \alpha \) and \( \beta \) in the transversal and sagittal plane axes, and \( \pm 1^\circ \) in the angle of the frontal plane axe, then \( \mu = 0 \).

\textit{Definition of }P(\text{Atlas}_R)

\[ P(\text{Atlas}_R) \] is the probability distribution of the radiographic points in the radiographic coordinate system. It can be considered as unknown because we know the value of \( \text{Atlas}_R \), and this allows to introduce the constant \( P(\text{Atlas}_R = \text{atlas}_R) \) in the constant \( Z' \) presented in Section 5.4.

\textit{Definition of }P(T^{RB}_{E_i} | \mathcal{T}^{RB}_E)

The distribution \[ P(T^{RB}_{E_i} | \mathcal{T}^{RB}_E) \] represents the probability of the transformation matrices from the echographic coordinate system to the coordinate system of the \textit{rigid body}, knowing their estimations \( T^{RB}_{E_i} \). This distribution is defined as:

\[ P(T^{RB}_{E_i} | \mathcal{T}^{RB}_E) = \prod_{i \in \mathcal{I}_E} P(T^{RB}_{E_i} | \mathcal{T}^{RB}_{E_i}), \]

where each \[ P(T^{RB}_{E_i} | \mathcal{T}^{RB}_{E_i}) \] is a multivariate normal distribution centred in \( T^{RB}_{E_i} \) with a standard deviation \( \Delta_3 \).

\[ P(T^{RB}_{E_i} | \mathcal{T}^{RB}_E) = \text{Normal}(T^{RB}_{E_i}, \mathcal{T}^{RB}_{E_i}, \Delta_3) \]

We obtained the covariance matrix value \( \Delta_3 \) by making several acquisitions of an echographic image in the same position. We fixed a \textit{rigid body} to the patient with a waistband and another \textit{rigid body} to the echographic sensor.

\textit{Definition of }P(\text{Atlas}_E)

\[ P(\text{Atlas}_E) \] corresponds to the probability distribution of the estimated echographic points \( \text{Atlas}_E \) in the echographic image coordinate systems \( E_i \). This distribution can be considered as unknown because we know the \( \text{Atlas}_E \) value, allowing us to introduce the constant \( P(\text{Atlas}_E = \text{atlas}_E) \) in the constant \( Z' \) presented in Section 5.4.
Definition of $P(\text{Atlas}_E|\text{Atlas}_E)$

$P(\text{Atlas}_E|\text{Atlas}_E)$ corresponds to the echographic points in the different coordinate systems of the echographic image $E_i$ according to their estimations $\text{Atlas}_E$. This distribution is defined as:

$$P(\text{Atlas}_E|\text{Atlas}_E) = \prod_{i \in I_E} P(P_i^{E_i}|P_i^{E_i}),$$

where $P(P_i^{E_i}|P_i^{E_i})$ is a multivariate normal distribution centred in $P_i^{E_i}$ and a covariance matrix $\Delta_4$.

For the covariance matrix $\Delta_4$, we consider that the surgeon is able to localize a characteristic point from an echographic image with an error of 1cm, but in reality, this value depends on the surgeon (i.e. one surgeon can be more accurate than another). We have estimated $\Delta_4$ using a test in which we placed a plaster acetabulum and femoral head and three iron markers in a gelatin-filled basin. We then took several echographic images in different positions to identify the markers, and from them we estimated the value of $\Delta_4$.

Definition of $P(\text{Atlas}_{RB}^E|\text{T}_{RB}^E \text{ Atlas}_E)$

$P(\text{Atlas}_{RB}^E|\text{T}_{RB}^E \text{ Atlas}_E)$ corresponds to the probability distribution of the echographic points in the coordinate system of the rigid body according to the transformation set $\text{T}_{RB}^E$ and the point set $\text{Atlas}_E$. We have:

$$P(\text{Atlas}_{RB}^E|\text{T}_{RB}^E \text{ Atlas}_E) = \prod_{i \in I_E} P(P_i^{RB}|P_i^{RB}, P_i^{E_i}),$$

where $P(P_i^{RB}|P_i^{RB}, P_i^{E_i})$ a Dirac distribution as follows:

$$P(P_i^{RB}|P_i^{RB}, P_i^{E_i}) = \begin{cases} 1 \text{ if } P_i^{RB} = f_1(T_{RB}^E, P_i^{E_i}), \\ 0 \text{ otherwise.} \end{cases}$$

where

$$f_1(T_{RB}^E, P_i^{E_i}) = T_{RB}^E P_i^{E_i}.$$

Definition of $P(\text{T}_{RB}^C|\text{Atlas}_{RB}^E)$

$P(\text{T}_{RB}^C|\text{Atlas}_{RB}^E)$ corresponds to the calculus of the transformation $\text{T}_{RB}^C$ between the rigid body $RB$ coordinate system and the atlas centre $C$. It is calculated from the points in $\text{Atlas}_{RB}^E$ and corresponds to a Dirac distribution defined as follows:
$$P(T_C^{RB} | P_i^{RB}) = \begin{cases} 1 & \text{If } T_C^{RB} = f_2(\text{Atlas}_E^{RB}) \\ 0 & \text{otherwise} \end{cases}$$

where $f_2(\text{Atlas}_E^{RB})$ is a function that describes the process of the following section.

The transformation matrix $T_C^{RB}$ is obtained from two elements: (i) the patient’s frontal plane corresponding to the $yz$ plane, and (ii) the patient’s atlas centre. These two elements are obtained from the echographic points. The transformation matrix $T_C^{RB}$ can be written as follows.

$$T_C^{RB} = \begin{pmatrix} R_C^{RB} & t \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} I_3 & t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} R_C^{RB} & 0 \\ 0 & 1 \end{pmatrix}$$ (4)

The rotation matrix $R_C^{RB}$ represents the orientation difference between the rigid body and the coordinate system of the atlas centre. The rigid body coordinate system has an arbitrary orientation and position. The rigid body is fixed to the patient’s hip in a more or less arbitrary orientation. Certainly, the rigid body positioning did not influence the acquisition of echographic points. In contrast, the coordinate system of the atlas centre has a defined orientation and position. In this case, the $yz$ plane is parallel to the patient’s frontal plane, and the $y$-axis coincides with the patient’s longitudinal axis (see Fig. 9a). In this case, the origin is placed at the atlas centre.

The patient’s frontal plane (Fig. 9) is described by the points $P_3, P_{16}, P_{31}$. Consequently, the coordinate system of the rigid body orientation is obtained as follows.

The $x$-axis is obtained from the normal $n$ of the frontal plane. If $v_1 = \overrightarrow{P_3 P_{16}}$ and $v_2 = \overrightarrow{P_3 P_{31}}$ we have:

$$n = v_1 \times v_2.$$  

Then

$$x = [x_x, x_y, x_z]^T = \frac{n}{\|n\|}.$$ 

The $z$-axis is the vector obtained from the normalization of $\overrightarrow{P_{31} P_3}$:

$$z = [z_x, z_y, z_z]^T = \frac{\overrightarrow{P_{31} P_3}}{\|P_{31} P_3\|} \text{ and } y = [y_x, y_y, y_z]^T = z \times x.$$ 

To obtain the $t$ translation vector, we require the six points $P_1, P_2, P_3, P_7, P_{30}$ and $P_{35}$. These points allow us to obtain the atlas centre defined as the centre of the parallelepiped enclosing the hip. This parallelepiped is aligned with the coordinate system $C$: in consequence, $t_x, t_y$ and $t_z$ are calculated from the pairs $(P_3, P_{35}), (P_1, P_7)$ and $(P_2, P_{30})$ (See Fig. 9b).

If
Fig. 9. The coordinate system obtained from the patient’s frontal plane described by the points \( P_3, P_{16}, P_{31} \) (a). The origin of this coordinate system is the patient’s atlas centre. The parallelepiped enclosing the hip and the parallelepiped centre (b).

\[
P_1' = [x_1', y_1', z_1'] = R_{RB}^{C} P_1
\]

\[
P_{35}' = [x_{35}', y_{35}', z_{35}'] = R_{RB}^{C} P_{35},
\]

the atlas centre is at the rigid body origin but with the atlas centre orientation. It is calculated as follows.

\[
x' = \frac{(x_{35}' + x_3')}{2} \quad y' = \frac{(y_1' + y_7')}{2} \quad z' = \frac{(z_2' + z_{30}')}{2}
\]

Then:

\[
\begin{bmatrix}
t_x \\
t_y \\
t_z
\end{bmatrix} = R_{RB}^{C} \begin{bmatrix}
x'
\\
y'
\\
z'
\end{bmatrix}.
\]
With this computation, the centre is always the same, regardless of the patient’s hip dimensions.

**Definition of** $P(e_R \mid Atlas_R \; T_C^R \; Atlas_R^C)$

$P(e_R \mid Atlas_R \; T_C^R \; Atlas_R^C)$ corresponds to the cycle constraint generated by the atlas radiographic points in the coordinate system of the radiographic image.

$$P(e_R \mid Atlas_R \; T_C^R \; Atlas_R^C) = \prod_{i \in I_R} P(\varepsilon_{Ri} \mid P_i^R \; T_C^R \; P_i^C),$$

with $P(\varepsilon_{Ri} \mid P_i^R \; T_C^R \; P_i^C)$, which is a normal distribution centred in $\|P_i^R - T_C^R \; P_i^C\|$, and a covariance matrix $\Delta_5$.

$$P(\varepsilon_{Ri} \mid P_i^R \; T_C^R \; P_i^C) = \text{Normal}(\varepsilon_{Ri}, \|P_i^R - T_C^R \; P_i^C\|, \Delta_5)$$

$\Delta_5$ represents the admitted error interval for the equality constraint of the closing cycle between the atlas points and the radiographic points. We consider that $\Delta_5$ has an *a priori* value with an admissible error of 3 mm.

**Definition of** $P(e_E \mid Atlas_E^{RB} \; T_C^{RB} \; Atlas_E^C)$

$P(e \mid Atlas_E^{RB} \; T_C^{RB} \; Atlas_E^C)$ corresponds to the probability distribution of the generated errors in the estimation of the echographic points.

$$P(e \mid Atlas_E^{RB} \; T_C^{RB} \; Atlas_E^C) = \prod_{i \in I_E} P(\varepsilon_{Ei} \mid P_i^{RB} \; T_C^{RB} \; P_i^C),$$

where each distribution $P(\varepsilon_{Ei} \mid P_i^{RB} \; T_C^{RB} \; P_i^C)$ is a multivariate normal distribution centred in $\|P_i^{RB} - T_C^{RB} \; P_i^C\|$ and a covariance matrix $\Delta_6$:

$$P(\varepsilon_{Ei} \mid P_i^{RB} \; T_C^{RB} \; P_i^C) = \text{Normal}(\varepsilon_{Ei}, \|P_i^{RB} - T_C^{RB} \; P_i^C\|, \Delta_6).$$

$\Delta_6$ represents the allowable interval error from the equality constraint of closing the cycle between the atlas points and the echographic points. We consider that $\Delta_6$ has an *a priori* value with an admissible error of 3 mm.

In the next section, we present the implementation, experimentation and results for the 3D hip volume acquisition of 52 individuals.

### 6 Implementation, experimentation and results

#### 6.1 Bayesian atlas learning

To execute the Bayesian atlas learning, we proceed as follows. From a database containing $m$ instances of a subset of the 70 characteristic points
(described in Section 2.1), we defined $I_{Atlas} \subset \{1, 2, \ldots, 70\}$ and the partition \{Atlas$_1^C$, Atlas$_2^C$, \ldots, Atlas$_m^C$\} of Atlas$^C$ as functions of the available points. Finally, we identified the averages and the covariance matrices of the multivariate normal distributions $P(Atlas_1), P(Atlas_2), \ldots, P(Atlas_m)$. The database was obtained from Thomas M. Kepple [Kepple, 1998] at the Biomechanics Laboratory of the National Health Institute in Bethesda United States. These data come from 52 human skeletons selected from the Terry collection of the Smithsonian Institute in Washington. Kepple provided 52 files corresponding to the 52 skeletons. Each file has 46 hip points, of which 30 points belong to the 70 characteristic points defined in Section 2.3. The 16 remaining points are not used for the Bayesian atlas construction. However, we will show how these points can be used for validation.

In our experiments, the index set $I_{Atlas}$ has 30 points. These were used for learning $P(Atlas^C)$, particularly:

$$I_{Atlas} = \{2, 3, 4, 7, 15, 16, 17, 24, 25, 26, 27, 28, 30, 31, 32, 35, 39, 40, 44, 45, 58, 59, 61, 64, 65, 66, 67, 68, 69, 70\}.$$

Once $I_{Atlas}$ was defined, we selected the sets of indexes $I_E$, $I_{E'}$, and $I_R$ that correspond respectively to the indexes of the echographic points to calculate $T^R_{CB}$ and to the radiographic points:

$I_E = \{2, 3, 7, 16, 26, 28, 30, 31, 35, 40\}$

$I_{E'} = \{2, 3, 7, 16, 30, 31\}$

$I_R = \{2, 3, 4, 7, 15, 16, 17, 24, 25, 26, 27, 28, 30, 31, 32, 45, 45, 58, 59, 61, 64, 67, 68\}$

Figure 10 shows the characteristic points of our atlas. The set of atlas points that are not echographic or radiographic are called non-observable points, that is, they are not observed by echography or radiography. Note that a non-observable point is inferred from the other atlas point values. Certainly, the covariance matrix of the Bayesian atlas returns the probability value of the point given the known data.

We partitioned Atlas$^C$ into six subsets \{Atlas$_1^C$, Atlas$_2^C$, \ldots, Atlas$_6^C$\}, defined as follows.

$$\begin{align*}
\text{Atlas}_1^C &= \{P_{12}^C, P_{13}^C, P_{15}^C, P_{30}^C, P_{31}^C, P_{32}^C, P_{66}^C, P_{69}^C\} \\
\text{Atlas}_2^C &= \{P_{35}^C, P_{36}^C, P_{40}^C, P_{45}^C, P_{66}^C, P_{70}^C\} \\
\text{Atlas}_3^C &= \{P_{15}^C, P_{16}^C, P_{17}^C, P_{24}^C, P_{61}^C, P_{67}^C\} \\
\text{Atlas}_4^C &= \{P_{30}^C, P_{31}^C, P_{32}^C, P_{40}^C, P_{45}^C, P_{66}^C, P_{69}^C\} \\
\text{Atlas}_5^C &= \{P_{28}^C, P_{25}^C, P_{44}^C, P_{59}^C\} \\
\text{Atlas}_6^C &= \{P_{26}^C, P_{27}^C, P_{45}^C, P_{64}^C, P_{68}^C\}
\end{align*}$$

Each subset has two characteristics.

1. All points in a subset belong to the same growing region of the hip [Moseley, 1986, Jacquemier, 1991].

2. At least one echographic point is contained in each subset.

This choice was motivated by two factors: first, we wanted to group points related by the natural growing process of the hip; second, we wanted to include
at least one point containing information about the three proportions (in the directions \(x, y, z\)) of the hip.

At this stage, all the main elements enabling the definition of our Bayesian program are defined (See Section 5.4 p. 257).

Once the patient’s echographic and radiographic data are obtained, it is necessary to calculate the following expression 5.

\[
\max_{\text{atlas}^C \in \text{Atlas}^C} \left( \int_{T^R_E \cdot T^R_B} P(\text{atlas}^C) P(T^R_E) P(T^R_B | T^R_E = T^R_B) P(\text{Atlas}_E | \text{Atlas}_R = \text{atlas}^E) P(e_E = 0 | \text{Atlas}_R = \text{atlas}^R) \right)
\]

with  \(\text{Atlas}^R_E \equiv f_1(T^R_E, P^E_i)\) and  \(T^R_C = f_2(\text{Atlas}^R_E)\).

In the next section, we explain the methods selected to effect the calculation.

6.2 Resolution method

We have written our Bayesian program in ProBT® , a program that was developed by the e-Motion team at GRAVIR’s laboratory. It was conceived for writing Bayesian programs. ProBT® is a C++ library that enables the development of software using Bayesian techniques. It is available for several platforms. The ProBT® library has two components: (i) an Application Program Interface (API) to construct Bayesian models, and (ii) a high-performance Bayesian inference machine (MIB) that will allow the execution of all probabilistic calculations in an exact or approximate mode.

5 Recall that \(\text{atlas}^E, \text{atlas}^C \subset \text{atlas}\).
As remarked previously, the resolution of the expression (6) is divided into two subproblems: (i) integral resolution and (ii) maximization. In the following sections, we briefly describe the techniques used by ProBT\textsuperscript{®} for the resolution of these two problems in our implementation.

**Numerical estimation of the integrals**

The integral calculations (sums) are the basis of Bayesian inference. Unfortunately, analytic integration methods seem limited in real-world applications where functions have complex forms and the integration spaces have very large dimensionality. Additionally, these techniques are not useful for inference in general purpose systems where the distributions are not simply probability charts.

In our implementation, we therefore opted for numeric integration by the Monte Carlo method, which is described in Chap. 4.1.

**Numerical optimization of the distribution**

ProBT\textsuperscript{®} also allows the user to select the optimization method. In our case, we opted for the gradient-descent method. This method looks to find a local maximum from an initial point given by $x^{(0)}$. This calculation is iterative, and the optimal point is found using:

$$x^{(n+1)} = x^{(n)} + \Delta x^{(n)}.$$  

We require an initial point from Atlas:\textsuperscript{C}:

$$\text{atlas}^0 = (p_2^0, p_3^0, ..., p_{70}^0) = (x_2^0, y_2^0, z_2^0, x_3^0, y_3^0, ..., x_{70}^0, y_{70}^0, z_{70}^0),$$

from which the gradient descent starts. To determine the initial point, we have proceeded as follows.

First, we calculated an approximation of the matrix $T_{RB}^C$ from the approximate echographic points and the matrices $T_{RB}^E$. The calculation of this matrix is similar to the calculation of $T_{RB}^E$, which was described in Section 5.5 (e.g. $T_{RB}^C$ is the inverse of $T_{RB}^E$). We could thus obtain an approximation of the echographic points according to the approximate atlas centre denoted atlas:\textsuperscript{E} with:

$$\text{atlas}^E = (p_{E2}, p_{E3}, p_{E7}, ..., p_{E40}) = (x_{E2}, y_{E2}, z_{E2}, x_{E3}, y_{E3}, ..., x_{E40}, y_{E40}, z_{E40}).$$

Similarly we denoted the set of radiographic point values by:

$$\text{atlas}^R = (p_{R2}, p_{R3}, p_{R4}, ..., p_{R68}) = (x_{R2}, y_{R2}, z_{R2}, x_{R3}, y_{R3}, ..., x_{R68}, y_{R68}, z_{R68}).$$

We could then apply the following rules to establish the value of $P^{0}_i = (x_i^0, y_i^0, z_i^0) \in \text{atlas}^0$. 
1. If the point is echographic or radiographic, then the \( x \) component of \( p_0^i \) is fixed by the \( x \) component of the echographic point \( p_{Ei} \), and the \( y \) and \( z \) components are fixed by the radiographic point \( p_{Ri} \).

2. Otherwise, if the point is echographic but not radiographic, then the point \( p_0^i \) is fixed by the value of the echographic point.

3. Otherwise, if the point is radiographic but not echographic, then the \( x \) component of the point \( p_0^i \) is fixed by the average of \( x_i \) in the atlas. The \( y \) and \( z \) values are then fixed by the value of the radiographic point.

4. Finally, if none of the preceding cases apply, then the point \( p_0^i \) is simply approximated by the average of point \( i \) in the Bayesian atlas.

The previous rule assumes that the information furnished by radiography is more certain than the information furnished by echography or the averages, while the information furnished by echography is more certain than the information furnished by the averages. This procedure allows us to initialize the gradient descent in a promising region of the searching space.

6.3 Results

In the following sections we describe our experiments. To test our system, we conducted two experiments: (i) \textit{in vivo} and (ii) in simulation.

The \textit{in vivo} experiment consisted of executing the steps described in Section 4, using a radiographic image from a real person and making an echographic study. The aim of this experiment was to validate the system’s feasibility and to identify any problems linked to the echographic study.

The simulation experiment consisted of generating artificial data from the database from Terry’s collection and then making the inference and the deformation. In this case, we simulated the radiographic and echographic data acquisition. The aim of this experiment was to make a statistical study to evaluate the method. The statistics obtained will give us an idea of the expected error values when the system is used in real cases and also an approximation of the computation time.

\textit{In vivo} experimental results

To conduct our \textit{in vivo} experiment, we searched for a volunteer who had anteroposterior hip radiography and who would agree to have an echographic study, so we could obtain the necessary data to test the system. We used the data from the anteroposterior hip radiographic image and the echographic images of the volunteer to obtain the 3D hip volume.

In Fig. 11 we present the 3D hip model obtained from the radiographic points (dark points) and the echographic points (light points) of the volunteer. The initial mesh deforms to adapt to the characteristic points of the patient.

The main difficulty that we found in this case was the acquisition of the set of echographic images. The image acquisition was performed by one of the authors, because no expert was available. Certainly, it is necessary to have a
knowledge of medical imagery to recognize (in echographic images) the bone structures containing the characteristic points. However, even with the errors generated by lack of experience, the results seem to be coherent, at least for the hip topology obtained.

In this work, the main limit was the accessibility of the patient’s medical data. Certainly, access to the MRI/CT/scanner images we required to make the validation was difficult to obtain because of confidentiality. On the other hand, we could not ask a volunteer to make a MRI/CT/scanner study without real medical reasons. This experiment is therefore complementary to the simulation experiments.

Simulation experiment results

In our simulation experiment, we generated artificial 2D and 3D data using the information provided from Terry’s collection. We proceeded as follows.

1. For the radiographic data, we first transformed the data of the individuals to the atlas centre, because the original data are with respect to an unknown coordinate system. We then obtained the transformation matrix $t^{R}_B$ from the atlas centre to the radiographic image. This transformation is generated randomly using the distribution $P(T^{R}_B)$. We next transformed the radiographic points in the atlas centre coordinate system to the coordinate system of the artificial radiography. These points were then atlas$_R$.

2. For each echographic point $i$ of the original database (according to the rigid body with an unknown position), we generated its transformation matrix $t^{RB}_{EE}$. In this matrix, the orientation is fixed randomly and uniformly, while its origin is fixed by a random drawing from a normal distribution. This distribution has a covariance matrix that returns points within a distance of 1.5 cm from the point $i$. In this way, we obtained the set of the transformation values $t^{RB}_{EE}$ of the variable $T^{RB}_{EE}$.
3. Finally, we calculated the points $p_{E_i}$ in $\text{atlas}_E$ with a random drawing, following a normal distribution centred on zero and with a covariance matrix returning points within a distance of 2.0 cm.

In this way, we experimented with 52 “virtual patients” from the database. When we were executing the $i$th patient’s hip reconstruction, we omitted his or her data from the learning.

We obtained the execution time of the Bayesian program and the average error between the inferred and the real atlas points. The experiment was made approximating the sum (in the Monte Carlo integration method) with $N = 100$.

We were also interested in the errors of the different sets of points: (i) echographic, (ii) radiographic, (iii) echographic and radiographic and (iv) non-observable.

We also obtained average distances for the database points that are not included in our atlas to the resulting mesh. These errors will allow us to measure the average distance that could exist between the model of a real patient and the inferred model. The average values of all tests are presented in Table 1.

### Table 1

<table>
<thead>
<tr>
<th>Average</th>
<th>Error in mm</th>
</tr>
</thead>
<tbody>
<tr>
<td>All points</td>
<td>8.1</td>
</tr>
<tr>
<td>Echographic points</td>
<td>6.1</td>
</tr>
<tr>
<td>Radiographic points</td>
<td>6.7</td>
</tr>
<tr>
<td>Echographic and radiographic points</td>
<td>6.1</td>
</tr>
<tr>
<td>Non-observed points</td>
<td>12.3</td>
</tr>
<tr>
<td>Points not included in the atlas</td>
<td>10.4</td>
</tr>
</tbody>
</table>

3. The in vivo experiment allowed us to conclude that it is possible to collect the data required by the system. The main difficulty is the echographic data acquisition, which requires expertise. In our case, the acquisition of the set of echographic images took approximately two hours, but an expert could have acquired it in a few minutes. If we consider the calculation time of the inference process, the hip model could be obtained in a few hours, including the acquisition of the radiographic and echographic data.

Our simulation experiments show that the hip models obtained are coherent to the hip topology. We displayed each of these models to allow a visual inspection and to appreciate where errors appeared. In this way, we remarked that for most of the control points where the error was important, the inferred point was on the mesh surface. We found this characteristic interesting. We consider that the topology of the obtained models respects the hip geometry characteristics (see Fig. 12): the optimization allows us to find a coherent model using the learning technique. For example, if we compare...
the hip model obtained from the inferred atlas with the model obtained from the initial points of the gradient descent (\textit{atlas}\textsuperscript{0}), we remark that this model has a deformed structure. This shows that the optimization leads to coherent points in the searching space.

The \textit{in vivo} and simulation experiments show that the method is feasible and promising. As far as we know, this is the first reconstruction method that has been tested experimentally.

References


Playing to train your video game avatar

Ronan Le Hy\textsuperscript{1} and Pierre Bessi\`ere\textsuperscript{2}

\textsuperscript{1} INPG – LIG Lab
\textsuperscript{2} CNRS – Grenoble Universit\é

The goal of this chapter is to demonstrate how, by using the Bayesian inverse programming technique, a player of a video game can teach an avatar how to play.

However, we first show how inverse programming is also very useful in simplifying the programming burden of a video game development team.

Bayesian inverse programming consists of expressing independently the conditional probabilities of the conditions, knowing the action. Although atypical, this modelling method appears to be convenient and generic and to lead to very simple learning schemes.\textsuperscript{3}

1 Introduction

Today's video games feature synthetic characters involved in complex interactions with human players. A synthetic character may have one of many different roles: a tactical enemy, a partner for the human player, a strategic opponent, a simple unit among many, or a substitute for the player when he or she is unavailable.

In all of these cases, the game developer's ultimate objective is for the synthetic character to act as if it were controlled by a human player. This implies the illusion of spatial reasoning, memory, commonsense reasoning, using goals, tactics, planning, communication and coordination, adaptation, unpredictability, and so on. In current commercial games, basic gesture and motion behaviours are generally satisfactory. More complex behaviours usually look much less lifelike. Sequencing elementary behaviours is an especially difficult

\textsuperscript{3} This work was partially funded by a PhD grant from the French Ministry of Research, by the European Project BIBA (Bayesian Inspired Brain and Artefact), by the European project BACS (Bayesian Approach to Cognitive Systems) and by the CNRS ROBEA project AV-Bayes. Preliminary results of this work have been published in Robotics and Autonomous Systems [Le Hy et al., 2004].
problem, as compromises must be made between too-systematic behaviour that looks automatic and too-random behaviour that looks ridiculous.

We address this problem of real-time reactive selection of elementary behaviours for an agent playing a first-person shooter (FPS) game called *Unreal Tournament*. In this kind of game, a group of people play together via the Internet. Each of them can control a virtual avatar. This avatar may act and navigate in a virtual 3D environment. It may also interact with the avatars of other players or with autonomous characters called *bots* controlled by a program. In FPS games, the main interaction with the other players and bots consists of trying to slaughter them while surviving as long as possible (see Fig. 1).

![An Unreal Tournament scene.](image)

1.1 Objective 1: simplifying the development team’s task

Programming bots is an important, difficult and time-consuming task for the game development team. Simplifying this task is our first objective in this work.

If an avatar could learn while being operated by the player, it could gain its autonomy and become a bot. Many players would love to have autonomous bots playing for them and like them when they are away from the keyboard. A first step in this direction is our second objective.

From the development team’s viewpoint, several goals must be reached.
1. **Human-like behaviour** – The goal for this kind of bot is to give the impression that the player is playing with a human opponent. Furthermore, the opponent should be challenging, but not invincible.

2. **Design–Development separation** – The industrial development scheme often draws a distinction between game designers and software developers. The development process should offer the designers ways to describe behaviours at a high conceptual level, without any knowledge of the engine’s internals or esoteric computer languages.

3. **Programming efficiency** – One crucial concern for developers is productivity. They require both expressivity and simplicity from the behaviour programming system.

4. **Behaviour tunability** – The ability to program a variety of different behaviours and to adjust each of them without modifying the system’s backend is essential to the designer.

5. **Limited computational resources** – The processing time allotted to AI in games is typically between 10% and 20% of the total processing time [Woodcock, 2001]. Therefore, it is important that the behaviour system should use little computation time.

1.2 **Objective 2: training the video game avatar**

The second objective is very simple to specify even if it is difficult to fulfil. The player wants to be able to train an avatar to play the game, without changing his or her usual way of playing. The ideal interface is the player’s usual one, plus a single on/off learning button telling the avatar when to learn.

The expected result is a clone of the player: a bot playing with the same efficiency and the same style. The ideal would be a bot that other human players could not distinguish from its teacher.

1.3 **The inverse programming principle**

To reach these objectives, we propose to use a special kind of Bayesian model called *inverse programming*.

Usually, bots’ behaviours in video games are specified using a scripting language. The behaviour is defined by a set of rules that sequence the activation of elementary reactive behaviours. Each rule is a list of conditions and constraints on the sensory variables that must be verified to select the corresponding reactive behaviour. The generic form of such a rule is: if condition$_1$ and condition$_2$ and ... and condition$_n$ then reactive behaviour.

There are two main ideas in inverse programming:

1. Knowing which reactive behaviour the robot is doing provides a lot of information on its sensory variables.
2. Knowing the reactive behaviour, we can consider, as a first approximation, that these sensory variables are independent of one another.
Consequently, the rules are replaced by probability distributions expressing
chunks of knowledge of the form if reactive\textit{behaviour}_i then (approximately)
\textit{condition}_j. This is why it is called \textit{inverse} programming.

2 Inverse programming of video game characters’ behaviours

For this work, we are using Unreal Tournament augmented with the \textit{Game-}
\textit{Bots} [Kaminka et al., 2002] control framework. This setup provides a thre-
dimensional environment in which players fight each other, taking advantage
of resources such as weapons and health bonuses available in the arena.

Our bot communicates with Unreal Tournament via a text protocol on a
Unix socket. It receives messages covering its perceptions, such as its position
and speed, health level, ammunition, and visible opponents and objects. The
environment is perceived by the bot as a graph with nodes that are character-
istic points of the topology. The bot perceives only what is in its field of
vision.

In return, it sends actions: move to a given point, rotate, change weapons.

Six elementary reactive behaviours have been programmed based on these
actions: attacking foes (\textit{attack}), searching for weapon bonuses (\textit{searchweapon}),
searching for health bonuses (\textit{searchhealth}), exploring the environment (\textit{explore}),
fleeing (\textit{flee}), and detecting danger (\textit{detectdanger}). In the attack mode, the
bot shoots at an opponent while maintaining a constant distance and strafing.
The two search behaviours try to collect either a weapon or a health bonus
that the bot has noticed in its environment. When exploring, the bot navig-
gates around the environment and discovers unknown parts of it. The fleeing
behaviour consists of trying to escape (locally) from an opponent. Finally,
the last reactive behaviour is trying to detect possible opponents outside the
current field of view of the bot.

The problem is to sequence these reactive behaviours in time.

2.1 Bayesian program

Relevant variables

The relevant variables are as follows.

\begin{itemize}
\item \textit{B}_t, the reactive behaviour of the bot at time \textit{t}, which can take any of the six
values: \{\textit{attack}, \textit{searchweapon}, \textit{searchhealth}, \textit{explore}, \textit{flee}, \textit{detectdanger}\}.
\item \textit{H}_t, the bot’s health level at time \textit{t}.
\item \textit{W}_t, the weapon that the bot is deploying at time \textit{t}.
\item \textit{OW}_t, the opponent’s weapon at time \textit{t}.
\item \textit{N}_t, a Boolean variable indicating whether a noise has been heard recently.
\item \textit{NO}_t, the number of nearby enemies at time \textit{t}.
\end{itemize}
- \( WP^t \), the proximity of a weapon bonus.
- \( HP^t \), the proximity of a health bonus.

Each of these eight variables must be considered at each instant between time 0 and \( T \). This leads to a conjunction of \( 8 \times (T + 1) \) variables: \( B_0^0 \land \ldots \land H P_0^T \).

### Decomposition

The joint distribution is decomposed as follows.

\[
\begin{align*}
P(B_0^0 \land H_0^0 \land W_0^0 \land OW_0^0 \land NO_0^0 \land WP_0^0 \land HP_0^0) \\
= \prod_{t=1}^{T} \left[ P(B_t | B_{t-1}) \times P(HP_t | B_{t-1}) \times P(WP_t | B_{t-1}) \times P(NO_t | B_{t-1}) \times P(W_t | B_{t-1}) \times P(N_t | B_{t-1}) \times P(NO_0 | B_{t-1}) \times P(W_0 | B_{t-1}) \times P(HP_0 | B_{t-1}) \right] \\
\times P(B_0^0 \land H_0^0 \land W_0^0 \land OW_0^0 \land N_0^0 \land NO_0^0 \land WP_0^0 \land HP_0^0)
\end{align*}
\]

In this decomposition, we assume that:

- the behaviour \( B_t \) at time \( t \) depends on the behaviour \( B_{t-1} \) at time \( t - 1 \); and
- the seven sensory variables may be considered to be independent of one another and independent of the past, knowing the behaviour \( B_t \).

A very similar assumption is often made for sensor fusion when using the naive Bayes scheme. Here, the common cause for the observations that justify the conditional independence hypothesis is an action instead of a phenomenon. The important difference is that an action can be selected, while a phenomenon can only be observed. Consequently, our distribution can be used to select the appropriate action, knowing the observations, while the sensor fusion distribution could only be used to infer the most probable value for the observed phenomenon.

### Parametric forms

The eight terms of the decomposition \( P(B_t | B_{t-1}) \cdots P(HP_t | B_t) \) are defined using tables specifying their discrete values. We will describe in detail how to specify these tables later.

### Identification

For the time being, we are interested in programming behaviours, and the values in these tables are specified by hand. There are no free parameters, and no identification is necessary. 

In the second part of the chapter, when we are interested in bots’ training, these values will be obtained from observation of the way that players operate their avatars. Identification will then be the crucial point of the process.

---

4 Recall that the notation \( B_0^t \) stands for the conjunction of \( T + 1 \) variables: \( B_0^0 \land B_1^1 \land \ldots \land B_T^T \).
Question

Every tenth of a second, our bot must make a decision on its behaviour. It must answer the following probabilistic question.

\[ P(B_t^i \mid b_t^{-1} \wedge h_t \wedge w_t \wedge ow_t \wedge n_t \wedge no_t \wedge wp_t \wedge hp_t) \]  \hspace{1cm} (2)

What is the probability distribution on behaviour at time \( t \) \( (B_t^i) \), knowing the behaviour at time \( t-1 \) \( (b_t^{-1}) \) and knowing all the sensory information at time \( t \) \( (h_t,\ldots, hp_t) \)?

This question leads to a probability distribution, from which we draw a value to decide the actual new behaviour. The answer to this question may be easily computed as it is proportional to the product of the individual terms.

\[
P(B_t^i \mid b_t^{-1} \wedge h_t \wedge w_t \wedge ow_t \wedge n_t \wedge no_t \wedge wp_t \wedge hp_t) 
\propto P(B_t^i \mid b_t^{-1}) \times P(h_t \mid B_t^i) \times P(w_t \mid B_t^i) \times P(ow_t \mid B_t^i)
\times P(n_t \mid B_t^i) \times P(no_t \mid B_t^i) \times P(wp_t \mid B_t^i) \times P(hp_t \mid B_t^i) \hspace{1cm} (3)
\]

Summary

The corresponding Bayesian program is then given by Fig. 2.

2.2 Table specification

Sensor variables knowing reactive behaviour

The tables corresponding to the eight nonuniform distributions appearing in the decomposition are specified by hand by the game developers.
For instance, the transition distribution \( P(B^t \mid B^{t-1}) \) may be defined by Table 1. This table expresses the game developer’s expectation of stability of the behaviours: when an avatar is doing something, the most probable action is that it will continue doing the same thing, unless it finds some good reasons to change. For instance, in the first column of Table 1, if the avatar is attacking, then its has a probability of 95\% to continue attacking and a probability of 1\% to switch to one of the five other possible behaviours.

Changing the value of the table modifies the behaviour of the avatar. A much more aggressive avatar (“berserk”) could be specified with a different distribution \( P(B^t \mid B^{t-1}) \), as in Table 2, where whatever its present behaviour, the avatar has a 48\% chance of switching back to attack.

### Table 1. \( P(B^t \mid B^{t-1}) \) “cautious” mode

<table>
<thead>
<tr>
<th></th>
<th>attack</th>
<th>searchweapon</th>
<th>searchhealth</th>
<th>explore</th>
<th>flee</th>
<th>detectdanger</th>
</tr>
</thead>
<tbody>
<tr>
<td>attack</td>
<td>( x )</td>
<td>( 10^{-1} )</td>
<td>( 10^{-1} )</td>
<td>( 10^{-1} )</td>
<td>( 10^{-1} )</td>
<td>( 10^{-1} )</td>
</tr>
<tr>
<td>searchweapon</td>
<td>( 10^{-1} )</td>
<td>( x )</td>
<td>( 10^{-1} )</td>
<td>( 10^{-1} )</td>
<td>( 10^{-1} )</td>
<td>( 10^{-1} )</td>
</tr>
<tr>
<td>searchhealth</td>
<td>( 10^{-1} )</td>
<td>( 10^{-1} )</td>
<td>( x )</td>
<td>( 10^{-1} )</td>
<td>( 10^{-1} )</td>
<td>( 10^{-1} )</td>
</tr>
<tr>
<td>explore</td>
<td>( 10^{-1} )</td>
<td>( 10^{-1} )</td>
<td>( 10^{-1} )</td>
<td>( x )</td>
<td>( 10^{-1} )</td>
<td>( 10^{-1} )</td>
</tr>
<tr>
<td>flee</td>
<td>( 10^{-1} )</td>
<td>( 10^{-1} )</td>
<td>( 10^{-1} )</td>
<td>( x )</td>
<td>( 10^{-1} )</td>
<td>( 10^{-1} )</td>
</tr>
<tr>
<td>detectdanger</td>
<td>( 10^{-1} )</td>
<td>( 10^{-1} )</td>
<td>( 10^{-1} )</td>
<td>( 10^{-1} )</td>
<td>( 10^{-1} )</td>
<td>( x )</td>
</tr>
</tbody>
</table>

### Table 2. \( P(B^t \mid B^{t-1}) \) “berserk” mode

<table>
<thead>
<tr>
<th></th>
<th>attack</th>
<th>searchweapon</th>
<th>searchhealth</th>
<th>explore</th>
<th>flee</th>
<th>detectdanger</th>
</tr>
</thead>
<tbody>
<tr>
<td>attack</td>
<td>( x )</td>
<td>( x )</td>
<td>( x )</td>
<td>( x )</td>
<td>( x )</td>
<td>( x )</td>
</tr>
<tr>
<td>searchweapon</td>
<td>( 10^{-1} )</td>
<td>( x )</td>
<td>( 10^{-1} )</td>
<td>( 10^{-1} )</td>
<td>( 10^{-1} )</td>
<td>( 10^{-1} )</td>
</tr>
<tr>
<td>searchhealth</td>
<td>( 10^{-1} )</td>
<td>( 10^{-1} )</td>
<td>( x )</td>
<td>( 10^{-1} )</td>
<td>( 10^{-1} )</td>
<td>( 10^{-1} )</td>
</tr>
<tr>
<td>explore</td>
<td>( 10^{-1} )</td>
<td>( 10^{-1} )</td>
<td>( 10^{-1} )</td>
<td>( x )</td>
<td>( 10^{-1} )</td>
<td>( 10^{-1} )</td>
</tr>
<tr>
<td>flee</td>
<td>( 10^{-1} )</td>
<td>( 10^{-1} )</td>
<td>( 10^{-1} )</td>
<td>( x )</td>
<td>( 10^{-1} )</td>
<td>( 10^{-1} )</td>
</tr>
<tr>
<td>detectdanger</td>
<td>( 10^{-1} )</td>
<td>( 10^{-1} )</td>
<td>( 10^{-1} )</td>
<td>( 10^{-1} )</td>
<td>( 10^{-1} )</td>
<td>( x )</td>
</tr>
</tbody>
</table>

The seven other probability distributions define the likelihood of making some observation with a sensor, knowing the behaviour. For instance, \( P(H^t \mid B^t) \), defined in Table 3, gives the probability distribution for \( H^t \) (the bot’s health level), knowing the behaviour \( B^t \). We read the first column this way: given that the bot is in state \( attack \), we state that it has a very low probability \((10^{-3})\) of having a low (poor) health level, a medium probability

\[5\] \( x \) stands for the missing value to have a normalized column. In this first column, \( x = 0.95 \).
of having a medium (fair) health level, and a strong probability (0.899) of having a high (good) health level.

<table>
<thead>
<tr>
<th></th>
<th>attack</th>
<th>searchweapon</th>
<th>searchhealth</th>
<th>explore</th>
<th>flee</th>
<th>detectdanger</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>$10^{-3}$</td>
<td>$10^{-1}$</td>
<td>$x$</td>
<td>$10^{-1}$</td>
<td>0.7</td>
<td>$10^{-1}$</td>
</tr>
<tr>
<td>Medium</td>
<td>$10^{-1}$</td>
<td>$x$</td>
<td>$10^{-2}$</td>
<td>$x$</td>
<td>0.2</td>
<td>$x$</td>
</tr>
<tr>
<td>High</td>
<td>$x$</td>
<td>$x$</td>
<td>$10^{-3}$</td>
<td>$x$</td>
<td>0.1</td>
<td>$x$</td>
</tr>
</tbody>
</table>

This form of specification allows us a convenient formalization of the constraints that we want to impose on the behaviour, in a condensed format, and separately on each sensory variable. Indeed, Table 3 formalizes the relation of the bot’s health level to its state: if it starts attacking, then its health is rather high; if it starts searching for a health pack, then its health is almost certainly low; if it starts fleeing, then its health is probably rather low, but with a high degree of uncertainty.

All six tables for the other sensory variables are built on similar patterns.

2.3 Results

Human-like behaviour

Several observations can be made when our bots are playing the game.

The first is that their behaviour corresponds to what we expected: neither too systematic nor too random. The bots seem to use spatial reasoning to achieve tactical goals, seem to adapt and react rapidly to events, yet also seem to use short-term planning to determine future actions.

The second is that they can play with humans, even if they cannot compete with the best players (see Table 4). Scores are the average point differences from the winning bot, over 10 games won when the first bot reaches 100 kills (for example, a bot with 76 points at the end of a game has a point difference of 24 from the winning bot, because the game ends as soon as any bot reaches 100 points). Therefore, a bot that wins all games would have a score of zero. Our bots compare well with the native Unreal Tournament bot, with skills corresponding to an average human player.

Design–development separation

To specify the behaviour of the bots, the only thing that the designer of the game must do is fill in the tables. No programming skill is required, nor any proficiency in probability. Behaviours are data, not programs. This ensures complete separation between programmers and designers. Furthermore,
Table 4. Performance comparison (1). Lower is better: minimum 0, maximum 100.

<table>
<thead>
<tr>
<th>Behaviour</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random behaviour</td>
<td>43.2</td>
</tr>
<tr>
<td>Unreal Tournament bot (3/8) (≈ average human)</td>
<td>11.0</td>
</tr>
<tr>
<td>Cautious bot, manual specification</td>
<td>12.2</td>
</tr>
<tr>
<td>Berserk bot, manual specification</td>
<td>8.0</td>
</tr>
</tbody>
</table>

it means that behaviours can easily be loaded and saved even while the game is running, and they can be exchanged within a community of players or developers.

Programming efficiency

To program the decomposition, we make the hypothesis that knowing $B^t$, any sensory variable is independent of the others. Although it may seem to reduce the expressivity of our model, it allows us to specify it in a very condensed way.

In the game industry, this kind of bot is usually programmed with a scripting language that specifies the behaviour as a simple finite state machine (FSM). Let us consider the case where we have $n$ sensory variables, taking an average number of $m$ possible values. In an FSM that models behaviour, each state would require a transition to every other state, in the form of a logical condition on the sensory variables. Thus, the programmer must discriminate among the $m^n$ possible sensory combinations to describe the state transitions.

Many different possible formalisms exist to specify this kind of FSM, but in any case, such scripts are hard to write, hard to maintain, and hard to extend when adding either new reactive behaviours or new sensory variables.

In contrast, for each sensory variable combined with each possible behaviour, our approach consists of giving a distribution (i.e., numbers summing to 1). In practice, this results in tables such as Tables 1, 2 and 3. To add a new reactive behaviour, we simply add a new column to each table. To add a new sensory variable, we specify a new table. Instead of specifying the conditions that make the bot switch from one behaviour to another, we specify the probability distributions for the sensors’ values when the bot begins a given behaviour. Using this way of specifying a sensor under the hypothesis that we know the behaviour is why we call this method inverse programming.

Although somewhat confusing at first, this is the core advantage of our way of specifying the sequence. We describe the influence of each sensor on the bot’s state separately, thereby drastically reducing the quantity of information required. The number of values required to specify a sequence completely is $b^2 + (n \times (b \times m))$ where $b$ is the number of elementary behaviours, $n$ is the number of sensory variables, and $m$ is the average number of possible values.
for the sensory variables. This is linear in the number of variables \( n \), instead of being exponential as in the FSM approach.

**Behaviour tunability**

To change the behaviour requires only changes in the values stored in the tables. This is how we developed the more aggressive bot, by changing Table 1 into Table 2.

This change has a direct impact on the performance of the bot, as the berserk bot reaches a score of 8.0 while the cautious one has a score of 12.2 (see Table 4).

The most important thing to note is that this behavioural adaptation is made without changing a single line of code. It could even be made dynamically during a game, to simulate a bot with different moods.

**Limited computation requirements**

Equation 3 shows that to select a reactive behaviour, the computer must perform comparisons between 6 values (one for each possible reactive behaviour). Each of these values is obtained as the product of 8 terms, and each of these terms corresponds to a memory access to one of the tables. It is obviously a very fast computation that requires only a tiny part of the computer power.

### 3 Training avatars

Our second objective is to transform an avatar into a bot.

We want to teach the avatar a behaviour, instead of specifying all the probability distributions by hand. The distributions that must be learned are the eight elementary distributions appearing in the decomposition.

\[
P(B^0_t \land H^0_t \land W^0_t \land OW^0_t \land N^0_t \land NO^0_t \land W^P_{0:t} \land HP^0_{0:t})
\]

\[
= \prod_{t=1}^{T} \left[ \left( \frac{P(B^t | B^{t-1})}{P(B^t | B^{t-1})} \times P(H^t | B^t) \times P(W^t | B^t) \times P(OW^t | B^t) \times P(N^t | B^t) \times P(NO^t | B^t) \times P(W^P_{t} | B^t) \times P(HP^t_{t} | B^t) \right) \right] \times P(B^0 \land H^0 \land W^0 \land OW^0 \land N^0 \land NO^0 \land W^P_{0} \land HP^0_{0}) \tag{4}
\]

If at each instant (every tenth of a second) we know both the values of the seven sensory variables \( (H^t, W^t, OW^t, N^t, NO^t, WP^t, HP^t) \) and the value of \( B^t \), this is a trivial problem of counting the number of occurrences of each situation. This information is available for the sensory variables, but may not be available for \( B^t \).

Determining a value for \( B^t \) at each time step can be done either by letting the player specify this value directly or by inferring it from the observation of the way the player is controlling the avatar in the usual interface to the game.
3.1 Behaviour selection

To find the value of $B^t$ directly, suppose we design a specific teaching interface where the player must click on radio buttons to select one of the six possible behaviours.

It is very easy to design such a specific interface, but it has the major disadvantage of changing the way the player usually interacts with the game.

The results obtained by this method (see Table 5) show that the player is not able to play efficiently with this new interface. Performance is even worse than a bot selecting its behaviours completely at random. Consequently, we should abandon this approach and find a method of recognizing the behaviour from the usual interactions.

### Table 5. Performance comparison (2). Lower is better: minimum 0, maximum 100.

<table>
<thead>
<tr>
<th>Behaviour</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random behaviour</td>
<td>43.2</td>
</tr>
<tr>
<td>Unreal Tournament bot (3/8) ($\simeq$ average human)</td>
<td>11.0</td>
</tr>
<tr>
<td>Cautious bot, manual specification</td>
<td>12.2</td>
</tr>
<tr>
<td>Berserk bot, manual specification</td>
<td>8.0</td>
</tr>
<tr>
<td>Berserk bot, behaviours selection</td>
<td>45.7</td>
</tr>
</tbody>
</table>

3.2 Heuristic recognition of behaviours

To solve this problem, it is possible to let the player use the natural interface of the game, and try to recognize behaviours in real time.

Our first attempt to recognize the human’s behaviour from low-level actions used a heuristic programmed in a classical imperative fashion.

This required identifying each behaviour’s critical characteristics (for instance, attack is characterized by the distance and speed of the bot relative to characters in the centre of the player’s field of view) and triggering recognition at several timescales. Recognition is achieved by examining a series of criteria in sequence; the first that matches is chosen. The first criterion is a characteristic event that is back-propagated to states in the past that have not yet been recognized (for instance, picking a health bonus indicates that the character has been looking for health). The second examines critical variables over a fixed period (for instance, danger checking is characterized by a complete rotation with little translation, in a short amount of time). Finally, some short-term variations of critical variables are examined (like attacking and fleeing, identified by moves in particular directions in the presence of opponents). Exploration is a default state, when a state does not match any of the criteria.
We do this recognition offline, on data representing 10 to 15 minutes of game play; processing these data and producing the tables that represent our behaviour takes 5 to 10 seconds. Consequently this kind of learning could be also done online in real time.

The results obtained this way are excellent (see Table 6).

The main disadvantage of this approach is that the heuristic is specific to the considered behaviours and is quite difficult to design.

Table 6. Performance comparison (3). Lower is better: minimum 0, maximum 100.

<table>
<thead>
<tr>
<th>Behaviour</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random behaviour</td>
<td>43.2</td>
</tr>
<tr>
<td>Unreal Tournament bot (3/8) (≃ average human)</td>
<td>11.0</td>
</tr>
<tr>
<td>Cautious bot, manual specification</td>
<td>12.2</td>
</tr>
<tr>
<td>Berserk bot, manual specification</td>
<td>8.0</td>
</tr>
<tr>
<td>Berserk bot, behaviours selection</td>
<td>45.7</td>
</tr>
<tr>
<td>Cautious bot, heuristic recognition</td>
<td>13.9</td>
</tr>
<tr>
<td>Berserk bot, heuristic recognition</td>
<td>4.4</td>
</tr>
</tbody>
</table>

3.3 Incremental expectation/maximization for recognition of behaviours

The very well-known Baum–Welch [Baum, 1972, Rabiner, 1989] algorithm, a special case of the expectation maximization (EM) class of algorithms [Dempster et al., 1977], was designed especially for learning when some data are missing. This is the case when trying to train the avatars, as the value of the behaviour $B_t$ at each time step is missing.

We apply an incremental version of the Baum–Welch algorithm as described by Florez-Larrahondo [Florez-Larrahondo, 2005]. In this algorithm, contrary to the classical Baum–Welch algorithm, the model parameters are re-estimated after each new observation. This algorithm can treat at least 10 sensory–motor acquisitions per second allowing learning online as the player is using the game.

The results obtained are very good, as shown in Table 7\(^6\).

It is interesting to compare the tables obtained with this learning-by-demonstration method and the tables specified manually.

For instance, the transition tables between $B_t^{t-1}$ and $B_t^t$ are very similar (see Tables 8 and 9). Human players, indeed, tend to have stable behaviours.

Similar comparisons can be made for the tables concerning the sensory variables knowing the behaviour. For instance, we can compare $P(H_t^t | B_t^t)$ (see Tables 10 and 11).

\(^6\) A movie of the behaviour of the obtained avatar may be downloaded at the following URL: http://www.bayesian-programming.org/videoB1Ch11-1.html
Table 7. Performance comparison (3). Lower is better: minimum 0, maximum 100.

<table>
<thead>
<tr>
<th>Behaviour</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random behaviour</td>
<td>43.2</td>
</tr>
<tr>
<td>Unreal Tournament bot (3/8) (≃ average human)</td>
<td>11.0</td>
</tr>
<tr>
<td>Cautious bot, manual specification</td>
<td>12.2</td>
</tr>
<tr>
<td>Berserk bot, manual specification</td>
<td>8.0</td>
</tr>
<tr>
<td>Berserk bot, behaviours selection</td>
<td>45.7</td>
</tr>
<tr>
<td>Cautious bot, heuristic recognition</td>
<td>13.9</td>
</tr>
<tr>
<td>Berserk bot, heuristic recognition</td>
<td>4.4</td>
</tr>
<tr>
<td>Incremental Baum–Welch</td>
<td>8.5</td>
</tr>
</tbody>
</table>

Table 8. $P(B^t \mid B^{t-1})$ obtained by the incremental Baum–Welch algorithm

<table>
<thead>
<tr>
<th></th>
<th>attack</th>
<th>searchweapon</th>
<th>searchhealth</th>
<th>explore</th>
<th>flee</th>
<th>detectdanger</th>
</tr>
</thead>
<tbody>
<tr>
<td>attack</td>
<td>0.78</td>
<td>0.0024</td>
<td></td>
<td>0.023</td>
<td>0.028</td>
<td>0.052</td>
</tr>
<tr>
<td>searchweapon</td>
<td>0.014</td>
<td>0.96</td>
<td>0.012</td>
<td>0.016</td>
<td>0.017</td>
<td>0.017</td>
</tr>
<tr>
<td>searchhealth</td>
<td>0.16</td>
<td>0.02</td>
<td>0.94</td>
<td>0.041</td>
<td>0.06</td>
<td>0.16</td>
</tr>
<tr>
<td>explore</td>
<td>0.017</td>
<td>0.0049</td>
<td>0.0078</td>
<td>0.86</td>
<td>0.038</td>
<td>0.048</td>
</tr>
<tr>
<td>flee</td>
<td>0.016</td>
<td>0.0047</td>
<td>0.009</td>
<td>0.018</td>
<td>0.78</td>
<td>0.068</td>
</tr>
<tr>
<td>detectdanger</td>
<td>0.017</td>
<td>0.0059</td>
<td>0.0068</td>
<td>0.038</td>
<td>0.048</td>
<td>0.66</td>
</tr>
</tbody>
</table>

Table 9. $P(B^t \mid B^{t-1})$ “cautious” mode ($x = 0.95$)

<table>
<thead>
<tr>
<th></th>
<th>attack</th>
<th>searchweapon</th>
<th>searchhealth</th>
<th>explore</th>
<th>flee</th>
<th>detectdanger</th>
</tr>
</thead>
<tbody>
<tr>
<td>attack</td>
<td>$x$</td>
<td>$10^{-1}$</td>
<td></td>
<td>$10^{-1}$</td>
<td>$10^{-1}$</td>
<td></td>
</tr>
<tr>
<td>searchweapon</td>
<td>$10^{-1}$</td>
<td>$x$</td>
<td>$10^{-1}$</td>
<td>$10^{-1}$</td>
<td>$10^{-1}$</td>
<td>$10^{-1}$</td>
</tr>
<tr>
<td>searchhealth</td>
<td>$10^{-1}$</td>
<td>$10^{-1}$</td>
<td>$x$</td>
<td>$10^{-1}$</td>
<td>$10^{-1}$</td>
<td>$10^{-1}$</td>
</tr>
<tr>
<td>explore</td>
<td>$10^{-1}$</td>
<td>$10^{-1}$</td>
<td>$10^{-1}$</td>
<td>$x$</td>
<td>$10^{-1}$</td>
<td>$10^{-1}$</td>
</tr>
<tr>
<td>flee</td>
<td>$10^{-1}$</td>
<td>$10^{-1}$</td>
<td>$10^{-1}$</td>
<td>$10^{-1}$</td>
<td>$x$</td>
<td>$10^{-1}$</td>
</tr>
<tr>
<td>detectdanger</td>
<td>$10^{-1}$</td>
<td>$10^{-1}$</td>
<td>$10^{-1}$</td>
<td>$10^{-1}$</td>
<td>$10^{-1}$</td>
<td>$x$</td>
</tr>
</tbody>
</table>

These two tables are quite different. The only obvious similarities are that when searching for a weapon, it seems that the avatar has a rather high health and that when fleeing the avatar has a rather low health. Indeed, Table 10 does not encode much information. The distributions are flat, not very different from uniform. This may be explained by the fact that picking up a health bonus is a rather more opportunistic behaviour than a planned one. Whenever a player is passing close to a health bonus, it is collected. Players rarely search for them.

3.4 Results

Two of the three experimental methods succeeded in training video game avatars online and in real time. For both of them, the avatar reached the
Table 10. $P(H^t \mid B^t)$ obtained by the incremental Baum–Welch algorithm

<table>
<thead>
<tr>
<th></th>
<th>attack</th>
<th>searchweapon</th>
<th>searchhealth</th>
<th>explore</th>
<th>flee</th>
<th>detectdanger</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>0.16</td>
<td>0.098</td>
<td>0.12</td>
<td>0.22</td>
<td>0.23</td>
<td>0.23</td>
</tr>
<tr>
<td>Medium</td>
<td>0.48</td>
<td>0.30</td>
<td>0.41</td>
<td>0.36</td>
<td>0.67</td>
<td>0.36</td>
</tr>
<tr>
<td>High</td>
<td>0.36</td>
<td>0.6</td>
<td>0.47</td>
<td>0.42</td>
<td>0.1</td>
<td>0.41</td>
</tr>
</tbody>
</table>

Table 11. $P(H^t \mid B^t)$ manually specified

<table>
<thead>
<tr>
<th></th>
<th>attack</th>
<th>searchweapon</th>
<th>searchhealth</th>
<th>explore</th>
<th>flee</th>
<th>detectdanger</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>$10^{-3}$</td>
<td>$10^{-1}$</td>
<td>0.989</td>
<td>$10^{-1}$</td>
<td>0.7</td>
<td>$10^{-1}$</td>
</tr>
<tr>
<td>Medium</td>
<td>$10^{-1}$</td>
<td>0.45</td>
<td>$10^{-2}$</td>
<td>0.45</td>
<td>0.2</td>
<td>0.45</td>
</tr>
<tr>
<td>High</td>
<td>0.899</td>
<td>0.45</td>
<td>$10^{-3}$</td>
<td>0.45</td>
<td>0.1</td>
<td>0.45</td>
</tr>
</tbody>
</table>

level of an average human player. The second method, using the incremental Baum–Welch learning algorithm, is preferable to the first one using heuristics, as the heuristics are problem specific and difficult to define.

We have achieved our objective of training an avatar by cloning the behaviour of the player. Indeed, the resulting avatars play with the same style as the player; for instance, their aggressiveness mimics the player’s.

Finally, as the behavioural decisions are made according to random draws on probability distributions, the resulting global comportment of the avatars seems neither too systematic nor too random. We questioned some human opponents of these avatars, and it appeared difficult for them to tell whether they were playing against a bot or a human player.

4 Conclusion and perspectives

In this chapter, we showed that it is possible to train a video game avatar by playing it.

Both our objectives have been reached: simplifying the development team task and enabling a player to train an avatar.

The work presented in this chapter is a proof of concept. There is much work still to be done to create an industrial toolbox.

We will continue our efforts in this direction, as we are convinced of the importance of the industrial perspectives in the video game industry.

- The first perspective is to reduce the time and cost of development of video games by simplifying the production of the very numerous necessary bots. Using the Bayesian inverse programming technique, the game designer will be able to define bots’ behaviours without writing a line of computer code. Eventually, bots will be trained by demonstration. Designers will have great flexibility and efficiency in tuning and changing these behaviours.
• The second perspective is to offer players a new functionality: training by playing.

References

Bayesian modelling of visuo-vestibular interactions

Jean Laurens$^1$ and Jacques Droulez$^1$

CNRS and Collège de France, LPPA Laboratory

1 Introduction

In addition to the five senses usually described, vertebrate species possess a sensory organ that detects motion of the head. This organ is the vestibular system, located in the inner ear. Motion information collected by the vestibular system is crucial for equilibrium. It also contributes to stabilizing the gaze in space during head movements. Motion information provided by the vestibular system generates compensatory eye movement, a phenomenon called the Vestibulo–Ocular Reflex (VOR). The importance of this function is illustrated by the following example (from Guedry [1974]): you can look at the lines on your hand and shake your head at the same time. The VOR provides efficient gaze stabilization in this condition. In contrast, if you shake your hand, looking at the lines becomes impossible.

1.1 Presentation of the vestibular organs

Several sets of sensory organs participate in motion perception (Fig. 1a). Within the vestibular system, the semicircular canals react to rotatory motion. These canals are circular tubes filled with a viscous liquid called endolymph. During an angular acceleration of the head, inertia generates a displacement of the endolymph relative to the canal (as an analogy, imagine you put a cup of tea in the middle of the turntable of a record player: when you start the rotation of the turntable, the cup will rotate but the tea remains stable in space). The displacement activates hair cells in the cupula and generates a signal along the vestibular nerve. However, during a prolonged rotation, viscosity causes the endolymph to rotate at the same speed as the canals (as the tea would after a few minutes in our example). In consequence, the signal coming from the canals fades to zero. One result of this effect is that when a long-duration rotation is followed by a quick stop, the deceleration activates the canals in a direction opposite to the initial acceleration (this is termed a
Fig. 1. Presentation of the vestibular system and simple psychophysical results. 
a: Schematic drawing of the vestibular system, after Gray [1918]. b: Geometrical conventions for egocentric coordinates and for rotations, and an illustration of the gravito-inertial ambiguity. c: Typical dynamics of the vestibulo-ocular reflex (VOR) during a constant-velocity rotation followed by a stop, showing the responses of the optokinetic nystagmus (OKN) during visual stimulation and of the optokinetic after-nystagmus (OKAN) after a visual stimulation. d: Somatogravic effect illustrated by eye movement recordings in the monkey. Roll tilt at high or low frequency elicits compensatory eye movements (ocular counter-rolling). Lateral acceleration creates an otolithic input similar to the one elicited by rolling. High-frequency acceleration creates a horizontal eye response whereas low-frequency acceleration elicits counter-rolling.
Bayesian modelling of visuo-vestibular interactions

In each ear, there are three semicircular canals that are roughly orthogonal. This allows us to detect rotations in all directions.

Another part of the vestibular system, the otoliths, detects linear acceleration. The otoliths are cavities filled with liquid. Inside these cavities, an agglomerate of crystals, the otoconia, rest on a layer of hair cells, the macula. When the head is accelerated, the otoconia move relative to the macula and activate the hair cells, generating a neural signal. There are two sets of otoliths in each ear, each of which is sensitive to two directions of motion. As a consequence, the otolithic system can sense accelerations in three dimensions. An important point is that the otoconia are deflected by gravity as well as linear acceleration. Therefore, the information provided by the otoliths is the gravito-inertial acceleration (GIA) \( F = G - A \), where \( G \) is the gravity and \( A \) is the acceleration (see Fig. 1b). As a consequence, tilting of the head and linear acceleration activate the otoliths in roughly the same manner. This creates an ambiguity called gravito-inertial ambiguity. This problem was described by Einstein, who showed that no linear accelerometer can distinguish gravity and acceleration.

1.2 Visual and proprioceptive information

Motion of the head is also perceived by the visual system. This system is not accurate and quick enough to sustain a good equilibrium during locomotion or an efficient gaze stabilization during high-frequency motion. However, visual information is essential at lower frequencies. Finally, during active head movements, proprioceptive information contributes to motion perception. In the present chapter, we will not take this information into account but will focus on experimental studies performed with passive stimulations.

1.3 Commonly experienced motion illusions

Motion perception is accurate in most natural situations. However, artificial or unusual stimulation can generate illusory motion percepts. Children enjoy standing and spinning around. After a few turns, this generates a sensation of dizziness, especially when one stops. This is a direct consequence of the dynamics described above. Another familiar illusion occurs when one sits in an immobile train and watches an adjacent train move. This generates an irressipible feeling of self-motion called vection. This effect is usually studied by placing stationary subjects inside a rotating drum (called an optokinetic drum). This generates an illusion of rotation.

Finally, another important – although more uncommon – illusion results from the gravito-inertial ambiguity. During a sustained linear acceleration, one feels tilted rather than accelerating. This effect can be experienced during the take-off of an aircraft. The pressure on the back exerted by the seat as well as the otolithic input generated by the acceleration are the same as those
generated when the plane tilts. Passenger who do not look through the window experience the feeling that the plane is tilted before it actually is. This illusion, called the somatogravic effect, is particularly important (and dangerous) during catapult launches on an aircraft carrier. If a jet pilot overestimates the tilt of the plane and attempts to compensate it, he or she will drive the plane downward and crash. Fortunately, modern aircraft are equipped with accurate inertial platforms that use rotation-sensitive gyroscopes and provide the pilots with reliable tilt information.

1.4 Experimental studies of motion perception and VOR

The importance of the vestibular system in aeronautics as well as its relative simplicity as a model of multisensory integration have motivated wide research efforts. Experiments have been conducted on a variety of species, particularly on monkeys that exhibit strong and robust VOR responses. These experiments allow us to formulate a variety of principles that govern visual and vestibular information processing.

During a constant-velocity rotation in the dark around a vertical axis, the velocity of the VOR is initially close to the rotation velocity. However, it decreases exponentially over time, in a manner similar to the canal signal (Fig. 1c). Yet, the VOR lasts longer than the canal signal. The time constant of the VOR is typically 20–25 s, whereas the time constant of the canals is 4 s. This indicates that the brain performs an integration over time of canals information, a process called velocity storage (see Raphan et al. [1977] for an overview of velocity storage and its implications for visuo-vestibular interactions). In contrast, if a subject is rotated around a horizontal axis – as on a barbecue – the VOR persists indefinitely Guedry [1965], Benson and Bodin [1966], Angelaki et al. [2000]. In this situation, the head is constantly reorienting with respect to gravity. This allows the subject to perceive the rotation even after the canal signal has faded out.

When a subject is translated laterally at a high frequency in the dark (e.g. 1 Hz), a linear VOR is observed, which is compensatory for the translation. During lateral tilt, which produces the same otolithic input but additionally activates the canals, the eyes compensate for the tilt by rolling in their orbits (see Fig. 1d). This indicates that the brain is able to use canal information to solve the gravito-inertial ambiguity. However, during lateral translation at a low frequency (e.g. 0.1 Hz), torsional eye movements are observed (see Fig. 1d, Angelaki [1998]). At these frequencies, the canal signal is weak, and the brain cannot use it to discriminate between translation and tilt. The fact that eye movements are compensatory for a tilt indicates that the brain favours a solution of the gravito-inertial ambiguity that corresponds to low accelerations. This is commonly explained by the fact that low-frequency accelerations are infrequent in natural situations.

When a monkey is placed in a rotating optokinetic drum, reflexive eye movements called Optokinetic Nystagmus (OKN) appear in the direction of
drum rotation. The OKN exhibits a specific dynamic: at the beginning of the visual stimulation (see Fig 1c, Raphan et al. [1977]), its velocity rises to about 60% of the rotation velocity. Then it rises exponentially to a value close to the drum velocity. Furthermore, if the light is shut down, the eye movements persist (they are then called the optokinetic after-nystagmus, OKAN). The OKAN velocity decays exponentially, with a time constant of 20–25 s. The dynamic of OKN and the existence of OKAN indicate that the eye movements do not simply “track” the patterns on the drum but correspond to an estimate of motion in the brain.

1.5 Modelling vestibular and visual information processing

Together with experimental studies, mathematical modelling has contributed to the understanding of visuo-vestibular interactions. Among others, the Raphan–Cohen model (Cohen et al. [1977], Raphan et al. [1979]) provides an accurate description of visuo-vestibular interactions during rotations around a vertical axis. Modelling responses to combined canal, otolith and visual stimulations in three dimensions is somewhat more challenging. Models such as Zupan’s Zupan et al. [2002] and the three-dimensional model of Raphan and Cohen Raphan and Cohen [2002] are able to simulate a variety of experimental paradigms, at the expense of a certain complexity.

One of the major issues in understanding vestibular information processing is the way the gravito-inertial ambiguity can be solved. To compute the linear acceleration of the head, one must subtract gravity (G) from the otolithic signal (F). This implies that the brain should know the orientation of the head relative to gravity. Different experimental studies have suggested that the brain possesses an internal estimate of gravity (\( \hat{G} \)) that is subtracted from F: this is the GIA resolution hypothesis. The orientation of \( \hat{G} \) in egocentric coordinates would be tracked using rotation information from the canals (for instance during high-frequency roll or tilt) Merfeld and Zupan [2002]. Furthermore, there would be a general tendency of \( \hat{G} \) to tend towards F, which would explain the somatogravic effect: during low-frequency acceleration, \( \hat{G} \) would remain close to F. The existence of \( \hat{G} \) is still a subject of debate: we refer the reader to Glasauer and Merfeld [1997], Merfeld et al. [1999] for theoretical and experimental evidence in favour of this hypothesis, and Raphan and Cohen [2002], Moore et al. [2005] for an alternative point of view.

Although the debate is still active, the GIA resolution hypothesis can explain a variety of experimental results in a simple way. It has led to the development of observer models that assume that the brain uses an internal estimate of motion. Expected sensory inputs corresponding to the internal estimate are computed. The difference between these expected signals and the signals actually received produces error signals that are used to correct the estimates.

The framework of observer theory can be extended by Bayesian inference. Whereas, in observer models, the modeller must specify the way the feedback
mechanisms work, Bayesian inference allows us to bypass this issue, because it relies on only a probabilistic internal model. Furthermore, Bayesian inference allows us to express the a priori probability distribution of motion. This is of wide interest in visuo-vestibular interaction processing. Indeed, it has long been recognized that the dynamic of the VOR (compensatory at high frequency but not at low frequencies) is adapted to natural conditions because in these conditions low-frequency rotations are infrequent. The same consideration holds for the somatogravic effect: long-duration linear accelerations are rare.

This motivated the use of Bayesian inference for the modelling of VOR. We built a Bayesian model and showed that a wide range of visuo-vestibular and canal–otolith interactions could be simulated in a simple fashion. Motion variables are linked by deterministic relations (e.g. acceleration is the derivative of velocity). Deterministic models of canals and otoliths can be found in the literature. We used a deterministic model of motion and sensory variables, to which we added the assumptions that information coming from the semicircular canals and from the visual system are corrupted by Gaussian noise, and that, a priori, angular velocity and linear acceleration follow Gaussian distributions. This allowed us to build a model using only four parameters. In a previous study Laurens and Droulez [2007], we showed that this model could reproduce motion perception in humans in a variety of situations. In the present chapter, we use it to simulate the VOR in monkeys.

1.6 Lesions of the canals

Lesions of the canals provide a demonstration of their contribution to visual and vestibular information processing. Canal plugging is a surgical operation in which the canals are obstructed. This reduces their time constant without affecting their gain at high frequency Rabbitt et al. [1999]. We performed simulations using a model adapted to the dynamics of plugged canals and compared their results to experimental results. The ability of our model to reproduce the consequences of canal plugging is an illustration of the flexibility and generality of the Bayesian framework.

2 Model

2.1 Motion variables

In this chapter, $X$, $Y$ and $Z$ refer to the three axes of the head, illustrated in Fig 1.b, in egocentric coordinates. For instance, if we describe a subject as lying horizontally and rotating around the $Z$ axis, we mean that the subject is rotating around an earth-horizontal axis (because in this position $Z$ is earth-horizontal).
Rotations and translations of the head in space are encoded using the following variables.

The orientation of the head in space is encoded using a rotation matrix $\Theta$. If the centre of the head is positioned at the origin of a geocentric reference frame, the first column of $\Theta$ represents the coordinates of a unit vector pointing along $X$. Similarly, the second and third columns of $\Theta$ are the coordinates of unit vectors pointing along $Y$ and $Z$.

The angular velocity of the head is encoded using the yaw, pitch and roll conventions. Yaw rotations are rotations around the $Z$ axis; pitch around the $Y$ axis and roll around $X$, as illustrated in Fig. 1b. When a rotation consists of a combination of yaw, pitch and roll rotation, the three rotations are applied successively and in that order.

If, between times $t$ and $t + \delta t$, the head rotates from angles $\delta y$, $\delta p$ and $\delta r$ in yaw, pitch and roll successively, $\Theta^{t+\delta t}$ can be computed as follows, using the matrix $R$ defined in Table 1.

$$\Theta^{t+\delta t} = \Theta^t R(\delta y, \delta p, \delta r)$$

$$R(y, p, r) = \begin{pmatrix}
\cos(y) \cos(p) & \cos(y) \sin(p) \sin(r) - \sin(y) \cos(r) & \cos(y) \sin(p) \cos(r) + \sin(y) \sin(r) \\
\sin(y) \cos(p) & \cos(y) \cos(p) + \sin(y) \sin(p) \sin(r) & -\sin(y) \sin(p) \cos(r) + \sin(y) \sin(r) \\
-\sin(p) & \cos(p) \sin(r) & \cos(p) \cos(r)
\end{pmatrix}$$

Table 1. Rotation matrix corresponding to angles $y$, $p$ and $r$ in yaw, pitch and roll.

Furthermore, instantaneous angular velocity is defined as the following vector.

$$\Omega = \begin{pmatrix}
\frac{\delta y}{\delta t} \\
\frac{\delta p}{\delta t} \\
\frac{\delta r}{\delta t}
\end{pmatrix}$$

Linear motion of the head is described by the position of the centre of the head in a geocentric reference frame, defined as a position vector $M$. The linear acceleration $A$ is the second derivative of $M$ with respect to time.

2.2 Sensory input

Given the motion of the head, we define the probability distributions of the sensory inputs as follows.

- We compute the deflection of the cupula $C^t$ of the semicircular canals, by applying a high-pass filter to the velocity $\Omega^t$. The vestibular signal $V^t$ is the sum of $C^t$ and a Gaussian noise.
- In the simulations presented in this chapter, visual input is restricted to angular motion. We assume that the visual input $O^t$ is the sum of $\Omega^t$ and a Gaussian noise.
• Otolithic input reports the vector $F$ described above. We did not add noise to this input because it does not affect our simulation results: it would only create a systematic underestimation of linear acceleration without affecting the other variables.

We compute $C^t$ by adapting a model proposed by Raphan and Cohen [2002]. This model projects angular acceleration onto the plane of the canals using a matrix $T_{can}$ and then simulates their dynamics using a first-order differential equation.

$$\frac{dC}{dt} = -\frac{1}{T_c} C - T_{can} \frac{d\Omega}{dt}$$

In this equation, $T_c$ is the time constant of the canals ($T_c = 4$ s) and the following.

$$T_{can} = \begin{pmatrix}
1 & 0 & 0 \\
-0.12 & -0.7 & 0.7 \\
-0.12 & -0.7 & -0.7 \\
\end{pmatrix}$$

The matrix $T_{can}$ encodes the orientation of the canals in space: its first row contains the coordinates of the lateral canals in the $(Z,Y,X)$ frame of reference (which corresponds to the yaw, pitch and roll axes of rotation). The second and third rows contain the coordinates of the anterior and posterior canals.

Given $C^t$, the vestibular canal signal $V^t$ is $V^t = C^t + \eta^t$, where $\eta^t$ is a three-dimensional vector; its elements follow independent Gaussian distributions with mean 0 and standard deviation $\sigma_V$.

Similarly, the visual input is $O^t = \Omega^t + \eta^t_v$. The elements of $\eta_v^t$ follow independent Gaussian distributions with mean 0 and standard deviation $\sigma_o$.

Given the acceleration $A$ and head orientation $Theta$, we compute the otolithic signal $F$ as follows: in a geocentric frame of reference, gravity is a vector $G = (0, 0, -9.81)$ and the gravito-inertial acceleration is $G - A$. We transform it into egocentric coordinates to compute $F$.

$$F = \Theta^{-1}(G - A)$$

2.3 A priori

Even in the absence of any sensory information, motion estimates for which the rotational velocity and acceleration are low are more probable. We describe this in a simple way using a Gaussian distribution. If we note:

$$\mathcal{N}_{x,\mu,\sigma} = \frac{e^{-\frac{(x-\mu)^2}{2\sigma^2}}}{\sqrt{2\pi}\sigma},$$

we define: $P(A^t) \propto \mathcal{N}_{|A^t|,0,\sigma_A}$.

We also compute the total rotational velocity $|\Omega| = \sqrt{y^2 + p^2 + r^2}$ and define:

$$P(\Omega) \propto \mathcal{N}_{|\Omega|,0,\sigma_{\Omega}}.$$
### 2.4 Bayesian inference

The state variables used in our model can be grouped in a vector $\xi^t = (\Theta^t, \Omega^t, A^t, C^t)$, and the sensory variables in a vector $S^t = (V^t, O^t, F^t)$. At time $t$, the purpose of Bayesian inference is to compute the probability distribution of $\xi^t$ given the sensory inputs received since the beginning of the simulation and given the initial distribution $P(\xi_0)$. The inference can be formulated in the following manner.

$$
P(\xi^t | S^t, ..., S_0) = \frac{1}{K} \sum_{\xi^{t-\delta t}} P(S^t | \xi^t).P(\Omega^t).P(A^t)$$

$$
\cdot P(C^t | C^{t-\delta t} \land \Omega^t \land \Omega^{t-\delta t}).P(\Theta^t | \Theta^{t-\delta t} \land \Omega^t)$$

$$
\cdot P(\xi^{t-\delta t} | S^{t-\delta t}, ..., S_0)
$$

The terms have the following meanings.

- $K$ is a normalization constant.
- $P(S^t | \xi^t) = P(V^t | C^t).P(O^t | \Omega^t).P(F^t | \Theta^t, A^t)$ is the probability distribution of sensory inputs given the state. The first two terms are Gaussian, and the third is a Dirac. It is equal to 1 if and only if equation 3 is verified.
- $P(\xi^t) = P(\Omega^t).P(A^t)$ is the a priori distribution.
- $P(C^t | C^{t-\delta t} \land \Omega^t \land \Omega^{t-\delta t})$ is a Dirac, equal to 1 if and only if equation 2 is verified.
- $P(\Theta^t | \Theta^{t-\delta t} \land \Omega^t)$ is a Dirac, equal to 1 if and only if equation 1 is verified.
\( P(\xi^t - \delta t | S^{t - \delta t} \land ... \land S_0) \) is the probability distribution computed at the last time step.

It should be noted that because \( P(\xi^t | \xi^{t-1}) \) is a Dirac, the complexity is greatly reduced. Indeed, for a given value of \( \xi^t - \delta t \), the space of possible \( \xi^t \) has three dimensions. If one knows \( \xi^t - \delta t \) and \( \Omega^t \), one can compute all the other variables in \( \xi^t \) using equations 2 and 3. It is also equivalent to know \( \xi^t - \delta t \) and \( C^t \), or \( \xi^t - \delta t \) and \( \Theta^t \), or \( \xi^t - \delta t \) and \( A^t \).

### 2.5 Particle filtering implementation

The major implementation issue is that the space of \( \xi^t - \delta t \) that must be scanned has nine dimensions: \( \Omega^{t-\delta t}, \Theta^{t-\delta t} \) and \( C^{t-\delta t} \) must be scanned but not \( A^{t-\delta t} \). To these, three dimensions must be added (the space of possible \( \xi^t \) given a value of \( \xi^{t-\delta t} \)). Because the model is non-linear and therefore cannot be solved exactly by a Kalman filter, we used the approximation method of particle filtering Maskell and Gordon [2002].

We use a set of \( N \) variables \( \xi^{t,i} \), also called “particles”. Each of these variables represents one possible head motion at time \( t \) and has an associated weight \( w^{i,t} \) that represents the probability of \( \xi^{t,i} \).

For each particle \( i \), the evolution of \( \xi^{t,i} \) over time is computed iteratively in the following manner. Starting from \( \xi^{t,i-\delta t} \), we draw a value \( \eta^{i,t} \) according to a Gaussian distribution. This allows us to compute a value \( \xi^{t,i} \) using the equations 1, 2 and 3. Then we obtain the term \( w^{i,t} = w^{i,t-\delta t} P(\xi^{t,i}) \). The probability of \( \xi^{t,i} \) is proportional to this term: it is \( w^{i,t} / \sum_j w^{j,t} \).

Moreover, a process of resampling is applied: at each iteration, a new set of \( N \) samples is drawn from the previous set. Each new particle is a copy of one of the previous ones, randomly chosen from the previous set. Each particle of the previous set will be chosen with probability \( w^i \) for each new particle. The new set replaces the previous one, and all weights are set to \( 1/N \). Resampling allows unlikely particles to be deleted while likely ones are duplicated, to avoid having the particles drift towards improbable states.

### 2.6 Simulation runs and results

To perform a simulation for a given experimental protocol, we compute the motion variables corresponding to the motion of the subject during the experiment. In the rest of this chapter, we refer to this as the “actual motion”. Then we compute the sensory inputs \( s^t \) generated by this motion. Finally we compute the probability distribution \( P(\xi^t | s^t \land ... \land s^0) \). We refer to this distribution as the motion estimate produced by the model. In a given simulation run, the sensory inputs are corrupted by Gaussian noise, and therefore they differ from one simulation run to another. Therefore, for a given experimental condition, we perform 30 simulation runs and pool the results.
We obtain the initial distribution $P(\xi^0)$ in the following manner. All simulations begin at $t = -5$ s, assuming that the position of the subject is known at $t = -5$ s and that the subject is immobile until $t = 0$. This simulates a subject who knows his or her position by using visual information before the light is turned off. During these five initial seconds of simulation, the uncertainty on $\xi$ reaches a stable level. The level of uncertainty at a given time influences the dynamic of the estimate.

For a given simulation, we present the results in the following way: for each relevant variable, we plot the distribution as an intensity plot. In most cases, this distribution is unimodal and bell-shaped. In these cases, it is meaningful to compute the mean of the probability distribution. We plot it and refer to it as the mean estimate.

Most experiments are performed in the dark to exclude visual information. In the corresponding simulations, we simply remove the visual information ($P(O^t|\Omega^t)$) from the model.

2.7 Parameters and canal plugging

We use the following parameters, which were manually set: $\sigma_V = 10^\circ$/s, $\sigma_o = 7^\circ$/s, $\sigma_\Omega = 40^\circ$/s and $\sigma_A = 3$ m/s$^2$.

The effect of canal plugging can be approximated by reducing the time constant of the canals (Rabbitt et al. [1999]). We described this by reducing the time constant of the canals to $T_c = 0.1$ s. This value was manually set to fit the VOR experiments in canal-plugged monkeys.

The time step was $\delta t = 0.1$ s. Each simulation run was performed with $N = 3000$ particles, and results were pooled over 30 simulation runs.

3 Results

3.1 Visuo-vestibular interactions

Simple visuo-vestibular interactions are studied using rotations around an earth-vertical axis that will not stimulate the otolithic system. Therefore, the results presented below can be understood without considering the otoliths. However, it should be kept in mind that during these experiments, the otoliths still indicate that the head is upright.

Another interesting point is that in these conditions, our model is linear. We use a stimulation velocity of $57^\circ$/s (i.e. 1 rad/s), but the response of our model to any other stimulation velocity would be proportional to the results shown below. Experimental studies in the monkey have shown that visuo-vestibular interactions are linear up to velocities of $60 - 100^\circ$/s. At higher velocities, saturation effects appear Dichgans [1977], Raphan et al. [1979].
Fig. 3. Vestibular and visual interactions, and consequences of canal plugging. In all figures, the probability distribution is displayed using a grey intensity scale. The dark line presents the mean of the probability distribution. The dotted line represents the stimulation velocity. a, c: velocity estimate during rotation in the dark at a constant velocity, with the normal model (a) and the canal-plugged model (c). The grey line represents the canal signal. b: estimate of head orientation in the horizontal plane. The mean of the distribution is not displayed. c, d: velocity estimate during a visual stimulation followed by a period of darkness, with the normal model (c) and the canal-plugged model (d).
Vestibular stimulation in the dark

We simulated a rotation at a constant velocity for 1 min followed by a stop. Simulation results are presented in Fig. 3a,b. As described above, the canal signal follows a high-pass filter dynamic. It is equal to the rotation velocity immediately at the onset of rotation, then it decays exponentially with a time constant of 4 s. At the end of the rotation, the deceleration produces a “negative” signal.

The velocity estimate follows a similar dynamic, with a longer time constant (28 s). The prolongation of the time constant indicates that the model integrates vestibular information over time. The dynamic of the velocity estimate produced by the model can be understood by simple probabilistic considerations. The Bayesian model includes a model of canal dynamics. If the canals were not noisy, the velocity estimate would always be equal to the stimulation velocity, i.e. $P(\Omega_t \wedge \Omega_{t-\delta t} \wedge ... \wedge \Omega_0 | \nu_t \wedge \nu_{t-\delta t} \wedge ... \wedge \nu_0) = 1$ if and only if $\Omega_t = \Omega_{t-\delta t} = ... = \Omega_0 = 57^\circ/s$. However, the presence of noise in the vestibular canals creates an uncertainty on the rotation velocity. In this condition, the a priori distribution $P(\Omega_t)$ makes low values of $\Omega_t$ more probable. This creates a bias that drives the motion estimate towards zero.

At the end of the rotation, the canal signal and the velocity estimate have decayed close to zero. In this situation, the deceleration reported by the canals is interpreted as an acceleration in a direction opposite to the initial rotation.

We show the estimate of head orientation in the horizontal plane in Fig. 3b. This estimate is the integral of head velocity over time. As time passes, this estimate becomes more and more scattered. Indeed, in this plane, there is no information about head orientation other than head angular velocity. Therefore, uncertainty on angular velocity accumulates over time.

Finally, we show the results obtained with the model adapted to plugged canals (Fig. 3c). With this model, the time constant of the velocity estimate is smaller (1 s). Angelaki et al. [1996] measured a corresponding reduction of the VOR time constant after canal plugging.

Visual stimulation

We simulated a visual stimulation at constant velocity ($57^\circ/s$) while the subject is immobile (Fig. 3d). The velocity estimate follows the typical dynamic of OKN measured in monkeys Raphan et al. [1977]. At the beginning of the stimulation, the estimated velocity rises immediately to $35^\circ/s$, then rises exponentially to a plateau ($55^\circ/s$). The time constant of this increase is 5 s. At the end of the stimulation, the estimated velocity decreases exponentially with a time constant of 25 s. This slow decrease is similar to the optokinetic after-nystagmus (OKAN).

When the simulation is performed with the model adapted to canal plugging, the dynamics are modified (Fig. 3e). The estimated velocity increases almost immediately at the beginning of the stimulation and drops quickly
at the end (with a time constant of 1 s). Similar results were observed by Angelaki et al. [1996].

We analyse these results as well as their implications for the understanding of visuo-vestibular interactions in the discussion.

### 3.2 Somatogravic effect and interaction with the canals

We simulated a constant acceleration of the head at 3 m/s². Results are presented in Fig. 4a. Immediately after the beginning of the acceleration, the acceleration estimate is 3 m/s². In a few seconds, this estimate decays, whereas an estimate of roll builds up, reproducing the somatogravic effect. The buildup of roll estimate is exponential, with a time constant of 1.3 s. This is in the order of the time constant reported by Telford et al. [1997], Paige and Seidman [1999] and the results of Angelaki [1998].

We illustrated canal–otolith interactions by simulating lateral oscillations and roll oscillations at 0.5 Hz, to reproduce the results of Angelaki [1998]. The peak acceleration was 3 m/s² for the lateral oscillations, whereas the peak roll tilt was 18°, resulting in almost identical otolith stimulations. Results are shown in Fig. 4b,c. At 0.5 Hz, the somatogravic effect is weak, and indeed the acceleration estimate is close to the actual acceleration during the translation simulation whereas the roll estimate is small. In contrast, during the roll oscillation simulation, the roll estimate is close to the actual roll whereas the acceleration estimate is weak.

We repeated these simulations using the model adapted to plugged canals. During a constant acceleration, the time constant of the simulated somatogravic effect is reduced to 0.5 s. Moreover, simulation results during lateral oscillations and roll oscillations are the same. Therefore, our model reproduces the disappearance of the ability to discriminate between translation and tilt after canal plugging (Angelaki et al. [2002]).

### 3.3 Off-vertical-axis rotation (OVAR)

During OVAR, the head is constantly reorienting relative to gravity, in contrast with rotations around a vertical axis. As a consequence, monkeys exhibit a horizontal VOR that lasts indefinitely during OVAR. However, the velocity of this VOR is slightly lower than the rotation velocity. For instance, Angelaki measured a VOR velocity of 45°/s during OVAR rotation at 57°/s.

During OVAR, the canal signal fades out as it does during rotation around a vertical axis. Therefore, the process by which the brain determines that there is a rotation during OVAR could seem independent of the canals. However, experiments with canal-plugged monkeys have given a surprising result: the velocity of the horizontal VOR during OVAR is reduced. At the same time, a torsional and a vertical VOR appear, with a sinusoidally modulated velocity Raphan and Cohen [1985], Angelaki et al. [2000].
Fig. 4. Somatogravic effect and canal–otolith interactions. Lateral acceleration and roll estimate during a constant lateral acceleration (a, d), during lateral oscillations at 0.5 Hz (b, e) and during roll oscillations (c, f), with the normal model (a–c) and with plugged canals (d–f).
Fig. 5. Off-Vertical Axis Rotation (OVAR). Yaw and roll velocity estimates during OVAR with a rotation velocity of $57^\circ$ /s and a tilt angle of $30^\circ$. a, b: results obtained with the normal model. c, d: results obtained with the plugged-canal model.
We simulated OVAR with a tilt angle of 30° and a rotation velocity of 57°/s, to match the parameters used in Angelaki et al. [2000]. When the simulation is run using the normal model, the estimate of yaw velocity (Fig. 5a) quickly stabilizes to a mean value of 38°/s, close to the value of 45°/s observed in Angelaki et al. [2000]. The estimate of roll velocity is close to zero, although a small sinusoidal modulation can be observed (Fig. 5b).

When the simulation is run with the plugged-canal model, the estimate of yaw velocity stabilizes at 6°/s (Fig. 5c). This is similar to the value of 14°/s measured in this condition Angelaki et al. [2000]. Furthermore, strong oscillations of the roll velocity estimate appear (Fig. 5d). The pitch velocity estimate behaves in a similar manner.

These results illustrate the ability of our model to capture the contribution of canal and otolith signals during OVAR. In the discussion, we go further in the analysis of these results and propose an explanation of the consequences of canal plugging.

4 Discussion

Velocity storage

OKAN is considered to be a manifestation of the process of velocity storage. After canal plugging, OKAN as well as other experimental protocols Paige [1983] suggest there is a strong reduction in velocity storage. Our model reproduces these results. Furthermore, the use of the Bayesian formalism allows us to interpret this process in relation to the canal information.

We also performed simulations with a variation of the model in which the canals are removed (not shown). The results are close to the results obtained with plugged canals. At the beginning of the stimulation, the velocity estimate rises instantaneously and drops immediately to zero at the end of the stimulation. At the end of the visual stimulation, the only information about head velocity is the \textit{a priori}, and the velocity estimate drops immediately to zero. Similar effects were observed experimentally Raphan et al. [1977].

The results obtained with the normal model (which includes the canals) is similar, but the dynamic is low-pass filtered. Indeed, during visual stimulation, the canal signal is zero, indicating there is no high-frequency acceleration. This results in a slower increase in the estimated head velocity and a slower decay at the end of the stimulation. The latter effect reproduced the OKAN, which is a manifestation of velocity storage.

Although the link between canal information and velocity storage is known, our model allows us to formulate this link in a mathematical (Bayesian) framework and to perform quantitatively accurate simulations using this framework.
Analysis of the dynamic of visuo-vestibular interactions

Another interesting characteristic of the Bayesian framework is that, in linear situations, the results can be predicted in a simple way. For instance, we can predict the value of the velocity estimate when it reaches its plateau during visual stimulation, when the information about angular velocity is the visual information (which indicates a velocity $\omega_\alpha = 57^\circ/s$ with a standard deviation $\sigma_\alpha = 7^\circ/s$) and the a priori (which indicates a velocity $\omega_\pi = 0^\circ/s$ with a standard deviation $\sigma_\pi = 40^\circ/s$). The result of the fusion of these data can be computed by the following formula.

$$\omega_{O,\pi} = w_\omega \omega_\alpha + w_\pi \omega_\pi = w_\alpha 57^\circ/s \text{ with } w_\alpha = \frac{1}{\sigma_\alpha^2}$$

This gives $\omega_{O,\pi} = 55^\circ/s$, in accordance with the simulation results. The standard deviation around this mean estimate is $\sigma_{O,\pi} = (\sigma_\alpha^2 \sigma_\pi^2 / (\sigma_\alpha^2 + \sigma_\pi^2))^{1/2} = 6.7^\circ/s$.

The initial quick rise at the beginning of visual stimulation with the normal model can also be predicted. This rise is almost instantaneous, which is equivalent to an (almost infinitely) high-frequency acceleration, for which the gain of the canals is 1. The canals indicate that this variation does not occur; therefore, they report a velocity of $\omega_V = 0$ with a standard deviation of $10^\circ/s$. The visual information reports a velocity of $57^\circ/s$ and the a priori a velocity of 0, as previously. The result of the fusion of these three values can be computed by fusing visual information and the a priori as previously, and then computing as follows.

$$\omega_{O,\pi,V} = w_{O,\pi} \omega_{O,\pi} + w_V \omega_V \text{ with } w_{O,\pi} = \frac{1}{\sigma_{O,\pi}^2}$$

This gives $\omega_{O,\pi,V} = 37^\circ/s$, close to the value of $35^\circ/s$ we obtained in our simulation.

The exponential rise and decay of estimated head velocity are more difficult to predict by simple algebraic computations. The time constant of the increase (5 s) is shorter than the time constant of the decreases (25 s), because visual information is a “strong” cue in favour of a velocity of $57^\circ/s$ ($\sigma_\alpha = 7^\circ/s$), whereas the a priori is a “weak” cue ($\sigma_\pi = 40^\circ/s$).

4.1 Somatogravic effect

The results obtained with oscillatory translation or roll illustrate the ability of the Bayesian model to use canal information to discriminate between acceleration and tilt. This ability comes from the use of an internal model of the gravito-inertial force (equation 3). Previous models that include an internal representation of gravity are able to predict this effect.
The ability of our model to reproduce the somatogravic effect is a consequence of the a priori in favour of low accelerations. During sustained acceleration (Fig. 4a,b) or low-frequency lateral oscillations (e.g. 0.01 Hz), the canals cannot provide accurate roll information. In this situation, the Bayesian model favours a roll estimate that minimizes the acceleration. The notion that the somatogravic effect is an adaptation to the rarity of sustained acceleration in everyday life has been proposed previously Guedry [1974], Paige and Seidman [1999]. The Bayesian framework allows us to formalize this mathematically.

4.2 OV AR

Our model reproduces the experimental results during OV AR. Furthermore, we aimed at gaining a better understanding of how it reproduces them. By analysing our model, we built a theory of the way the brain interprets sensory signals during OV AR. In this part, we will briefly present it in the context of the two simulations presented above.

During OV AR, otolith information indicates that the head is constantly reorienting relative to gravity. The Bayesian model attempts to find a motion estimate in which the estimated orientation of the head is always close to the one reported by the otoliths, to minimize the estimate of linear acceleration. This process is similar to the somatogravic effect described above. In our simulations, the estimate of head orientation relative to gravity is indeed always close to its actual value. In brief, the Bayesian model is looking for an angular motion that “accounts” for the reorientation of the head relative to gravity.

In the simulation with normal canals, the motion estimate consists (to a first approximation) of a rotation around Z. This motion estimate is close to the actual motion and does indeed correspond to the reorientation of the head. We will call it “estimate A”.

In the simulation with the plugged canals, the motion estimate (B) is a combination of a rotation around the Z axis at a constant velocity, together with oscillations around the X and Y axes (i.e. in roll and pitch). This combination also results in a head orientation relative to gravity close to the one reported by the otolith: it is another way to account for the reorientation.

The two motion estimates (A and B) account for the otolithic input equally well; however, there are two differences between them.

- When we compute the total angular velocity $|\Omega|$, we find that it is lower in B ($14^\circ$/s) than in A ($38^\circ$/s). Therefore, estimate B is more probable according to a a priori in favour of low acceleration.
- Motion estimate B includes oscillations in pitch and roll that are in mismatch with the canal signal. During OV AR, the canals are not activated. This fits with any rotation at a constant velocity (like rotation estimate A) but not with these oscillations.

In summary, motion estimate B corresponds to a lower angular velocity than A, but it does not fit with the canal signal. In the simulation with normal
4.3 Other results and predictions of the model

Our model was also able to simulate more complex experiments. In Laurens and Droulez [2007], we applied it to motion perception experiments in humans and reproduced motion perception during centrifugation.

As presented above, the somatogravic effect is linked to canal information. A consequence of this is that, after canal plugging, the somatogravic effect should be quicker. Angelaki did indeed measure an enhancement of this effect at 0.5 Hz Angelaki et al. [2002]. We were unable to reproduce this effect at 0.5 Hz. Indeed, at this frequency and using the parameter $T_c = 0.1$ s for describing plugged canals, the difference between normal and plugged canals is small. However, our model predicts this enhancement at lower frequencies.

Another effect can be predicted: during visual stimulation, if the visual input deteriorates, the initial rise of the OKN should be lower. A deteriorated visual input could be created by using an optokinetic stimulation made of moving dots. By adding a Gaussian noise to the motion of the dots, or adding some incoherently moving dots, one could increase the uncertainty of the visual information (e.g. increase $\sigma_o$). This should result in a decrease of the initial OKN velocity as well as the plateau velocity.

4.4 Application to robotics

Many robots include inertial motion sensors for providing accurate high-frequency information. However, artificial accelerometers are subject to the gravito-inertial ambiguity. This is not an issue on robots that use wheels and navigate only on horizontal ground. However, legged robots or outdoor robots must discriminate between translation and tilt.

The rotatory information coming from the canals can solve this problem during high-frequency motion. In a robot, rotation information coming from gyroscopes can play the same role. This process is implemented in modern aircraft, and Mayne pointed out the parallel between canal–otolith information and artificial inertial navigation systems in Mayne [1974]. Still, our model provides new methods in this respect. First, the use of Bayesian inference provides optimal information that takes into account the accuracy of the sensors. Furthermore, the Bayesian framework allows us to add other information to the inertial system. For instance, our model uses an $a\ priori$ on linear acceleration as a way of discriminating between translation and tilt during low-frequency motion. In a robot, this $a\ priori$ could be replaced by uncertain acceleration information reconstructed from the motor commands. In a rolling robot, odometry typically fails if the wheels slip. Our model would easily detect this event.
Visual information coming from a video camera is usually complex to process. By fusing inertial and motor information in an optimal way, our model could provide accurate self-motion information that could greatly reduce the complexity of visual processing.

Application to robotics raises the concern of computation time. The particle filter inference program is not parsimonious. With a recent laptop, the inference can be run in real time, at 10 Hz and with 3000 particles, but it uses more than half of the available computing power. However, other implementations of the model, for instance using an extended Kalman filter, could require much less computing power.

5 Conclusion

One of the strengths of our model is the limited number of assumptions and parameters it incorporates. This makes it a simple and general tool for simulating and analysing visuo-vestibular interactions. Its ability to reproduce the consequence of canal plugging by lowering the time constant of the canals, without affecting any other parameter, is an illustration of the efficiency of the Bayesian approach. Furthermore, our model can be simplified to simulate specific stimulations. We showed how the initial rise of OKN can be predicted by a simple probabilistic computation. Visuo-vestibular interactions during rotations around an earth-vertical axis are linear: in this case, our model can be reimplemented by a Kalman filter.

A general result predicted by our model is that during a high-frequency stimulation, the motion estimate is always close to the actual motion. Indeed, at these frequencies the canal information is reliable enough to compute head orientation accurately and to solve the gravito-inertial ambiguity. The visual stimulation experiment is an exception, but it is a situation in which vestibular and visual information are not coherent.

Our model is based on the assumption that the brain uses an internal model of motion, together with the principle of Bayesian inference. Accordingly, part of the results (such as the OVAR) we presented can be reproduced by other models based on the internal model hypothesis. Still, other parts of the results, such as the dynamics of OKN or the somatogravic effect, are governed by the noise on sensors. Therefore, we propose a mathematical interpretation of these effects that is based on Bayesian inference.

References


Bayesian modelling of perception of structure from motion

Francis Colas\textsuperscript{1}, Pierre Bessière\textsuperscript{2}, Jacques Droulez\textsuperscript{3}, and Mark Wexler\textsuperscript{3}

\textsuperscript{1} INRIA Rhônes-Alpes - E-Motion
\textsuperscript{2} CNRS - Grenoble Université
\textsuperscript{3} Collège de France - LPPA

We use multiple sensory modalities to perceive our environment. One of these is optic flow, the displacement and deformation of the image on the retina. It is generally caused by a relative motion between an observer and the objects in the visual scene. As optic flow depends largely on three-dimensional (3D) shapes and motions, it can be used to extract structure from motion (the \textit{sfm} problem). Motion parallax and the kinetic depth effect are special cases of this phenomenon, noticed by Von Helmholtz [1867], and experimentally quantified by Wallach and O’Connell [1953].

Extraction of shape from motion is a difficult issue for two main reasons: it is an ill-posed and inverse problem. Many different combinations of shape and motion can lead to the same optic flow, so reconstructing a shape from the optic flow cannot lead to a unique result. In this aspect, \textit{sfm} is an ill-posed problem. Furthermore, geometry and optics can lead to the expression of optic flow given 3D structure and motion. However, \textit{sfm} is interested in the opposite; that is, \textit{sfm} is an inverse problem.

Previous approaches to the \textit{sfm} problem rely mostly on the \textit{rigidity assumption}, the hypothesis that optic flow is caused by the motion of a rigid object. Under this assumption, the number of degrees of freedom of the motion is greatly reduced, and it has been shown that little optic flow information is required to recover the motion and the structure of the object [Ullman, 1979, Mayhew and Longuet-Higgins, 1982]. This assumption is supported by human performance in some psychophysical experiments Wallach and O’Connell [1953], Koenderik [1986]. Some more recent models, relying on local velocity information rather than on the full optic flow field, are consistent with human performance [Todd and Bressan, 1990, Todd and Norman, 1991].

Extensive studies have investigated the perception by an immobile observer of optic flow caused by a mobile object. However, \textit{sfm} can also occur with a static object for an observer in motion [Rogers and Graham, 1979]. Until recently, it was believed that for the same optic flow, perception is the same with object motion or with subject motion [Wallach et al., 1974, Rogers...
and Graham, 1979]. However, it has since been shown that the observer’s movements influence the perceived 3D shape and motion [Rogers and Rogers, 1992, Dijkstra et al., 1995, Wexler et al., 2001b].

Therefore, the stationarity assumption has been introduced to account for the way that self-motion changes perception [Wexler et al., 2001b,a, Wexler, 2003]. The stationarity assumption states that the visual system prefers the solution having minimal motion in an observer-independent, allocentric reference frame. Therefore, the stationarity assumption is the minimization of absolute motion, whereas the rigidity assumption can be seen as the minimization of relative motion between points of an object. Neither of these assumptions can explain human performance in the sfm task, and until now, no coherent model has been proposed to integrate these two assumptions.

In this chapter, we propose a generic Bayesian model that integrates the stationarity and rigidity assumptions for the perception of 3D planar surfaces from optic flow. The model accounts for the sfm performance of moving and stationary observers, as well as a number of other results reported in the literature. We begin with a generic model of sfm, then we give its instantiation for the perception of a plane. We finally compare the results of this model with six experiments found in the literature.

1 Model

1.1 Generic model

The generic Bayesian model that we propose is a model of what an observer can deduce from the limited information received from optic flow. To cope with this limited information, this model uses probabilities to represent and handle the uncertainties that it faces. We construct our model according to the hypotheses evoked above. The first two are the stationarity (H1) and rigidity (H2) assumptions. Our model also assumes that the structure of the object is independent of both its motion and the motion of the observer (H3). We follow the Bayesian programming framework to specify this model [Lebeltel et al., 2004].

From relevant information to variables

For the general case of perception of an object by optic flow, we propose a model that takes into account: (i) the observed optic flow (denoted $\Phi$); (ii) the 3D structure of the object (denoted $\Theta$); (iii) the motion of the object (denoted $X$) in the observer’s reference frame; (iv) the motion of the observer in the allocentric reference frame (denoted $M$); and (v) the context of observation (denoted $\Lambda$).
From dependencies to decomposition

At the core of a Bayesian model lies the joint probability distribution over all its variables. This joint distribution is the expression of the hypotheses of a model. The structural part in the specification of the joint distribution summarizes the dependencies and independencies between the variables. This structure is called decomposition.

Hypothesis H1 is the rigidity assumption, which states that the observed optic flow is most likely that of a rigid object. As a consequence, the optic flow depends on the relative motion, the structure of the object, and the conditions of observation, but is independent of self-motion. This corresponds to the following mathematical simplification:

\[ P(\Phi \mid \Theta \land M \land X \land \Lambda) = P(\Phi \mid \Theta \land X \land \Lambda). \] (1)

The stationarity assumption (H2) states that the object motion is most likely to be small in the allocentric reference frame. Therefore, the relative motion depends on self-motion. We use Bayes’ rule to write:

\[ P(M \land X) = P(M) P(X \mid M). \] (2)

Hypothesis H3 states that the structure of the object is independent of both the relative motion of the object and self-motion. This translates as a product of independent factors in the decomposition:

\[ P(\Theta \land M \land X) = P(\Theta) P(M \land X). \] (3)

We also make the assumption that the conditions of observations are independent of the position of the plane as well as the motion of the object and self-motion. This is expressed as:

\[ P(\Theta \land M \land X \land \Lambda) = P(\Theta \land M \land X) P(\Lambda). \] (4)

Finally, using Bayes’ rule, we can write:

\[ P(\Theta \land M \land X \land \Lambda \land \Phi) = P(\Theta \land M \land X \land \Lambda) P(\Phi \mid \Theta \land M \land X \land \Lambda). \] (5)

Combining equations 5, 1, 3, 4, and 2, we obtain the decomposition shown in equation 6.

\[ P(\Theta \land M \land X \land \Lambda \land \Phi) = P(\Theta) P(M) P(\Lambda) \times P(X \mid M) \times P(\Phi \mid \Theta \land X \land \Lambda). \] (6)

This decomposition is the structural expression of our assumptions.
Physical and physiological laws → distributions

To derive a usable expression for the joint distribution, we must specify each of the factors in the above decomposition. The first factor, \( P(\Theta) \), is the prior on the structure of the object. It represents what our model expects before any observation. It can be an uninformative prior or it can reflect some bias in the perception. The exact parametrical form depends on the actual experiment. In the same way, the second factor \( P(M) \) represents the expectation by an observer of her or his own motion. If we assume that the model has an exact knowledge of its self-motion (as will be the case later in this chapter), this probability distribution is simplified in the final inference and thus can be left unspecified. The same holds for the probability distribution over the conditions of observation \( P(\Lambda) \). The inference will simplify this factor for any question in which the conditions of observations are exactly known.

The fourth factor, \( P(X | M) \), specifies the relative motion expected from a given self-motion. According to stationarity, the object is more likely to undergo a smaller absolute motion. Therefore, the most probable relative motion should be defined as the opposite of self-motion. The actual parametrical form varies once again with the experiment, but a general expression could be proportional to the exponential of the opposite of kinetic energy (Gibbs distribution). In some cases (in an appropriate parameter space), this means a Gaussian distribution.

Finally, the last factor in decomposition 6 is the distribution on optic flow, given the structure of the object, the relative motion between the object and the observer, and the conditions of observation: \( P(\Phi | \Theta \land X) \). Following the rigidity assumption, this distribution states that the most probable optic flow is the theoretical flow of the object in this particular configuration, given this particular motion.

Formalized questions

A probabilistic question is a distribution over some variables of the model, possibly given the knowledge of the values of other variables. With a completely specified joint distribution, the answers to such questions can be mechanically inferred with both Bayes’ and marginalization rules.

This model is designed for the perception of structure from optic flow. The first question that we can ask is the probability of the object structure or shape given the optic flow, the self-motion, and the conditions of observation, written as \( P(\Theta \mid \varphi \land m \land \lambda) \).\(^4\) This question is answered by the following expression, given by Bayesian inference.

\[
P(\Theta \mid \varphi \land m \land \lambda) = \frac{P(\Theta \mid \varphi \land m \land \lambda) \cdot \sum_{x \in X} P(x \mid m) P(\varphi \mid \Theta \land x \land \lambda)}{\sum_{\theta \in \Theta} P(\theta) \cdot \sum_{x \in X} P(x \mid m) P(\varphi \mid \Theta \land x \land \lambda)}.\quad (7)
\]

\(^4\) We use an upper case letter for a variable and lower case for the instantiation of a variable with a particular value.
This is the computation that we have used to obtain the results shown later. Given our hypotheses, this distribution represents our knowledge about the structure of the object (including its relative position with respect to the observer) that one can infer from the observation of optic flow and self-motion.

Furthermore, with this model, one can ask other questions. For example, one may be interested in the estimation of self-motion from optic flow: \( P(M \mid \varphi) \). This question can be used to studyvection, in which optic flow induces the sensation of self-motion. In the same manner, one can also study the direction of perceived self-motion, called heading. With a single Bayesian model, different issues can be investigated.

1.2 Perception of a plane

We can adapt this model to account for given situations. In this chapter, we consider a set of experiments dealing with the perception of a planar object through its optic flow. The observation is monocular, and the participant is sometimes allowed to move along some degrees of freedom while maintaining fixation. The observation can take place in either a small or large field of view.

With this information, we instantiate the template model following the Bayesian programming framework.

Variables

The first variable of the model is the optic flow. We must choose a representation space adapted to the specifics of the experiments. When observing a plane, the optic flow is uniquely determined by the derivatives of the image velocity with respect to the image coordinates, up to the second order [Longuet-Higgins, 1984]. Therefore the chosen semantics and domain of \( \Phi \) are these derivatives. We can separate the optic flow into multiple components depending on the order of derivation: \( \Phi = \Phi^0 \land \Phi^1 \land \Phi^2 \).

As we consider a plane, the 3D structure of the object is restricted to its position. The participant maintains fixation on a given point of the plane. This way, only two parameters are required to specify the structure \( \Theta \). The extrinsic orientation of a plane in 3D space is often parameterized by two angles, slant and tilt (see Fig. 1(a)). Slant is the angle, in 3D space, between the plane’s normal vector and the normal of the fronto-parallel plane. Tilt is the angle, in the fronto-parallel plane, of the projection of the plane’s normal. For practical reasons in the computations, we use the depth gradients along both the transversal and vertical axes (respectively \( X \) and \( T \)), as depicted in Fig. 1(b).

The motion \( X \) of the object in the reference frame of the observer is defined by its rotation \( \Omega \) and translation \( T \) components along the transversal, vertical and sagittal axes. Likewise, self-motion \( M \) is a set of translation and rotation velocities of the eye of the observer. Finally, the context of observation \( A \) reduces, for the experiments considered, to the size of the field of view, reduced to two extreme cases: small and large fields of view.
Fig. 1. Illustration of the parameters of a plane. The object plane is shown in blue; the \((x, y, z)\) reference frame is shown in red, and the observer (shown by its eye) is located at \((0, 0, d > 0)\). (a) Illustration of the tilt \(\tau\) and slant \(\sigma\) of a plane. The slant is the angle between the object plane and the fronto-parallel plane. The tilt is the angle between the vertical axis and the intersection of the object plane and the fronto-parallel plane. (b) Illustration of parameters \((X, Y)\) of a plane. In green, we show the intersections of the object plane with both horizontal and sagittal planes. The respective slopes of these lines are the \(X\) and \(Y\) parameters of the plane.

**Decomposition**

The instantiated model inherits the conditional independencies of the generic one. However, we can add some new simplifications in the joint distribution, depending on our knowledge of our experiments. In this case, the choice of parameterization of the optic flow allows us to make each component independent of the others, conditionally to knowledge of the relative position, relative motion and conditions of observation:

\[
P(\Phi \mid \Theta \land X \land A) = P(\Phi^0 \mid \Theta \land X \land A) \times P(\Phi^1 \mid \Theta \land X \land A) \times P(\Phi^2 \mid \Theta \land X \land A).
\]

Furthermore, we can state that only the perception of second-order optic flow depends on the size of the field of vision and that constant optic flow is generated only by the relative translation in the fronto-parallel plane. Finally our operative decomposition is:

\[
P(\Theta \land M \land X \land A \land \Phi) = P(\Theta) P(M) P(A) \times P(X \mid M) \times P(\Phi^0 \mid T) \times P(\Phi^1 \mid \Theta \land X) \times P(\Phi^2 \mid \Theta \land X \land A).
\]
Distributions

As in the general case, we specify an exact parametrical form for each factor of our decomposition. The difference resides in our knowledge of our particular experiments.

We chose the prior on the position of the plane to be uninformative. That is, we state that each position of the plane is equally probable. In terms of probability distributions, this means that it is invariant by rotation around the fixation point. For a given position \( \theta = (\chi, \nu) \) of the plane, the probability density becomes:

\[
P(\chi \wedge \nu) = \frac{1}{2\pi} \left(1 + \chi^2 + \nu^2\right)^{-\frac{3}{2}}.
\]

As in the general model, the factors \( P(M) \) and \( P(\Lambda) \) can be left unspecified.

For \( P(X \mid M) \), the expression of stationarity, we chose a Gibbs distribution with kinetic energy computed in the global reference frame. As \( X \) is expressed as the linear and angular velocities with respect to each axis, this means that \( P(X \mid M) \) is a product of Gaussian distributions.

Finally, rigidity is expressed by the factors \( P(\Phi^0 \mid T) \), \( P(\Phi^1 \mid \Theta \wedge X) \), and \( P(\Phi^2 \mid \Theta \wedge X \wedge \Lambda) \). These distributions are chosen as Gaussian around the exact values of the optic flow computed by the theoretical equations (detailed in appendix A). The influence of the size of the field of view in the distribution of the second-order optic flow is in the covariance matrix (see the implementation for details).

Question

This is a model of the perception of a plane from optic flow. The question that we ask, to compare our model with the experimental results, is: what knowledge about the relative position of the plane can be deduced from the observation of optic flow, self-motion and the conditions of observation? As noted above, this becomes, in probabilistic terms:

\[
P(\Theta \mid \varphi \wedge m \wedge \lambda).
\]

With a more specific decomposition than the generic model, we can write the answer to this question with greater precision:

\[
P(\Theta \mid \varphi \wedge m \wedge \lambda) \propto P(\Theta) \sum_{t \in T} P(\varphi^0 \mid t) \sum_{\omega \in \Omega} P(\omega \mid m) P(\varphi^1 \mid \Theta X) P(\varphi^2 \mid \Theta \wedge X \wedge \Lambda) \tag{9}
\]

Implementation

For the instantiated model, a number of parameters are necessary for the calculations.
Table 1. Domains of the variables.

<table>
<thead>
<tr>
<th>Variable</th>
<th>symbol</th>
<th>min</th>
<th>max</th>
<th>values by dimension</th>
<th>dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Depth gradient</td>
<td>( \Theta )</td>
<td>-4.125</td>
<td>4.125</td>
<td>33</td>
<td>2</td>
</tr>
<tr>
<td>Angular velocity</td>
<td>( \Omega )</td>
<td>-1.375 rad.s(^{-1})</td>
<td>1.375 rad.s(^{-1})</td>
<td>11</td>
<td>3</td>
</tr>
<tr>
<td>Linear velocity</td>
<td>( T )</td>
<td>-1.375 m.s(^{-1})</td>
<td>1.375 m.s(^{-1})</td>
<td>11</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 2. Covariance matrices for each factor of the joint distribution. From top to bottom: distribution over the relative translation, relative rotation, order-0 optic flow, order-1 optic flow, order-2 optic flow in a small field of vision and order-2 optic flow in a large field of vision. \( Id_{n\times p} \) stands for the identity matrix of size \( n \) by \( p \).

<table>
<thead>
<tr>
<th>Distribution parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \sigma_T = 0.3 \times Id_{3\times3} ) in ( m.s^{-1} )</td>
</tr>
<tr>
<td>( \sigma_\Omega = 1.2 \times Id_{3\times3} ) in ( rad.s^{-1} )</td>
</tr>
<tr>
<td>( \sigma_{\phi^0} = 1.0 \times Id_{2\times2} ) in ( m.s^{-1} )</td>
</tr>
<tr>
<td>( \sigma_{\phi^1} = 0.025 \times Id_{4\times4} ) in ( s^{-1} )</td>
</tr>
<tr>
<td>( \sigma_{\phi^2</td>
</tr>
<tr>
<td>( \sigma_{\phi^2</td>
</tr>
</tbody>
</table>

With the notable exception of Gaussian distributions in some particular conditions (linearity for instance, which is not the case here; see the appendix), Bayesian inference is usually done with discretized or sampled variables. Table 1 gives the details of the ranges (minimum, maximum and number of samples in between) and dimensionality of each component of \( \Theta \) (top row), of the relative rotation (middle row), and of the relative translation (bottom row). Other variables do not require discretization as their values are known for the inference.

Moreover, some of the distributions in our decomposition involve parameters; for instance, the Gaussians on relative motion and optic flow. Table 2 presents their respective covariance matrices. We notice that each one is diagonal and that the covariance on optic flow is greater in a small field of view for second-order optic flow than in a large field. Indeed, because second-order optic flow is quadratic in the distance from the fixation point, it is harder to perceive in a small field of vision than in a large field.

The results presented in the following section were all computed with this single set of parameters using the ProBT inference engine for the calculations [Lebeltel et al., 2004].

2 Results

In this section, we present six psychophysics experiments concerned with the monocular perception of a rotating planar patch with a neutral or non-informative texture. Therefore, the only cue for the plane’s relative position
was its motion via the optic flow. These experiments were previously reported by different teams and admit some variations around a common set-up, involving the motion of the observer’s head or eyes, or the plane, or both, as well as the size of the displayed stimulus. This is in accordance with the choice of the variables of our model. For each experiment, we present the results of the Bayesian model and compare its performance with the participants’.

2.1 Protocol

All these experiments share the same basic conditions of experimentation. They were carried out by different teams, and the variations mostly concern the testing of the various assumptions. Here, we present the common features, and we will detail the differences when necessary.

The stimulus is an animation showing a 3D slanted plane in rotation. This animation is displayed on a monitor or on a screen. The plane is figured by a random-dot texture, considered to be uninformative concerning the slant and tilt of the plane (see Fig. 2). The size of the display depends on the condition of observation. It can be either a small field of vision (inside a cone of 5 to 10° of semi-angle) or a large field of vision (around 60°).

Fig. 2. Example of a frame of the animation presented to the participants. The display is composed of white dots of the object plane, projected on a black background. As the plane moves in 3D, the dots move according to their position on the object plane. The dots are chosen so that the distribution of their projection is invariant with respect to a 2D planar rotation.

The observation is monocular: the non-dominant eye of the participant is masked. The participants are also fitted with sensors so the experimenter can measure the position of their head or eye. Therefore, their motion can be used to check whether it is not too important (if they are asked not to move) and to change the projection of the dots of the object plane on the screen according to their changing point of view. The stimulus is displayed for less than 1 s.

The participants are asked to judge the slant or the tilt of the plane. To do so, after the presentation of the stimulus, they must align a probe (like a planar grid) using an input device (a joystick for example) with the mean position of the perceived plane.
Fig. 3. Some ambiguities in first-order optic flow that have been used in the studies cited. (a) An example of an optic flow field that presents a number of ambiguities: all configurations shown in this figure lead to this flow. (b) These two configurations, which differ by simultaneous reversals of relative depth and 3D motion, both yield the optic flow shown in (a). This ambiguity is called depth reversal. (c) Depth reversals can also occur for moving observers. These two configurations have the same relative motion between object and observer as in (b) and therefore yield the same optic flow. However, one solution is stationary in an allocentric or observer-independent reference frame, while the other solution undergoes a rotation in this frame, twice as fast as the observer’s motion. (d) The same ambiguity when the observer tracks a moving surface with the eyes. One solution undergoes a translation only, while the other undergoes the same translation and a rotation. (e) Ambiguity between slant and rotation speed: a larger slant coupled with a slower rotation may give the same optic flow as a lower slant together with a faster rotation.

2.2 Depth reversal

Depth reversal is a well-known effect in 3D vision: many depth cues are ambiguous as to the sign of relative depth (cf. the Necker cube). In SFM, the simplest instance of this ambiguity is the observation of a rotating plane through a small opening. In this case, there is ambiguity regarding the tilt and direction of rotation, as illustrated in Fig. 3(b). However, it has been shown
[Dijkstra et al., 1995] that this ambiguity does not occur with a large field of vision. We investigate this simple effect as a first example of our model.

The experiment that we use as a reference was described by Cornilleau-Pérès et al. [2002]. In this particular experiment, the participants are asked not to move. The optic flow is induced by the plane, in rotation around an axis in the fronto-parallel plane.

Cornilleau-Pérès et al. [2002] report the results in terms of the rate of tilt reversals. A tilt reversal is defined to occur when the absolute error in the estimation of the tilt angle is greater than $90^\circ$. The reversal rate can be considered as a measure of the ambiguity, as illustrated in Fig. 3(b). The middle column of Table 3 presents the results of the experiment, and we observe that the reversal rate drops from close to its maximal value (50%) in a small field of vision to below 5% in a large field of vision.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Experiment</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small field</td>
<td>48.8%</td>
<td>44.6%</td>
</tr>
<tr>
<td>Large field</td>
<td>3.1%</td>
<td>3.3%</td>
</tr>
</tbody>
</table>

Table 3. Influence of the size of field of vision on the reversal rate. Both the experiment [Cornilleau-Pérès et al., 2002] and the Bayesian model exhibit smaller reversal percepts in a large field of vision.

As detailed above, the Bayesian model computes the probability distribution over the orientation $\Theta$ of the plane, given the optic flow, the field of view and the observer’s movement (example in Fig. 4). An ambiguity in the optic flow interpretation, such as the one illustrated in Fig. 3, results in a multimodal probability distribution. To compare the reversal rate reported by Cornilleau-Pérès et al. [2002] with the model output, we compute the sum of probabilities corresponding to tilt errors greater than $90^\circ$ (see Table 3).

This result is accounted for by the rigidity assumption. In our model, this assumption is expressed by a probability distribution over the optic flow. The tilt ambiguity is a consequence of the invariance of the first-order components of optic flow ($\Phi^1$) with respect to tilt reversal; therefore, only the second-order components can disambiguate the stimulus.

The reduction of reversal rate in a larger field is explained in the model by the increased influence of second-order optic flow components on the speed of a dot, which is quadratic in the size of the field of view. In contrast, the influence of first-order optic flow is only linear. Therefore, the relative influence of second-order versus first-order optic flow increases with the size of field. In our model, this is expressed by a reduced uncertainty of the second-order optic flow for large fields of view.

Qualitatively, in so far as this uncertainty is greater in small fields, the probability of depth reversal will always be higher in a small field than in a
Fig. 4. Examples of probability distributions on the orientation of a plane. The polar angle is the tilt of the plane, the radius is the tangent of the slant angle, and the colour stands for the probability, with darker colours representing higher probabilities. The peaks represent the most likely percepts, with the integral of the probability around a peak corresponding to the probability of the associated percept. The top panel shows a result with a high rate of depth reversals, while the bottom result displays a low reversal rate.

large field. Figure 5 shows the quantitative evolution of the reversal rate in the model as a function of this parameter.

2.3 Depth reversal in moving and immobile observers

Recently, self-motion has been shown to modify depth perception from optic flow. This can be seen most clearly in studies that find differences in SFM performance between moving and immobile observers, while keeping optic
Bayesian modelling of perception of structure from motion 3 29

Fig. 5. Influence of the uncertainty of second-order optic flow on the reversal rate in the Bayesian model. A small field of vision leads to a greater uncertainty, and hence to more depth reversals.

flow the same in the two conditions. Thus, actively generated optic flow can lead to a different perception of 3D shape than the same optic flow viewed passively by an immobile observer.

One of the ways in which self-motion modifies SFM is by diminishing the ambiguity that leads to depth reversals [Rogers and Rogers, 1992, Dijkstra et al., 1995, Wexler et al., 2001a,b]. An optic flow field such as the one shown in Fig. 3(a) leads, for an immobile observer, to total ambiguity between the solutions shown in Fig. 3(b), and therefore a depth reversal rate of up to 50% for a small field of view. On the other hand, for a moving observer (Fig. 3(c)), the ambiguity is reduced in favour of the solution that is most stationary in an observer-independent reference frame (left solution in Fig. 3(c)).

The experimental data used as a reference are taken from Van Boxtel et al. [2003]. The plane is observed in a small field of vision. There are two different conditions corresponding to Figs 3(b) and (c): either the participant is immobile and the plane is in rotation around the vertical axis (immobile condition), or the participant is moving his or her head transversally, and the plane is immobile in the allocentric reference frame (active condition).

The experimental results for both the active and immobile conditions are shown in Fig. 6. They reveal a bimodal distribution of tilt perception when the subject is immobile. There are two preferred responses: around 0°, corresponding to the simulated plane, and 180°, corresponding to the depth-reversed plane. In the active condition, the same optic flow is produced by the participant’s displacement in front of an immobile plane. In this case, the depth-reversed plane is rarely reported, leading to a dominant peak in the distribution around 0°.
Fig. 6. Distributions of error on tilt angle for both active (top) and immobile (bottom) conditions, by Van Boxtel et al. [2003]. The results show depth reversals in the immobile condition and its almost complete disappearance in the active condition.

Figure 7 shows the results of our model in the same two situations. We notice the bimodality in the immobile condition similar to the experimental results as well as the decrease of depth reversals in the active condition. In the Bayesian model, the bimodality derives from the symmetry of first-order optic flow mentioned above. Furthermore, the difference between active and immobile conditions can be accounted for only by the conditional distribution on motion in an observer-independent reference frame. This distribution is the expression of the stationarity assumption in our model. In the immobile condition, the simulated and depth-reversed planes have the same speed, as depicted in Fig. 3(b); only the direction of motion changes. In the active condition, however, the simulated plane is stationary in an observer-independent reference frame, whereas the depth-reversed plane has a high velocity (see Fig. 3(c)). Therefore, the stationarity assumption, as implemented in the model, ensures that the reversed plane is less probable, because it corresponds to a higher velocity in an observer-independent reference frame.

2.4 Ambiguity between slant and speed

The slant of a plane is difficult to extract from optic flow. Indeed, the rotation around an axis lying in the fronto-parallel plane is entangled with surface slant (the angle between the surface normal and the direction of gaze). Starting from
Bayesian modelling of perception of structure from motion

![Figure 7](image-url)

**Fig. 7.** Probability distributions of tilt errors in active and immobile conditions. As in the experimental results shown in Fig. 6, the ambiguity diminishes drastically in the active condition.

A given slant and motion configuration, simultaneously increasing slant and decreasing motion leads to approximately the same optic flow.

The experimental data that we consider are taken from Domini and Caudek [1999]. The experimental conditions involve a static monocular observer. The stimulus consists of a plane rotating along a fronto-parallel axis. The observer is asked to make a judgement about the slant of the plane. The planes can have two different slants and two different angular velocities. The relationship between the chosen slants is such that the tangent of the second slant is twice that of the first. The same holds for the velocity, where the second is twice that of the first (see Fig. 3(e)).

The experimental results, from Domini and Caudek [1999], are shown in Table 4. The columns on the left show the evolution of the perception of the tangent of the slant angle while changing the values of angular speed or the simulated slant. These data show that the slant of the plane is hardly recovered as an independent variable, arguing against a veridical (Euclidean, see the review by Domini and Caudek [2003]) analysis of optic flow by human
observers. Moreover, the perceived slant for small simulated slant and high angular speed is very close to the one perceived in the case of large simulated slant at low speed. Finally, this experiment shows that increasing the simulated slant or increasing the angular speed yields the same increase in perceived slant (around 23% each time).

The right columns of Table 4 show the predictions of our model in the same experimental conditions. Our model shows the slant/speed ambiguity found in the experimental results. In particular, the perceived slant for small slant with high angular speed is very close to the perceived slant for large slant with low angular speed. These results also show an increase in slant perception while increasing either slant or speed. As in the experimental data, this increase is roughly the same (50 to 60%) in both conditions, although greater than in the experimental data.

The perceived slant comes from a trade-off between our prior over the orientation (tilt and slant) of the plane and the distribution over the relative motion from the stationarity assumption (see the methods section for details).

Note that the values of perceived slant from the model are slightly smaller than those from the experimental data, especially for a small simulated slant. We have chosen to provide the results of our model with a unique set of parameters for all the experiments of this section. These parameters are therefore a trade-off between the best parameters fitting each experiment.

<table>
<thead>
<tr>
<th>Angular speed</th>
<th>Experiment</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small slant (1.5)</td>
<td>1.13</td>
<td>1.29</td>
</tr>
<tr>
<td>Large slant (3)</td>
<td>1.28</td>
<td>1.71</td>
</tr>
</tbody>
</table>

Table 4. Mean perceived tangent of slant as a function of simulated slant tangent and angular speed for both the experimental data [Domini and Caudek, 1999] and the Bayesian model. Note the growth of perceived slant with increasing angular speed and very similar perceived slant for large simulated slant/slow rotation and small simulated slant/fast rotation.

The slant/speed ambiguity results from ambiguities in first-order optic flow. In both situations (small slant, high speed compared to large slant, low speed) the optic flow is the same up to the first order, as shown in Fig. 3(e) and only the second-order optic flow could disambiguate the stimuli. These results confirm the low weighting of second-order components of optic flow in a small field of view. This low weighting arises from the uncertainty attached to the distribution over the second-order optic flow.

First-order optic flow depends on a parameter called \( def \), the product of the tangent of the slant and angular speed [Domini and Caudek, 2003].\(^5\) Therefore,
slant and speed cannot be recovered individually from first-order optic flow. Domini and Caudek [2003] propose a maximum-likelihood model to account for their psychophysical results. With a small size of field, in the absence of self-motion and translation, and disregarding second-order optic flow, the likelihood of our Bayesian model reduces to the Gaussian $P(\Phi_1 | \Omega \wedge \Theta)$. The norm of the first-order optic flow in this case is $\sqrt{\omega_X^2 + \omega_Y^2} \sqrt{\chi^2 + \upsilon^2} = |\Omega| \tan \sigma$. Their model is thus a special case of our Bayesian model.

### 2.5 Ambiguity of translation in depth

Another symmetry or ambiguity of first-order optic flow is shown in Fig. 8. A rotation in depth generates the same (up to first-order) optic flow as a translation in depth together with a different rotation in depth, around an axis that differs by $90^\circ$ from the original rotation. It has been found [Wexler et al., 2001a, Wexler, 2003] that the two solutions are perceived with different frequencies, depending on the observer’s movement and the origin of the depth translation—whether the observer moves towards the surface, or the surface moves towards the observer (see Fig. 8). These results can be summarized by stating that there is a strong bias towards perceiving the solution that minimizes motion in an observer-independent reference frame. Thus, these results provide further support for the stationarity assumption. However, the observer's percepts are also, by and large, in agreement with the rigidity assumption. Therefore, they provide a useful testing ground for our model, which incorporates both the stationarity and rigidity assumptions.

In the psychophysical studies, two conditions were tested: in the active condition, the observer moves his or her head in depth; in the immobile condition, the observer remains still but receives the same optic flow as in a previous active trial [Wexler et al., 2001a, Wexler, 2003]. In the active condition, the optic flow is generated by a plane rotating in depth at a fixed distance from the observer (the plane’s centre therefore undergoes depth translation as well). Therefore, in the active condition 8(d), the rigidity assumption favours the simulated plane, while the stationarity assumption favours the alternative solution. In the immobile condition, on the other hand, both the rigidity and stationarity assumptions favour the simulated plane.

The experimental results are presented in Fig. 8(c’), (d’), and in Table 5 as the fraction of trials in which the observers perceive the alternative, non-rigid plane. Recall that optic flow is the same in the active and immobile conditions; only the observer’s motion differs. Providing that only first-order optic flow

---

6 Other conditions, involving conflict between the observer’s motor command and self-motion, were also tested [Wexler, 2003] and were found to lead to different response distributions.

7 The reason why the rigidity assumption favours the simulated plane rather than the alternative solution is that the symmetry of Fig. 8 only holds for first-order optic flow. Second-order terms break the symmetry and lead to non-rigidity of the alternative solution.
Fig. 8. Illustration of the effect of head motion on the perception of 3D structure [Wexler et al., 2001a, Wexler, 2003]. (a) An ambiguous 2D optic flow field that can have different 3D interpretations, discovered by J. Droulez (cf. Fig. 3(a)). The arrows represent motions of projections of points in 3D space on the retina. It is fairly easy to see that the 3D configuration shown in (c) will generate this flow. However, the configuration shown in (c′) can also generate the flow in (a), and the reason for this is shown in (b) and (b′): if the amplitudes of the translation and rotation in (c′) are adjusted correctly, the rotation can exactly cancel the expansion flow from the depth translation in one of two dimensions. The planes in (c) and (c′) have the same slant and angular speed but different tilts, and they rotate about different axes. (d), (d′) Because optic flow depends only on the relative motion between object and observer, the same ambiguity holds for an observer moving forward and experiencing the optic flow in (a). If the observer’s speed is equal and opposite to the translation in (c′), the stationarity of the solutions is reversed with respect to (c) and (c′): it is now the centre of (d′) that is stationary in space, while (d) translates at the same speed as the observer. (c′′), (d′′) Data from Wexler [2003], showing the frequencies of the perceived solutions for stationary (c′′) and moving (d′′) observers, with the bars on the left corresponding to solutions (c) and (d), and the bars on the right to solutions (c′) and (d′). Although the optic flow is the same in the two cases, perceptions of 3D structure are very different, showing the effect of the observer’s action.
components are available, the rigidity assumption alone would predict equally low rates for the alternative solutions in the two conditions, whereas stationarity alone would result in a rate close to 100% in the active condition and a low rate in the immobile condition. Second-order optic flow components, if available, would decrease the rate for the alternative non-rigid solution.

As explained above, the discrepancy between the actual values of the experimental results and the model are because of the unique parameter set used for all six experiments. More precisely, different groups of participants already exhibit differences in their results. See, for instance, Fig. 8(c"") and the bottom-left histogram in Fig. 10. Both correspond to the same conditions, but the results are numerically different. Priors in our model can be adjusted to fit some results better at the expense of other experiments.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Experiment</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>54.3%</td>
<td>79.6%</td>
</tr>
<tr>
<td>Immobile</td>
<td>3.6%</td>
<td>17.8%</td>
</tr>
</tbody>
</table>

Table 5. Rate of alternative, non-rigid responses for the ambiguous depth-translation stimulus. Experimental results from Wexler [2003] (which do not explicitly state the immobile results). The higher rate in the active condition than in the passive one is because of the stationarity assumption. In the immobile condition, both the stationarity and rigidity assumptions favour the same percept.

Because our model implements both the rigidity and stationarity assumptions, they are in competition when the most rigid and most stationary objects do not match. In this experiment, such a mismatch happens in an active condition. The model deals with this kind of contradiction in a way similar to Bayesian fusion [Lebeltel et al., 2004]. Other instances of Bayesian fusion are exemplified in the literature [Landy et al., 1995, Ernst and Banks, 2002]. The uncertainty, as quantified by the probability distributions, will ensure the optimal balance between the rigidity and stationarity assumptions.

2.6 Effect of shear on tilt perception

Another point that we tested with the Bayesian model is the effect of the shear component of optic flow on SfM performance. The shear angle is the absolute difference between the tilt angle and the direction of the frontal translation. It is called “winding angle” by Cornilleau-Pérès et al. [2002]. Psychophysical studies have found that SfM performance in immobile human observers (namely, judgement of tilt) deteriorates drastically as shear increases [Cornilleau-Pérès et al., 2002], but that this deterioration is much less drastic in active observers generating optic flow through their own head movements [Van Boxtel et al., 2003]. Examples of minimal and maximal shear in optic flow are shown in Fig. 9. Shear can be parameterized by the shear angle, which
takes values between 0°, corresponding to no shear, and 90°, corresponding to maximal shear.

Fig. 9. Illustration of shear in optic flow. Shear can be parameterized by the shear angle, defined as 90° minus the absolute value of the difference between tilt and axis angles. Configurations corresponding to two values of shear angle are shown, 0° (minimum shear) and 90° (maximum). The bottom row shows the optic flow resulting from each configuration.

We compare model results with experimental findings by Van Boxtel et al. [2003]. The experiment involves a monocular observer who is either immobile or moving in a direction perpendicular to the gaze direction (active condition). In both cases, the observer receives the same optic flow. In the active condition, the simulated plane is stationary in an observer-independent reference frame. In the immobile condition, the plane rotates around an axis in the fronto-parallel plane. The observer’s task is to report the plane’s orientation by aligning a probe so that it appears parallel to the plane.

Figure 10 shows the distribution of absolute tilt errors from the experimental results [Van Boxtel et al., 2003], in both active and immobile conditions, for minimal and maximal shear. First, we see that mean errors increase with increasing shear. However, this effect is much stronger in the immobile condition (where response is almost at the chance level).

Figure 11 shows the distribution of absolute tilt errors for the same conditions as given by the model. The variation of the precision between low and high shear is similar to the experimental results.

In the model, the main factor inducing the shear effect is the relative weights of the rotation prior and the translation prior. For a small shear, the absolute motion satisfying the first-order optic flow equations for a large tilt error is composed of a rotation and a translation. For a high shear, a large error corresponds to an absolute motion composed of two rotations with the same
Bayesian modelling of perception of structure from motion

2.7 Influence of eye movements on 3D vision

Using a sinusoidally curved surface that underwent lateral translation while being pursued with the eyes by the subject, Naji and Freeman [2004] found few depth reversals. However, when the same optic flow was presented without pursuit (i.e. with the translation subtracted), depth reversals were prevalent. We simulated a very similar experiment, the only difference being that we used a planar rather than a curved surface. Because planes can undergo depth reversals in the same way as curved surfaces, the main effect of Naji and Freeman, or something very close to it, can be simulated within the framework of our model.
As can be seen in Fig. 3(d) (analogous to condition C of Naji and Freeman [2004]), depth reversals can take place in the pursuit condition. Both solutions undergo the same translation, and one of the solutions additionally undergoes a rotation. In the fixation condition (analogous to condition B of Naji and Freeman [2004]), the same optic flow leads to two solutions undergoing equal and opposite rotations, as shown in Fig. 3(b). Finally, Naji and Freeman [2004] have a third condition (A) where the object translates as in condition C but in which the observers were required to fixate on a stationary point rather than pursue the object.

The rate of depth reversals is calculated from subjects’ responses in a depth-order task. Figure 12 shows the experimental results in these three conditions. The graphs show the estimation of the phase with respect to the amplitude of the stimulus. The phase is the analogue of the orientation of the plane in Figs 3(d) and (b), whereas the amplitude stands for the slant of the plane (negative slant being a reversal). We notice that translation (A and C) allows for the disambiguation of the stimulus, whereas rotation exhibits a symmetric behaviour. We notice that the perception is more precise in the pursuit condition (C) than the immobile condition (A).
Fig. 12. Rate of ‘top-far’ perception with respect to the strength of the stimulus [Naji and Freeman, 2004]. Condition A corresponds to a translating object without eye pursuit, condition B to a rotating object, and condition C to a translating object with pursuit. Conditions A and B show that translation allows for a disambiguation contrary to passive rotation. Furthermore, a comparison of conditions A and C shows that pursuit of the object leads to better perception.

In comparison, Fig. 13 shows the results of the Bayesian model in the transposed conditions. We can see that the major properties are reproduced, in particular, the broader uncertainty in condition A compared with condition C, as well as the ambiguity in condition B.

Fig. 13. Results from the model. As for the experimental results, conditions A and C allow for a disambiguation of the stimulus, and condition C is less uncertain than condition A.

Up to now, subjective responses have been limited to the plane orientation. An additional element must be included in the model to account for the ‘top-far’ responses. This decision was made using a simple Bayesian program. As can be seen in conditions A and C in Fig. 12, the observers exhibited some preference towards a ‘top-far’ perception. This preference is included as a prior in the Bayesian post-processing. However, it is to be noted that observers seem to have a preference for a ‘top-near’ perception in condition B.
The results in condition B are the same as in the immobile condition above. The small asymmetry of both top and bottom curves comes from the second-order optic flow that induces a reversal rate strictly less than 50%.

The difference between the model results in conditions A and C comes from the stationarity of the reverse percepts. In condition C, the reverse percept undergoes a greater rotation than in condition A. Therefore, the stationarity assumption assigns a smaller probability to it, hence yielding a smaller reversal rate.

3 Discussion

3.1 Probabilistic expression of assumptions

A Bayesian model infers the logical consequences of a given set of assumptions with some observations. The inference can occur as soon as a joint probability distribution is defined. Therefore, the modeller must express the assumptions in a Bayesian way.

Each choice in a specification is an assumption. As there are multiple steps in the specification of a joint probability distribution, there are multiple levels of expression. The first choice is the variables and their domain. The variables absent at this step cannot have a meaningful influence in the model. Then, the joint distribution is decomposed into a product of factors using conditional independencies, which express a lack of relationship between variables and therefore simplify the inference. The last level of expression of assumptions is in the specification of each distribution and its parameters appearing into the decomposition.

Each choice reduces the number of degrees of freedom of the joint distribution. The more drastic restrictions are in the choice of variables and their domains, while the less important are in the choice of the parameters of the distributions. Any reduction can be postponed to a later stage, but the earlier it is done, the more the inference can take advantage of it to simplify the computations.

3.2 Choices in our Bayesian model

The issue of specification is therefore to express our assumptions in terms of choice of variable, simplification of the joint distribution using conditional independencies, and choice of parametric forms and parameters.

The first main assumption is that of rigidity, which states that the optic flow more likely to be observed is generated by a plane in relative motion. The parametric space of the optic flow is derived from this assumption. The optic flow is defined by eight parameters. While sufficient for a plane, the optic flow is, in general, more complicated. This means that other eventual components are not relevant variables in our model, and are therefore ignored. It could be
interesting to investigate an eventual effect of these components in the human perception of a plane. As far as the model is concerned, such investigation can be studied with additional components in the optic flow variable. Rigidity is also pre-eminent in the choice of the parametric form of the probability distribution over optic flow, given relative motion, position of the plane, and the conditions of observation. We fixed this as a Gaussian distribution. However, it would be possible to evaluate this choice of distribution by measuring the evolution of performance with respect to some additional noise in the stimulus and comparing it with the predicted evolution of the model. The other main assumption of our model is that of stationarity, which states that the motion of the plane is more likely to be small.

The variables chosen to describe the motion are the translation and rotation components along the three axes, according to the experiments chosen as references. This is restrictive in the sense that it does not take into account eventual accelerations and even more complex trajectories. Most reported studies deal with uniform motion; however, investigation of the influence of accelerations in the perception of structure could benefit from the model. The model can be adapted to implement and predict the results of different hypotheses to be compared with experimental results.

The parameters are the last elements of choice in the model. We obtained the results presented above with a single set of parameters. Each experimental result gives information on the exact effect highlighted by the experiment on some parameters. However, each experiment has different optimal parameters; therefore, the final set of parameters chosen results from a trade-off between all the experiments.

3.3 Model results

The results of the model display some discrepancies with the results of the experiments. For example, for the first experiment described, the reversal rate of the model in a small field is 44.6% compared with 48.8% in the experiment [Cornilleau-Pérès et al., 2002]. There are two main reasons for this difference. First, the Bayesian model is a model of an observer. It is not specifically designed to reproduce mean results across observers. Nevertheless, the results of our model are less variable than the results reported between observers (in this case, the minimum reversal rate reported by Cornilleau-Pérès et al. [2002] is around 38%).

As explained above, the set of parameters is the same across all the results of our model. However, there are variations in the precise experimental conditions between the different teams responsible for the measured results. For instance, the rate of reversal measured in a small field of view for an immobile observer by Van Boxtel et al. [2003] is 35%, compared with the 48.8% measured by Cornilleau-Pérès et al. [2002]. This can be explained by differences in the protocols that are not taken into account as relevant variables in the Bayesian model. Therefore, as a general rule, the parameters we chose for the
Bayesian model are a trade-off between all the results. Accordingly, the results of the model cannot match the experimental results precisely. The Bayesian model not only accounts for previously reported results but also can be used to make predictions and eventually to propose new experiments. For example, we propose the investigation of the relative influence of stationarity and rigidity in large fields of view. In this case, in an experimental set-up similar to that of Wexler [2003], our model predicts that rigidity will be of greater importance in the perception of second-order optic flow through a diminution of the standard deviations on these components.

Another prediction of the Bayesian model involves the shear effect. In our model, this effect is accounted for by relative weight between rotation and translation components in a small field of vision. Our model predicts a reduced shear effect in large fields of vision, and this has been found in human observers [Cornilleau-Pérès et al., 2002].

A Equations of optic flow

Let \( \mathcal{P} \) be the object plane, \((\chi, v)\) its depth gradients, \(\tilde{M}\) with coordinates \((\tilde{x}, \tilde{y}, \tilde{z})\) a point of this plane in the 3D reference frame, and \(M\) with coordinates \((x, y)\) its projection in the image plane. The equation of the plane is:

\[
\tilde{x}\chi + \tilde{y}v - \tilde{z} = 0. \tag{10}
\]

We have the slant of the plane \(\sigma = \arctan \sqrt{\chi^2 + v^2}\) and the tilt \(\tau = \arctan \frac{v}{\chi}\).

Let \(P\) be the projection of a 3D point in the image.

\[
P : \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{pmatrix} \rightarrow \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \frac{\tilde{x}}{1-\tilde{z}} \\ \frac{\tilde{y}}{1-\tilde{z}} \end{pmatrix}. \tag{11}
\]

Let \(t = (t_x, t_y, t_z)\) and \(\omega = (\omega_x, \omega_y, \omega_z)\) be respectively the translation and rotation vector of the object plane.

Considering the points as functions of time, we can write:

\[
M(t) = P \circ \tilde{M}(t). \tag{12}
\]

Optic flow is the displacement of the points in the image:

\[
\phi = \frac{dM}{dt}, \tag{13}
\]

\[
\phi = \frac{dP}{dM}(\tilde{M}) \times \frac{d\tilde{M}}{dt}. \tag{14}
\]

\(\frac{dP}{dM}\) is the Jacobian of \(P\).
The plane $\mathcal{P}$ undergoes translation $t$ and rotation $\omega$. Therefore the motion $\frac{d\tilde{M}}{dt}$ of $M$ is as follows.

$$
\frac{d\tilde{M}}{dt} = t + \omega \wedge \tilde{M}
$$

Substituting 15 and 16 in equation 14, we obtain the following.

$$
\phi = \frac{dP}{dM} \times \frac{d\tilde{M}}{dt}
$$

$$
\phi = \begin{pmatrix}
\frac{t_x + \chi \omega_y \tilde{x} + (\upsilon \omega_y - \omega_z) \tilde{y}}{1 - \frac{z}{z}} \\
\frac{t_y + (\omega_z - \chi \omega_x) \tilde{x} - \upsilon \omega_x \tilde{y}}{1 - \frac{z}{z}} \\
\frac{t_z + \omega_y \tilde{y} - \omega_y \tilde{x}}{1 - \frac{z}{z}}
\end{pmatrix}
$$

By the definition of $P$ (equation 11), $\frac{\tilde{x}}{1 - \frac{z}{z}} = x$, $\frac{\tilde{y}}{1 - \frac{z}{z}} = y$ and $\frac{\tilde{z}}{1 - \frac{z}{z}} = 1 + \chi x + \upsilon y$. We can finally rewrite the equation 17 to obtain the equations 18 of the optic flow of a plane, as follows.

$$
\phi = \begin{pmatrix}
\frac{t_x + \chi \omega_y \tilde{x} + (\upsilon \omega_y - \omega_z) \tilde{y}}{1 - \frac{z}{z}} \\
\frac{t_y + (\omega_z - \chi \omega_x) \tilde{x} - \upsilon \omega_x \tilde{y}}{1 - \frac{z}{z}} \\
\frac{t_z + \omega_y \tilde{y} - \omega_y \tilde{x}}{1 - \frac{z}{z}}
\end{pmatrix}
$$

$$
\phi = \begin{pmatrix}
\frac{t_x + x [\omega_z + \chi (t_x + \omega_y)] + y [-\omega_z + \upsilon (t_x + \omega_y)]}{1 - \frac{z}{z}} \\
\frac{t_y + x [\omega_z + \chi (t_y - \omega_x)] + y [t_z + \upsilon (t_y - \omega_x)]}{1 - \frac{z}{z}} \\
\frac{t_z + x \chi t_z - \omega_y + y^2 (\upsilon t_z + \omega_x)}{1 - \frac{z}{z}}
\end{pmatrix}
$$

with:
\[ \begin{align*}
\phi^0 &= \begin{pmatrix} t_x \\ t_y \end{pmatrix} \\
\phi^1 &= \begin{pmatrix} t_x + \chi(t_x + \omega_y) & -\omega_z + \nu(t_x + \omega_y) \\
\omega_z + \chi(t_y - \omega_x) & t_x + \nu(t_y - \omega_x) \end{pmatrix} \\
\phi^2 &= \begin{pmatrix} \chi t_z - \omega_y \\
\nu t_z + \omega_x \end{pmatrix}
\end{align*} \]

References


Building a Talking Baby Robot: 
A contribution to the study of speech 
acquisition and evolution

Jihène E. Serkhane¹, Jean-Luc Schwartz¹, and Pierre Bessière²

¹ CNRS - Institut de la Communication Parlée (ICP)
² CNRS - Grenoble Université

Speech is a perceptuo-motor system. A natural computational modelling framework is provided by cognitive robotics, or more precisely speech robotics, which is also based on embodiment, multimodality, development, and interaction. This chapter describes the bases of a virtual baby robot, an articulatory model that integrates the non-uniform growth of the vocal tract, a set of sensors, and a learning model. The articulatory model delivers sagittal contour, lip shape and acoustic formants from seven input parameters that characterize the configurations of the jaw, the tongue, the lips and the larynx. To simulate the growth of the vocal tract from birth to adulthood, a process modifies the longitudinal dimension of the vocal tract shape as a function of age. The auditory system of the robot comprises a “phasic” system for event detection over time, and a “tonic” system to track formants. The model of visual perception specifies the basic lip characteristics: height, width, area and protrusion. The orosensory channel, which provides tactile sensations on the lips, the tongue and the palate, is elaborated as a model for the prediction of tongue–palatal contacts from articulatory commands. Learning involves Bayesian programming, in which there are two phases: (i) specification of the variables, decomposition of the joint distribution and identification of the free parameters through exploration of a learning set; and (ii) utilization, which relies on questions about the joint distribution.

Two studies were performed with this system. Each of them focused on one of the two basic mechanisms that are thought to be at work in the initial periods of speech acquisition: vocal exploration and vocal imitation. The first study attempted to assess infants’ motor skills before and at the beginning of canonical babbling. It used the model to infer the acoustic regions, the articulatory degrees of freedom and the vocal tract shapes that are most likely to be explored by actual infants according to their vocalizations. Subsequently, the aim was to simulate data reported in the literature on early vocal imitation, to test whether and how the robot was able to reproduce them and to gain
some insights into the actual cognitive representations that might be involved in this behaviour.

Speech modelling in a robotics framework should contribute to a computational approach to sensory–motor interactions in speech communication, which seems crucial for future progress in the study of speech and language ontogeny and phylogeny.

1 Introduction

1.1 Linking perception and action in speech robotics

Speech perception and production are often studied independently. However, speech is obviously a sensory–motor system. This is the starting point of the so-called “Perception for Action Control” Theory (PACT) [Schwartz et al., 2002], in which we argue that perception is a set of tools, processing and representations that enable control of actions. The PACT proposes that as the perceptual and the motor representations are acquired together during speech development, they constrain each other in adulthood, even though they belong to different domains. The main idea is that to study the perceptual and motor representations that underlie speech in adults and shape the world’s languages, a relevant strategy is to focus on how they develop in concert with each other during speech acquisition.

In this approach, a natural computational modelling framework is provided by cognitive robotics, a promoter of which is R. Brooks through the Cog project, which focuses on the notions of “[...] embodiment and physical coupling, multimodal integration, developmental organization, and social interaction.” [Brooks et al., 1999].

Embodiment, multimodality, development and interaction are also the core of “Speech Robotics” [Abry and Badin, 1996, Laboissière, 1992], a research program in which we try:

1. to elaborate a sensory–motor virtual “robot” able to articulate and perceive speech gestures (embodiment: Boë et al. [1995a]; Schwartz and Boë [2000]) and able to learn multisensorial–motor links (multimodality: Schwartz et al. [1998]) in parallel with the growth of its vocal apparatus;
2. to determine what could be the exploration strategies by which this robot could evolve from vocalizing and babbling to the control of complex speech gestures (development: Abry et al. [2006]); and
3. to explore how communication principles in a society composed of such agents could shape the acoustic and articulatory structures of human languages (interaction: Berrah et al. [1996]).

The present project concerns a preliminary stage of this research program. It aims at providing the foundations of a model of speech development, that is, the implementation of the virtual baby robot, which is a growing sensory–motor system able to learn and to interact [Schwartz et al., 2002].
1.2 A viewpoint on speech development

The viewpoint supported is that the development of orofacial control in speech relies on two fundamental behaviours: the progressive exploration of the vocal tract’s sensory–motor abilities, and the imitation (overt simulation) of caretakers’ language sounds. That is to say, articulatory exploration should be the way by which infants discover the abilities of their vocal tracts and learn relationships between movements and percepts. At the same time, imitation should capitalize on the knowledge acquired by exploration to tune step by step the control of the articulatory system to produce the gestures and sounds of the target languages.

The first attempts to simulate speech development in robotics were based on the assumption that infants explore their entire space of articulatory–acoustic realizations, then select their native language items out of all the possible ones [Bailly, 1997, Guenther, 1995]. In other words, infants were supposed to start by uttering all possible speech sounds in languages (in agreement with Jakobson [1968]). However, direct observation shows that infants do not do this [Kent and Miolo, 1995]: whatever their ambient language, they only produce a certain subset of what can be performed with their phylogenetically inherited sensory–motor apparatus. Moreover, on a computational level, exhaustive exploration complicates the learning of sensory–motor links [Bessière, 2000].

Infants do not explore the whole articulatory–acoustic space to master their vocal tract behaviours. Furthermore, sensory–motor developmental facts, likely to be linked with speech development, can be roughly classified according to whether they are a matter of exploration or of imitation. (a) At birth, infants are able to imitate three gestures from vision: tongue and lip protrusions, and mandible depression [Meltzoff, 2000]. Although these movements, employed in adult speech, are not obviously linked with speech development, they are nonetheless available before first vocalizations. (b) At a few weeks old, infants vocalize. Moreover, they tend to direct their productions towards vowel sounds that they often perceive (early vocal imitation: Kuhl and Meltzoff [1996]), and to match a vowel sound to the moving image of the face that utters it (multimodal integration: Kuhl and Meltzoff [1992]). (c) At about seven months, they become babblers: their mandibles move upwards and downwards in a rhythmic way, while their vocal folds vibrate. This is what has been referred to as canonical babbling [Koopmans-Van Beinum and Van Der Stelt, 1986, MacNeilage and Davis, 1990]. (d) Later on, children begin to control, more or less successively, the number of jaw cycles, the movements of the articulators carried by each cycle independently one of each other, and finally the full shape of their “vocal resonator” (motor coordination). This enables them to master sounds and sequential patterns of their ambient languages [Vilain et al., 2000].

Section 2 depicts the sensory, articulatory and learning models that comprise the virtual robot. At first, the aim was to specify its early motor skills:
articulatory exploration was assessed from the acoustic description of vocalizations produced by actual infants both before (phase b)) and at the beginning (phase c) of canonical babbling (Section 3; and see Serkhane et al. [2002]). As for the imitation issue, a model of imitation was proposed and exploited to simulate an experiment on actual infants. The influences of parameters that tune the robot’s first imitation abilities were studied, which provided some information about the sensory–motor representation likely to underlie this behaviour in infancy (Section 4; and see Serkhane and Schwartz [2003]). Section 5 gives some plans for the future of this project in relation to ontogeny and phylogeny.

2 The vocalizing baby robot

On the production level, the Variable Linear Articulatory Model (VLAM: Boë [1999]) provides the robot with a virtual vocal tract that integrates the non-uniform growth of the human tract. As for perception, the auditory, the visual and the tactile modalities are available with a model for each modality. The relationships between the tract movements and their perceived consequences are learned (during exploration) and used (in imitation) within a Bayesian robotics formalism.

2.1 The articulatory model

The Variable Linear Articulatory Model (VLAM) is a version of the Speech Maps Interactive Plant (SMIP: Boë et al. [1995a]) that integrates a model of vocal tract growth. The core of the SMIP is Maeda’s model [Maeda, 1989] or a variant proposed by Gabioud [Gabioud, 1994]. Its elaboration consisted of a thorough statistical analysis of 519 hand-drawn midsagittal contours corresponding to a 50 frames/sec. radiographic film synchronized with a labiographic film that contained 10 sentences in French, recorded at the Strasbourg Institute of Phonetics [Bothorel et al., 1986]. The midsagittal contours were analysed with a semi-polar grid, and a guided principal component analysis found that seven parameters explained 88% of the variance in the observed tongue contours for the selected (adult) speaker. A linear combination of the seven parameters enables the regeneration of a midsagittal contour of the vocal tract. The weights of each parameter were normalized using the standard deviation around the mean position of the observed values as reference. The lip shape was modelled from measurements analysed at ICP [Abry and Boë, 1986, Guiard Marigny, 1992].

Hence, the articulatory model delivers sagittal contour and lip shape from the seven input parameters (hereafter $P_i, i = 1 \ldots 7$), which may be interpreted in terms of phonetic commands, and which correspond respectively more or less to the jaw ($J$), the tongue body ($Tb$), dorsum ($Td$) and tip ($Tt$), the lip protrusion ($Lp$) and height ($Lh$), and the larynx height ($Lx$) (Fig.
1. The area function of the vocal tract is estimated from the midsagittal dimensions, with a set of coefficients derived from tomographic studies. The formants and the transfer function are calculated from the area function, and a sound can be generated from formant frequencies and bandwidths.

From this basis, it was possible to implement a growth model that enabled us to replace the “adult” robot with a “baby” one. Systematic measurements of the vocal tract from birth to adulthood do not exist at present. However, it was possible to take advantage of cranio–facial measures established at different ages by Goldstein [19880]. These data were closely fitted by (double) sigmoidal curves, which characterize the general skeletal and muscular growth. To give an account of the vocal tract growth, the articulatory VLAM model (Variable Linear Articulatory Model) developed by Maeda [Boë and Maeda, 1998] describes the evolution of the horizontal and vertical dimensions from a newborn to a female or male adult. As proposed by Goldstein, the growth process was introduced by modifying the longitudinal dimensions of the vocal tract according to two scaling factors: one for the anterior part of the vocal tract and the other for the pharynx, interpolating the zone between. The non-uniform growth of the vocal tract can thus be simulated year by year and month by month. Similarly, typical $F_0$ values were adjusted to follow the growth data presented by Mackenzie Beck [1997]. A more detailed presentation of the model, together with the assessment of its agreement with both morphological and acoustical data on infants and children, can be found in Ménard et al. [2002, 2004].
2.2 The sensory models

Auditory model

The tracking of speech gestures must involve a way to capture and characterize the basic components of the speaker’s vocal actions, namely timing and targets [Schwartz et al., 1992]. A series of influential studies in the Pavlov Institute of Leningrad in the 70s led Chistovich to propose a basic architecture for the auditory processing of speech sounds. It consists of one system specialized for temporal processing and detection of acoustic events, and the other continuously delivering various analyses about the spectral content of the input [Chistovich, 1980]. The neurophysiological bases for these systems are already available in primary neurons in the auditory nerve, or secondary neurons in the cochlear nucleus (which is the first auditory processing centre in the central nervous system). This provides the basis of the auditory system of the robot (Fig. 2).

![Fig. 2. The auditory model.](image-url)

The system specialized for event detection is based on so-called “phasic” units in the central nervous system, namely “on” and “off” units responding only to quick increases and decreases of the neural excitation in a given spectral region. We developed a physiologically plausible module for the detection of articulatory–acoustic events such as voicing onset/offset, bursts, and vocalic onset/offset [Piquemal et al., 1996, Wu et al., 1996] in the cochlear nucleus. These events, which allow the labelling of every major discontinuity in the speech signal, are crucial for the control of timing in speech production [Abry et al., 1985, 1990].
The system specialized for spectral processing requires so-called “tonic” units responding continuously to a given stimulus, and then enabling precise statistics and computations about the variations of excitation depending on their characteristic frequency. Although the debate on the role of formants in the auditory processing of speech is far from closed (e.g. Bladon [1982] or Pols [1975]), it seems that the basic neurophysiological ingredients are available for formant detection in the auditory nerve, through spatio-temporal statistics [Delgutte, 1984]; and higher in the auditory chain, as early as in the cochlear nucleus, through lateral inhibition mechanisms for contrast reinforcement. Hence, formants are the basic spectral parameters characterizing speech sounds in our system.

Visual model

In the multisensorial framework, the robot requires eyes as much as ears. Indeed, it is quite well known that speech is not only heard but also seen (e.g. Dodd and Campbell [1987] or Campbell et al. [1998]). Speech-reading enables us to follow speech gestures partly when audition is inadequate, particularly in hearing impairment; it improves speech intelligibility in noisy audio conditions or with foreign languages; it intervenes in gesture recovery even if the visual input conflicts with the audio one, as in the famous McGurk effect [McGurk and MacDonald, 1976]; and visual input is implied in the development of speech control and in the acquisition of phonology in conjunction with cued speech for hearing-impaired people (see Schwartz et al. [2002] for a review of audiovisual fusion in the context of a theory of perception for action control). The visual sensor should be able to capture what can be seen on the speaker’s face: lip geometry, jaw position, and probably some parts of the tongue. At present, the visual inputs of the robot are the basic lip characteristics: height, width, area and protrusion.

Tactile model

The orosensorial channel, which contains the tactile sensations of the lips, the tongue and the palate, is commonly absent from models of the perceptual representation of speech gestures. However, humans possess a highly developed representation of the oral space. This is illustrated by data on oral stereognosis in which subjects are able to integrate tactile and motor information to identify three-dimensional objects placed in their mouths [Bosma, 1967]. The tip of the tongue and the lips are among the most sensitive parts of the body surface, as displayed by two-point discrimination data. The neurophysiology of the tactile orosensory system has been described in a number of reviews (see e.g. Hardcastle [1976], Landgren and Olsson [1982] or Kent et al. [1990]). Most of the oral mucosa, and particularly the tongue, are supplied with mechanoreceptors of different types, able to provide detailed information about the position of the jaw, lips and tongue, and the velocity and direction
of movement. Histological data show that the density of sensory endings decreases progressively from the front to the rear of the mouth: the tip of the tongue seems the best endowed with receptors in the oral system, in agreement with its accurate tactile acuity. Several studies show the importance of the tactile sensor for speech control. MacNeilage et al. [1967] cited the case of a patient with a generalised congenial deficit in somesthesic perception: she produced totally unintelligible speech though she had no apparent auditory or motor trouble. Hoole [1987] and Lindblom et al. [1977] showed the influence of oral sensitivity for the production of perturbed speech (bite-block experiments).

The above facts motivated the elaboration of a model for the prediction of palatal contacts of the tongue from articulatory commands [Schwartz and Boë, 2000]. In this model, patterns of palatal contacts are described by five variables (hereafter $L_i$, $i = 1 \ldots 5$), defining the number of contacts per line along five lines that go from the periphery to the middle of the palate [Recasens, 1991] (Fig. 3).

![Fig. 3. The palatal tactile sensor of the baby robot (see text).](image)

The $L_i$ values are predicted from the articulatory commands $P_j$ by a linear-with-threshold associator:

$$L_i = f(w_{ij} \times P_j + w_{i0}),$$

where $w_{ij}$ and $w_{i0}$ are the weights and the bias to learn, and $f$ is a threshold function limiting $L_i$ to their ranges of variation; that is, from 0 (no palatal contact in the corresponding line) to their maximal possible values (respec-
tively 9, 8, 7, 5, 4). The values of $w_{ij}$ and $w_0$ were tuned by minimizing a summed square error between observed and predicted $L_i$ values (Fig. 4).

![Fig. 4. Predicted (black) and observed (grey) palatal configurations for prototypical [i], [a], [o] (from top to bottom).](image)

To test the behaviour of this model, a set of predicted palatal contacts were computed for a great number (about 1000) of articulatory configurations that lead to formant frequencies in the acoustic regions of the vowels [i], [a], [u]. Though these configurations vary largely in their articulatory parameters, it appeared that the predicted palatal contacts were quite coherent (Fig. 5) and were in line with the variability of contacts observed by Recasens [1991] for vowels embedded in various consonantal contexts. Hence, the model seems able to provide useful predictions when adequately linked with the articulatory and acoustical structure of the gesture.

**Simplified perceptual models**

To focus on learning, in this study, we chose to take into account a restricted and simplified set of sensory variables that were easily interpretable in phonetic terms.

The auditory variables were the first two formant frequencies ($F_1$, $F_2$) expressed in Bark, a scale of frequency perception [Schroeder et al., 1979].

The simplified tactile system relied on the vocal tract geometry, which can be described by the following systems [Boë et al., 1995b]; (i) the area ($A_c$) and the distance from the glottis ($X_c$) of the main constriction along the vocal tract, as well as the interlip area ($A_l$) when produced by the robot vocal tract.
Fig. 5. Predicted palatal configurations (left) for 1000 articulatory configurations around [i] (formants on the right; the same was done for [a] and [u]).

[Fant, 1960]; and (ii) the coordinates \((X_h, Y_h)\) of the tongue’s highest point in a fixed system of reference [Boë et al., 1992]. The visual system estimates \(AI\) when it comes from a peer’s face. This set of variables is displayed in Fig. 6.

Fig. 6. The simplified sensory models.
3 Simulating vocal exploration before and at the beginning of babbling

As infants do not begin by exploring all possible speech sounds, we first tried to assess the articulatory abilities available both before and at the beginning of canonical babbling, that is, at four and seven months. To obtain this information from the two first formant frequencies of vocalizations produced by real subjects at these developmental stages, three specially designed analysis techniques were developed. They were termed acoustic framing, articulatory framing and geometric framing. Their descriptions and results will be given after the data that they processed are presented.

3.1 Phonetic data

We had two sets of data from studies in developmental phonetics. The first one is composed of vowel-like sounds produced by four-month-old subjects, during early vocal imitation tests from Kuhl and Meltzoff [1996] (see Section 4.1 for further details). Matyear and Davis supplied us with the second set of data, collected to study syllable-like productions in canonical babbling [Matyear, 1997, Matyear et al., 1998]. We selected their seven-month-old subjects’ vowel-like sounds at canonical babbling onset. These two studies were carefully acquired and carefully labelled and analysed in a series of paradigms and protocols described in great detail in the original publications. In each case, the two first formant values and a phonetic description were available for analysis.

3.2 Acoustic framing

Method

All the sounds generated by the VLAM belong to the Maximal Vowel Space (MVS: Boë et al. [1989]). MVS corresponds to what an infant at a given age would be able to produce if he/she used his/her complete set of articulatory commands, defined as all values between \(-3\) and \(+3\) times the standard deviation, thus covering the whole range of possible values for each parameter. MVS thus stands for all “possible speech sounds” plotted on a multiformant \((F_i)\) map. The \((F_1, F_2)\) plane displays the vocalic triangle developed by phoneticians; its corners include the vowels \([i a u]\). Acoustic framing consists of superimposing an age-specified set of actual data on the MVS of the VLAM at the same age. Hence, it tests whether actual vocalizations belong to this MVS and assesses the acoustic space region(s) explored by four- and seven-month-old infants.
Results

Each set of actual vocalizations did belong to the corresponding MVS (Fig. 7 and 8). Moreover, the actual data did not entirely cover the space corresponding to mature motor control products. More precisely, the four-month-old vocalizations, displayed as black dots superimposed on the MVS in grey in Fig. 7, were relatively centred and mid-high: the most fronted, backed and open productions did not seem to be exploited. At seven months (Fig. 8), the vocalic productions exploited the high–low dimension more than at the earlier stage.

![Fig. 7. Acoustic framing of four-month-old babies' vocalizations (black dots). The grey dots correspond to the four-month MVS. The $F_1$ are expressed in Hertz.](image)

3.3 Articulatory framing

Method

Certain regions of the MVS, generated by the seven articulatory parameters of the VLAM, were not exploited by the actual data. Articulatory framing allowed us to evaluate infants' motor abilities by constraining the motor variables of the VLAM. In other words, this method aims at assessing the minimal set of articulatory degrees of freedom required to reproduce the observed vocalic sounds. We built several articulatory submodels with different sets of the VLAM motor parameters. A given submodel was therefore characterized by its number of articulators and their ranges of variation. Two hundred and
Building a Talking Baby Robot

Fig. 8. Acoustic framing of seven-month-old babies’ vocalizations (black dots). The grey dots correspond to the seven-month MVS. The $F_i$ are expressed in Hertz.

fifty-five submodels were comparatively assessed with respect to the efficiency with which they reproduced each collection of phonetic data, using the probabilities given the actual vocalizations: $P(M_i|f_1 \land f_2)$, where $M_i$ denotes the $i$th submodel, characterized by the set of acoustic outputs that it generates, while $f_1 \land f_2$ stands for the actual data formant values. The winner is the submodel that best fitted a given set of actual data: it maximized the conditional probability criterion.

**Results**

The results for the four-month-old data indicate that exploration at four months is rather reduced around the neutral configuration. It involves at least three articulatory parameters including at least one for the tongue, and the jaw seems to play a minor role in this exploration. The winner submodel (Fig. 9) exploited the lower lip height ($Lh$), tongue body ($Tb$) and dorsum ($Td$) degrees of freedom. At seven months, the explored region is much larger, and the jaw now plays a dominant role, leading to a large exploration of the open–close contrast and its associated $F_1$ dimension in the formant space (Fig. 10). This result agrees with babblers’ mandible use.

### 3.4 Geometric framing

**Method**

Articulatory framing enabled us to infer the tongue configurations that could have yielded the acoustic data recorded at four and seven months. Geometric
Fig. 9. The articulatory framing of the four-month-old babies’ vocalizations by the selected three-parameter articulatory sub-model. The black dots correspond to the actual data, and the grey ones correspond to the submodel acoustic outputs. The grid shows the boxes employed to compute the probability criterion. The Fi are expressed in Hertz.

Fig. 10. The articulatory framing of the seven-month-old babies’ vocalizations by the selected four-parameters articulatory submodel. The same conventions apply as in Fig. 9.
framing is a method of exhaustive inversion: each vocalization corresponds to a set of tract shapes (geometry) corresponding to acoustically plausible products. The vocalization is produced by the winner. Two systems were exploited to describe the vocal tract geometry (see Section 2.2): Xc, Ac, Al and Xh, Yh. Thus, a given vocalic sound could be associated with the mean and variance of these geometric variables in the group of corresponding tract shapes. As compensation leads to rather high variances, especially in central vocalizations, for clarity’s sake, we only displayed the dispersion ellipses of four “prototypes” added to each set of real data: [i a u] were chosen at positions roughly equivalent to those for adults in the MVS, whereas [ø] was produced by all commands set to zero. [i a u ø] thus served as landmarks.

Results

At four months (Fig. 11), the average tract shapes (plotted by grey stars on the figure) had highest tongue points rather centred and gathered (around [ø]). The constrictions were slightly fronted and fairly wide. At seven months (Fig. 12), the tongue positions showed a larger exploration of the high–low and front–back dimensions than at the earlier stage. Moreover, we found that before canonical babbling (four months), all the articulatory configurations leading to the first two formant frequencies falling within the [u] region had palatal constrictions. This result is of interest with regard to how the developmental path followed by articulatory exploration may shape adult speech. Although three types of tract constriction (palatal, velo-pharyngeal and pharyngeal) should be able to produce the vowel [u] with identical first three formants [Boë et al., 2000a], the only one found in the native (adult) speakers of all the languages tested is palatal [Wood, 1979]. The velo-pharyngeal [u] is seldom observed in perturbation experiments (lip-tube: Savariaux et al. [1995]) while the pharyngeal one has never been recorded. According to Abry and Badin [1996], the palatal [u] would be the first [u] production strategy picked during speech development: its dominant position in adulthood would stem from its early sensory–motor mapping. This hypothesis is in agreement with the palatal nature of the productions in the acoustic region around [u] in the simulations at four months.

3.5 Conclusion

The results of the simulation of vocal exploration in infants show that speech development does not begin with exhaustive exploration of the tract potential. We may suggest that “explore all possible speech sounds, then select what is necessary to communicate” would be a much more time- and energy-consuming strategy than, for instance, “explore, according to currently available abilities, and try to produce what is perceived in the ambient language just to develop the necessary motor skills”. The second strategy should provide a higher adaptive value than the first one, as more resources would be
left for the development of other biological functions. From an evolutionary point of view, this would account for the first strategy counter-selection.

To sum up, before canonical babbling, infants appear to use the lower lip height ($L_h$), tongue body ($T_b$) and dorsum ($T_d$) commands, which is consistent with newborn imitation studies. Furthermore, the importance of $T_d$ is in agreement with its likely role in suckling. The jaw articulator ($J$) would play only a minor role at this stage, and it would then become significant in canonical babbling.
4 Simulating early vocal imitation

In this section, we tried to simulate Kuhl and Meltzoff’s experiments on early vocal imitation [Kuhl and Meltzoff, 1996], which occurs, at least, before canonical babbling. The purpose was to gain some insights into the cognitive representations that might be involved in early vocal imitation and to test whether and how the robot could reproduce, at least, the actual infants’ imitation performance. First, an overview of Kuhl and Meltzoff’s experiment as well as a description of how the problem was translated into Bayesian terms will be given. Then, the implementation of imitation and the corresponding results will be presented.

4.1 An overview of Kuhl and Meltzoff’s experiment on early vocal imitation

Seventy-two subjects aged from 12 to 20 weeks old were exposed to audiovisual adult face–voice stimuli corresponding to the vowels [i], [a] and [u]. Only 45 of them happened to produce vowel-like utterances during the experiment.
Their subsequent vowel-like productions were phonetically and acoustically described. The system of transcription was that of the set of English vowels, but the transcribed items were merged into three main classes: the /a/-like, the /i/-like, and the /u/-like. Table 1 provides the resulting confusion matrix; that is, the number of /i/-like, /u/-like and /a/-like vocalizations (according to the criterion presented above) for each of the three possible adult targets [i a u]. In sum, the prebabblers produced vocalic sounds significantly more often categorized as being like the “target” after they had been exposed to this stimulus than otherwise. Globally, the subjects performed around 59% of responses that are congruent (hereafter %CR) with an imitative behaviour. Furthermore, about 16.5%, 47% and 36.5% of their vocalizations sounded /i/-, /a/- and /u/-like, respectively.

Table 1. The confusion matrix of early vocal imitation reported in Kuhl and Meltzoff [1996]. Each cell provides the number of /i/-like, /u/-like and /a/-like vocalizations (see text) for each of the three possible adult targets [i a u]. Among the 72 infants in the experiment, only 45 produced vowel-like utterances. Altogether, the 45 infants uttered 224 vowel-like vocalizations along the experiment.

<table>
<thead>
<tr>
<th></th>
<th>i</th>
<th>a</th>
<th>u</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>i-like</td>
<td>22</td>
<td>11</td>
<td>4</td>
<td>37</td>
</tr>
<tr>
<td>a-like</td>
<td>25</td>
<td>66</td>
<td>14</td>
<td>105</td>
</tr>
<tr>
<td>u-like</td>
<td>20</td>
<td>18</td>
<td>44</td>
<td>82</td>
</tr>
<tr>
<td>Total</td>
<td>67</td>
<td>95</td>
<td>62</td>
<td>224</td>
</tr>
</tbody>
</table>

4.2 Specifying the model

The Bayesian program describing the talking baby robot model is given in Fig. 13.

Variables

A large part of the work described earlier in this chapter was intended to select the relevant variables of the model.

The motor parameters chosen were selected according to the results of articulatory framing at four months (Section 3.3), i.e. the lower lip height (Lh), the tongue body (Tb) and dorsum (Td) commands.

The selected auditory variables (Section 2.2) were the first two formant frequencies ($F_1$, $F_2$) expressed in Bark.

The formants of a vocalic sound are functions of the tract shape; its mid-sagittal section can be described by three variables: the interlip area ($AI$) and the coordinates ($X_h$, $Y_h$) of the highest tongue point in a fixed system of reference. As mentioned in Section 2.2, $X_h$ and $Y_h$ are potential outputs of
the somesthetic system, and \( Al \) can be either a somatosensory or a visual variable (depending on whether this piece of information comes from the self or others).

The variables \( Lh, Tb, Td, F_1 \) and \( F_2 \) are discretized with 16 values each. Their ranges were fixed according to the findings of the different framings described above.

The variables \( Xh, Yh \) and \( Al \) are also discrete. We explore different discretizations for them: \( \{16, 16, 8\} \), \( \{8, 8, 4\} \), \( \{4, 4, 2\} \) and \( \{2, 2, 1\} \). These different choices will be discussed in detail in the sequel.

**Decomposition**

The three variables \( Xh, Yh \) and \( Al \) that characterize the geometric shape of the vocal duct were selected as pivots of the decomposition. Indeed, knowing the shape of the vocal duct, it is possible to deduce independently, on the one hand, the produced sound \( (F_1 \) and \( F_2) \) and, on the other hand, the motor commands \( (Lh, Tb \) and \( Td) \). The corresponding formal hypotheses are the following.

1. The three geometric variables \( Xh, Yh \) and \( Al \) are independent of one another when nothing is known about either the produced sound or the motor commands.

**Fig. 13. Talking baby robot Bayesian model**
2. Knowing the vocal duct shape is sufficient to predict the first formant.

\[ P(F_1 \mid Xh \land Yh \land Al) \] (2)

3. Knowing the vocal duct shape is also sufficient to predict the second formant. In particular, this second formant may be considered as conditionally independent of the first one when the shape is known.

\[ P(F_2 \mid Xh \land Yh \land Al) \] (3)

4. Knowing the interlip area is sufficient to infer the lip height.

\[ P(Lh \mid Al) \] (4)

5. Of course, knowing the position of the highest tongue point defines the position of the tongue’s body.

\[ P(Tb \mid Xh \land Yh) \] (5)

6. Knowing the tongue’s body and highest point gives information about the tongue’s dorsum.

\[ P(Td \mid Xh \land Yh \land Tb) \] (6)

**Parametric forms**

The distributions \( P(Xh) \), \( P(Yh) \) and \( P(Al) \) are assumed to be uniform.

The six other distributions are assumed to be Gaussians\(^3\). Their parameters (mean and standard deviation) are functions of the conditioning variables of each distribution.

**4.3 Learning the model**

To become an actual (and useful) description of the robot’s (and baby’s) sensory–motor behaviour, the distributions composing this probabilistic structure must be learnt from a set of experimental data.

These data were generated by a random exploration of the articulatory–geometrico–acoustic skills of the four-month-old baby, as specified in Section 3 (R 4m in the following). Values are drawn randomly for the three motor variables \( (Lh, Tb \text{ and } Td) \), then the VLAM direct model is used to generate the corresponding values of the geometric variables \( (Xh, Yh \text{ and } Al) \) and auditory variables \( (F_1 \text{ and } F_2) \). These data are then used to learn the means and standard deviations of all the Gaussians.

The robot’s “proficiency” in inversion—that is, in exploiting its map via Bayesian inference to draw motor values likely to make it reach a given

---

\(^3\) To be exact, as the variables are discretized, these distributions are “bell shapes”: the discrete approximation of Gaussian proposed by ProBT(r).
target-state of its perceptual variables—will mainly depend on the learning database size (DBS) and the degree of discretization of the geometric parameters (GDD).

We explore 15 different DBS sizes in the range from 1 to 60,000 items.

Indeed, as $Xh$, $Yh$ and $Al$ are the pivot of the description, the GDD partly determines the accuracy of the distributions the robot learns; it sets the minimal gap required to distinguish two items in the geometric domain and the size of the learning space; that is, the number of articulatory and auditory distributions that must be learned for the description to represent the whole range of the R4m abilities.

We explore four different GDDs in which the variables $Xh$, $Yh$ and $Al$ are respectively discretized to \{16, 16, 8\}, \{8, 8, 4\}, \{4, 4, 2\} and \{2, 2, 1\} possible different cases. This yields 2048, 256, 32 and 4 “boxes” in the geometric space, respectively. The number of boxes characterizes the somesthetic acuity of the robots (i.e. the number of different vocal duct shapes that the robot is able to distinguish and reach).

### 4.4 Trade-off between learning set and somesthetic accuracy

There is a trade-off between the GDD and the DBS because a given geometric box must include enough configurations for the matched motor and auditory distributions to be learned.

This trade-off is studied through the ability of the model to perform inversion of vocalizations in its exploration domain (i.e. answering the question $P(Lh \land Tb \land Td \mid f_1 \land f_2)$ for values $f_1$ and $f_2$ reachable in R4m). Figure 14 illustrates the results for the auditory inversion of 1000 items randomly chosen out of the R4m abilities.

![Fig. 14. Assessing the GDD/DBS trade-off. Mean formant error at the output of the inversion process (in Bark) as a function of the DBS (GDD as parameter).](image)
At maximal DBS, that is for the largest amount of learning, the error decreases, as the GDD increases, and reaches values lower than 0.5 Bk (roughly, formant jnd) for the highest two GDD values.

Moreover, for a given GDD, the error tends to decrease as the DBS rises, to a limit that is the lowest that the given GDD can produce.

However, all the GDDs except the roughest provide unstable scores as long as the DBS is below a certain value. This is because not all geometric boxes are actually learned (under-learning phase).

Indeed, the smallest DBS that is required to have an error at most 10% from the GDD-matched lowest error was found to be three times the size of the geometric space (in boxes). In other words, more boxes in the geometric space (a larger the GD) require more precise variables, thus requiring a larger DBS for the robot’s map to be representative of its sensory–motor skills (at least three times larger than the GDD).

4.5 Implementing auditory and audiovisual imitation

Once a model, defined by a given GDD, has been learned on a given DBS, it can be submitted to imitation tests.

Because the experimental data were obtained in an audiovisual configuration, we submitted the robot to two imitation tasks, audio-only and audiovisual imitation, to assess the role of multimodality in this framework.

In auditory imitation (hereafter A), the perceptual target is the \((f_1, f_2)\) pair of a vowel (question \(P(Lh \land Tb \land Td \mid f_1 \land f_2)\)), while in the audiovisual one (AV) it is \((f_1, f_2, al)\) triplet, including in the target the lips area (question \(P(Lh \land Tb \land Td \mid f_1 \land f_2 \land Al)\)).

Two target sets were the focus of imitation experiments, “external” and “internal” \([i a u]\) items.

The former corresponded to those of the four-month-old VLAM, the latter were their closest simulations within the R4m capacity of adult targets. This means that both target sets were adapted to the four-month articulatory–acoustic space (“normalized” targets), but the first one consisted of \([i]\), \([a]\) or \([u]\) targets outside the true vocalization space at four months, while the second one consisted of the three corners of this space.

For each target, 300 motor configurations were drawn from the distributions. The formants produced by each articulatory pattern were computed, and the sound was categorized as \([i]\), \([a]\) or \([u]\) according to its nearest target in the \((F_1, F_2)\) plan, in terms of Euclidean distance. This allowed us to compute congruent imitation scores \(\%CR\) for A and AV imitation, for both external and internal targets, and for various values of GDD and DBS.

4.6 A and AV imitation results

The congruent response scores \(\%CR\) as functions of the GDD and the DBS in the AV inversion of the internal and external \([i a u]\) targets are displayed
in Figures 15 and 16, respectively. A inversion scores, not displayed here, are systematically slightly lower. Furthermore, the following trends appear.

**Fig. 15.** %CR for the AV inversion of the “internal” [i a u] vowels, as a function of the DBS (GDD as parameter). “Infants” stands for the score obtained by 12–20-week-old infants in the study by Kuhl and Meltzoff [1996].

**Fig. 16.** %CR for the AV inversion of the “external” [i a u] vowels, as a function of the DBS (GDD as parameter). “Infants” stands for the score obtained by 12–20-week-old infants in the study by Kuhl and Meltzoff [1996].

**GDD/DBS Trade-off and under-learning**

Of course, the same GDD/DBS trade-off as in Fig. 14 is found in all cases.
Under-learning happens when the imitation scores are lower than their asymptotes for a given GDD (DBS not large enough for this GDD), and results in rather erratic behaviour of %CR scores. Globally, under-learning is greater for external than for internal targets, and it ends more quickly for AV than for A imitation.

External vs. internal targets: the risk of over-learning

The scores for external targets are lower than for internal ones, which is quite understandable, considering that the former are outside the R4m vocalization space while the latter are not.

More surprisingly, in the A case, the imitation score never reaches 100% with external targets even with the highest GDD and DBS configurations; that is, 2048 geometrical boxes and 60,000 items in the learning set. This is ascribable to the over-learning problem. Indeed, when the description is completely representative of the robot sensory–motor abilities (e.g. with a maximal DBS), all the distributions of the motor variables have small variances; that is, they are very accurate. However, none of them matches the target that the robot attempts to imitate. Hence, the system draws articulatory configurations regardless of their irrelevance given the sound.

In other words, the GDD should contain a small number of large boxes for the robot to be able to imitate vocalic sounds that are beyond its sensory–motor abilities.

The problem is overcome if visual information is also provided: because the VLAM [j a u] interlip areas belong to the R4m ones, the robot is enabled to select configurations that produce the nearest sounds to the target.

Early vocal imitation does not require much learning

Altogether, it is striking to notice that the robot requires neither a high GDD nor a large DBS to perform as well as, or even better than, actual infants.

For example, for external targets that are out of its motor abilities (which corresponds more closely with the experimental conditions of the imitation data in the Kuhl & Meltzoff study), it achieves 60% CR (as infants did) or more with a DBSs of 50 and 25 values and a GDD of 32 boxes, in the A and AV inversions, respectively.

4.7 Conclusion

The major lesson in this second study is that a very small number of vocalizations (less than a hundred) are necessary for a robotic learning process to provide imitation scores at least as high as those of 20-week-old infants. This is because the imitation task studied by Kuhl & Meltzoff is basically a three-categories problem, which can be described rather simply and roughly in articulatory–acoustic terms; hence the success of the present robotic experiment. This shows that actually, more than learning, the problem is of course
control – achieving a desired articulatory configuration – which the infant is
not able to do easily at four months.

The A and AV imitation experiments displayed a trade-off between the
somesthetic acuity of the tract shape representation (GDD) and the amount
of information (DBS) required to build a sensory–motor map that is repre-
sentative enough of the robot skills. Furthermore, our results show that the
GDD must be rough for the robot to be able to imitate vocalic sounds that
are out of its articulatory–acoustic abilities. This is interesting because, in
fact, infants must acquire, by imitation, the speech sounds of their ambient
languages although they are not endowed from birth with the matching motor
capacity. Moreover, this investigation supports the view that the forma-
tion of the cognitive representation likely to underlie early vocal imitation would
require less learning with audiovisual speech perception than without vision.
This gives some evidence that the latter can facilitate phonetic development
and is congruent with the slight differences in speech development between
seeing and visually impaired children [Mills, 1987].

Overall, this preliminary work confirms that infants complement their very
early visuo-facial imitation abilities by using auditory-to-articulatory relation-
ships, and shows that a very small amount of data is enough to produce
realistic imitation scores, if the discretization is rough enough.

5 Perspectives in the study of ontogeny and phylogeny

The experiments described here anchor both the production and the per-
ceptual representations of the baby robot in actual infants’ perceptuo-motor
ground.

The continuation of this work will allow the robot to grow up, mimicking as
much as possible the developmental process at work in human speech acquisi-
tion. This involves the various steps described in Section 1.3, and particularly
the acquisition of frame and content control in the production of syllables
[Davis and MacNeilage, 1995, MacNeilage, 1998]. Throughout this process,
an important output of the work will be information about the perceptual
and motor representations acquired by the system at the various developmen-
tal stages. In a way, it should provide a window on the representations of
speech in the baby’s and child’s brain, which cannot be directly observed by
simple means.

Another challenge will be to study how speech as a linguistic system may
be patterned by both perceptual and motor constraints. This route towards a
“substance-based” approach to phonology that simulates speech phylogeny is
not new. One of its precursors is found within the adaptive variability theory
of Lindblom and colleagues, with a number of important results on the predic-
tion of vowel systems (see e.g. Liljencrants and Lindblom [1972] or Lindblom
[1986, 1990] and the extension that Schwartz et al. [1997] proposed through
the “dispersion–focalization theory”) and of consonant systems (e.g. Lindblom
More recently, Steels and others introduced the concept of speech games in societies of talking agents (e.g., Steels [1998], Berrah et al. [1996] and De Boer [2000]). The definition of more realistic agents, able to act, perceive and learn in a biologically, developmentally and cognitively plausible way, is crucial in that context.

Integrating perception and action within a coherent computational framework is a natural way to understand better how speech representations are acquired, how perception controls action and how action constrains perception. This provides also a natural framework to integrate various sources of knowledge about the speech process, including behavioural and developmental data, neurophysiological and neuropsychological facts about the neural circuits of perception, action and language, and linguistic knowledge about phonology or syntax, and to attempt to draw some links between these complex ingredients to begin writing the story of the emergence of human language. We believe that modelling speech communication in a robotic framework should contribute to a computational approach, which is relevant for future progress in the study of speech and language ontogeny and phylogeny.

Acknowledgements

This chapter is an adapted version of a paper published in Interaction Studies under the same title. It is republished here with the explicit authorization of John Benjamins Publishing Company. We thank them.

This work was prepared with support from the European ESF Eurocores program OMLL, and from the French funding programs CNRS STIC Robea and CNRS SHS OHLL, and MESR ACI Neurosciences Fonctionnelles.

It benefited from discussions with and suggestions from Louis-Jean Boë, Barbara Davis, Chris Matyear, Emmanuel Mazer and Christian Abry.

References


J.-L. Schwartz, J. Robert-Ribes, and P. Escudier. Ten years after summerfield ... a taxonomy of models for audiovisual fusion in speech perception. In B. D.


Index

<table>
<thead>
<tr>
<th>Term</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>A priori</td>
<td>30, 34, 36, 39, 59, 104, 107, 110, 111, 114, 175, 176, 187, 188, 192, 194, 197, 201, 208, 221, 236, 259, 261, 265, 298, 300, 301, 305, 309–312</td>
</tr>
<tr>
<td>Accident</td>
<td>8, 79</td>
</tr>
<tr>
<td>Action</td>
<td>9, 13, 185, 208</td>
</tr>
<tr>
<td>Actuator</td>
<td>3, 6, 196</td>
</tr>
<tr>
<td>Adaptive cruise control</td>
<td>8, 79</td>
</tr>
<tr>
<td>Algorithm</td>
<td></td>
</tr>
<tr>
<td>Baum-Welch</td>
<td>286–288</td>
</tr>
<tr>
<td>BOF (Bayesian Occupency Filter)</td>
<td>8, 79–85, 87–94, 97–100</td>
</tr>
<tr>
<td>EM (Expectation-Maximization)</td>
<td>111, 112, 120, 286</td>
</tr>
<tr>
<td>Generalize Distributive law</td>
<td>46</td>
</tr>
<tr>
<td>Global alignment</td>
<td>111, 113–116, 118, 119</td>
</tr>
<tr>
<td>Global alignment with uncertainty</td>
<td>113, 115, 118, 119</td>
</tr>
<tr>
<td>K-mismatch</td>
<td>113</td>
</tr>
<tr>
<td>Least-square</td>
<td>217</td>
</tr>
<tr>
<td>MCSEEM (Monte Carlo Simultaneous Estimation and Maximisation)</td>
<td>10, 46, 215, 216, 225, 229, 230, 240</td>
</tr>
<tr>
<td>Monte Carlo Markov Localization</td>
<td>165</td>
</tr>
<tr>
<td>MRBT (Multi-Resolution Binary Tree)</td>
<td>46</td>
</tr>
<tr>
<td>Particle filter</td>
<td>13, 14, 63, 68, 163, 173, 195, 209, 313</td>
</tr>
<tr>
<td>POMDP (Partially Observable Markov Decision Process)</td>
<td>13, 104, 118, 119, 121, 126, 129, 162, 163, 171, 190, 191, 195</td>
</tr>
<tr>
<td>SIFT (Scale-Invariant Feature Transform)</td>
<td>105</td>
</tr>
<tr>
<td>SLAM (Simultaneous Localization and Mapping)</td>
<td>8, 80, 104–106, 116, 120, 121, 127, 129</td>
</tr>
<tr>
<td>SRA (Successive Restriction Algorithm)</td>
<td>46</td>
</tr>
<tr>
<td>Allocentric</td>
<td>167, 318, 319, 326, 329</td>
</tr>
<tr>
<td>Ambiguities</td>
<td>12, 74, 135, 136, 138, 326, 332</td>
</tr>
<tr>
<td>Ant</td>
<td>166, 168</td>
</tr>
<tr>
<td>API</td>
<td>23, 44, 47, 267</td>
</tr>
<tr>
<td>Artificial Intelligence</td>
<td>4, 13, 140, 225</td>
</tr>
<tr>
<td>Atlas</td>
<td>10, 246–272</td>
</tr>
<tr>
<td>Attention</td>
<td>9, 185, 194, 199–204, 207–209</td>
</tr>
</tbody>
</table>
Index

Focusing 9, 185
Automotive Industry 7, 8, 79
Autonomous X, 4, 9, 53, 57, 75, 94, 96, 103, 104, 116, 123, 136–138, 185, 188, 208, 275, 276
Navigation 53, 103, 104
Robotics 4, 188, 208
Avatar 10, 274–277, 279, 281, 284, 286–288

Babbling 11, 347–350, 357, 361–363
BACS (Bayesian Approach to Cognitive Systems) X, 275
Bayesian
CAD systems 9, 10, 215, 216, 219, 225, 229, 231–233, 239, 240
Filter 8, 14, 49, 67–69, 81, 82, 91–93, 163
Maps 9, 160, 162, 163, 171–180, 208, 209
Net 8, 13, 14, 48, 49, 135–154, 156, 157, 163, 190, 217, 260
Occupancy filter 8, 79–85, 87–94, 97–100
Programming X, 7, 14, 21, 37, 38, 44, 48, 49, 89, 109, 110, 135, 136, 140–142, 185, 192, 318, 321, 347
Subroutines 37
Behaviour
Animal IX
Avoidance 38, 96, 97
Combination 38, 172
Homina 38, 41–43
Phototaxy 38
Reactive 28, 31, 106, 168, 277, 278, 280, 283, 284
Belief 118, 120, 150, 195

BIBA (Bayesian Inspired Brain and Artefacts) X, 122, 181, 204, 205, 209, 275
CAD 7, 9, 80, 215, 229, 233
Calibration 13, 60, 216, 235–237
Camera 8, 27, 53, 54, 94, 98, 99, 104, 120, 122, 123, 136, 154, 149, 313
Cartesian 55, 58, 84, 92, 161, 165, 170, 179
Maps 161, 179
Cataglyphis 166
Causality 4, 141
Closest Point of Approach
Distance 96
Time 96
Closest point of approach 95, 96
Closing the loop 120, 126
Cognition 8
Cognitive map 103, 105, 127, 128, 166, 167
Cognitive science 12
Collision 8, 55, 74, 79, 80, 93–97, 240, 244
Conditional independence 25, 30, 37, 191, 192, 279
Confidence 55, 56, 68–71, 73, 74, 88, 140
Conflicts 13, 353
Conjunction 21–26, 29, 41, 45–47, 64, 279, 353
Cortex 106, 128
CPA see Closest point of approach
Cruise control see Adaptive cruise control
Cycab 53, 54, 56, 57, 73, 86, 88, 94, 96, 97

Data
Index  381

Association  8, 61, 63, 65, 79–81, 98, 100
DBN (Dynamic Bayesian Network)  14, 138, 163, 164, 190
Disjunction  21, 22
Distribution
  Dirac  35, 36, 58, 87, 88, 90, 91, 240, 258, 259, 262, 301, 302
  Gaussian mixture  111–113, 145, 150
Joint probability  24, 44, 85, 112, 141, 144, 319, 340
Uniform  30, 60, 61, 66, 69, 70, 85, 86, 111, 145, 173, 176, 178, 196, 200, 224, 236
DNA  114
Dynamic Bayesian Network  see DBN (Dynamic Bayesian Network)
Dynamic Environment  72, 79–81, 169, 170
Dynamic Programming  114
Echographic  10, 246, 247, 251–259, 261–263, 265–271
Egocentric  55, 168, 294, 297, 298, 300
Entropy  12, 14, 66, 119, 120, 173
Concentration theorem  12
MaxEnt  12, 14
EPFL (Ecole Polytechnique Fédérale de Lausanne)  X, 27, 122, 123, 125
Experimental Data  6, 23, 25, 144, 145, 238, 329, 331, 332, 366, 368
Eye  11, 136, 148, 157, 185, 208, 293, 294, 296, 297, 321, 322, 325, 326, 337, 339, 353
Particle  13, 14, 63, 68, 163, 173, 195, 209, 313
Fingerprint  8, 103, 104, 106–111, 113, 115–123, 126–130
Fusion  8, 12, 33–38, 47, 71, 80, 100, 138, 150, 172, 217, 279, 310, 335, 353
Data  38, 71, 217
sensor  33–35, 38, 47, 80, 100, 138, 150, 187, 279
Fuzzy Logic  71
Game  10, 274–278, 280–288, 372
garbage model  146
Gateways  105
Gaussian Mixture  111–113, 145, 150
Gaze  10, 149, 293, 295, 330, 336
Uncertainties  9, 215–217, 231–234
<table>
<thead>
<tr>
<th>Category</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gesture Recognition</td>
<td>138, 157</td>
</tr>
<tr>
<td>GPS (Global Positioning System)</td>
<td>53, 123</td>
</tr>
<tr>
<td>Haptic</td>
<td>12, 13</td>
</tr>
<tr>
<td>Head direction cell</td>
<td>166, 167</td>
</tr>
<tr>
<td>Hidden variable</td>
<td>3, 6, 7, 137</td>
</tr>
<tr>
<td>Hierarchies</td>
<td>X, 9, 13, 25, 37, 162, 168, 169, 171, 174, 180</td>
</tr>
<tr>
<td>Hippocampus</td>
<td>104, 127–129, 166, 167</td>
</tr>
<tr>
<td>Ignorance</td>
<td>4</td>
</tr>
<tr>
<td>Ill-posed problem</td>
<td>12, 38, 317</td>
</tr>
<tr>
<td>Illusion</td>
<td>12, 140, 151, 275, 295, 296</td>
</tr>
<tr>
<td>Imitation</td>
<td>11, 347, 349, 350, 357, 362–364, 368, 370, 371</td>
</tr>
<tr>
<td>Incomplete</td>
<td>3, 5, 10, 135–137, 170, 243, 253</td>
</tr>
<tr>
<td>Incompleteness</td>
<td>3–6, 12, 14, 44, 136, 157</td>
</tr>
<tr>
<td>Industrial</td>
<td>4, 277, 288</td>
</tr>
<tr>
<td>Industry</td>
<td>7, 8, 79, 283, 288</td>
</tr>
<tr>
<td>Inference</td>
<td>IX, X, 5, 7, 8, 10, 12–14, 21–23, 26, 37, 44, 46, 72, 73, 85, 91, 92, 100, 136, 138, 139, 165, 176, 195, 202, 222, 225, 229, 267–269, 271, 297, 298, 301, 312, 313, 320, 324, 340, 366</td>
</tr>
<tr>
<td>By imitation</td>
<td>157</td>
</tr>
<tr>
<td>Inverse</td>
<td>9, 10, 12, 38, 215, 216, 275, 277, 278, 280, 283, 288, 317</td>
</tr>
<tr>
<td>Programming</td>
<td>10, 275, 277, 278, 280, 283, 288</td>
</tr>
<tr>
<td>Jaw</td>
<td>11, 347, 349, 350, 353, 359, 362</td>
</tr>
<tr>
<td>Kalman</td>
<td>13, 83, 105, 136, 162, 163, 165, 173, 174, 178, 209, 217, 302, 313</td>
</tr>
<tr>
<td>Extended Kalman Filter</td>
<td>105, 136, 313</td>
</tr>
<tr>
<td>Filter</td>
<td>13, 14, 83, 105, 136, 162, 163, 165, 173, 174, 178, 217, 302, 313</td>
</tr>
<tr>
<td>Khepera</td>
<td>21, 26–29, 31, 34, 41, 43</td>
</tr>
<tr>
<td>Kidnapping</td>
<td>122, 126</td>
</tr>
<tr>
<td>Kinematic graph</td>
<td>217–219, 231, 233, 236, 238, 240, 255, 256</td>
</tr>
<tr>
<td>Kinematics</td>
<td>216, 233, 235</td>
</tr>
<tr>
<td>Koala</td>
<td>163, 176</td>
</tr>
<tr>
<td>Larynx</td>
<td>11, 347, 350</td>
</tr>
<tr>
<td>Learn</td>
<td>28, 31, 103, 127, 139, 147, 276, 277, 348, 349, 354, 366, 367, 372</td>
</tr>
<tr>
<td>Logical proposition</td>
<td>21, 22</td>
</tr>
</tbody>
</table>

---

The index provides a list of topics and their associated page numbers, grouped under various categories such as Haptic, Heuristic, Hidden variable, etc.
MAP (Maximum A Posteriori) 46, 221
Marginalization 23, 41, 45, 93, 320
Decision process (MDP) 118, 163, 171, 195
HMM (Hidden Markov Models - Input/output) 164, 188, 190
HMM (Hidden Markov Models) 13, 14, 49, 136, 162, 163, 171, 188
Localization 13, 162–165, 171, 173, 174, 188, 190, 191
Partially Observable Decision Process (POMDP) 13, 104, 118, 119, 121, 126, 129, 163, 171, 190, 191, 195
Mesh 10, 219, 231, 246, 247, 252, 253, 269–271
Mixture 14, 44, 111–113, 145, 150
Modalities 138
Modality 12, 130, 136, 157, 317, 350
Model
Articulatory 11, 347, 350, 351
Comparison 68
Graphical 13, 46
Mixture 14, 44
Observation 64, 69, 70, 91, 163, 165, 195, 196, 200
probabilistic IX, X, 14, 23, 24, 70, 136, 162, 163, 165, 170, 171, 180, 217
Symbolic 136
Transition 69, 163
Variable Linear Articulatory 350, 351, 357, 358, 302, 366, 368, 370
Modeller 9, 29, 215, 216, 229, 233, 297, 340
Models 136
Neuromimetic 136, 137
Probabilistic 136
Symbolic 136
Parallax 11, 317
Movement 10, 28, 41, 55, 68, 71, 82, 86, 103, 168, 178, 238, 244, 245, 255, 293–297, 318, 327, 333, 335, 337, 349, 350, 354
MRI 10, 244, 245, 270
Negation 21, 22
Nervous system IX, 167, 207, 352
Neurobiology 103, 127, 129
Noise 4, 82, 83, 98, 105, 278, 298–300, 302, 305, 312, 313, 341
Noisy 4, 7, 305, 353
Normalization 22, 23, 45, 111, 119, 258, 263, 301
Object
Composite 152
Object recognition 8, 135–138, 140, 148, 157
Observation model see Model – Observation
Obstacle  27–32, 38, 39, 41, 42, 55, 57, 72, 74, 75, 80, 81, 87, 96, 204, 205
Avoidance  28, 38, 72, 74, 75, 205
Oclusion  8, 79–82, 87, 97, 98, 100, 114, 115
Occupancy  8, 13, 79–94, 96, 98, 100, 105, 106
Odometry  106, 169, 312
Optical flow  11
Outlier  60–63
OVAR  11, 306, 308, 309, 311, 313
Palate  11, 347, 353, 354
Parallel Map theory (PMT)  167, 168
Path integration  168
Pedestrians  57, 98
Perception-space
Tracking  53
Perceptual aliasing  66, 68, 104, 106, 107, 129, 179
Phenomenon  3, 23, 24, 32, 81, 177–179, 189, 196, 237, 279, 293, 317
Phylogenetic  167
Place cell  127, 128, 166, 167
Preliminary knowledge  6, 7, 22, 32, 110, 143, 144
Dependencies  143
Preprocessing  138, 140
Probabilistic
Computation  IX, 23, 313
If-then-else  42, 44
Inference  IX, 5, 12, 46, 136
Logic  IX
Model  IX, X, 14, 23, 24, 70, 136, 162, 163, 165, 170, 171, 180, 217
Reasoning  IX, 12, 22, 77, 80, 81
Probability as logic  12
ProBAYES  XV, XVI, 48, 79, 185, 215, 243
ProBT  23, 30, 44, 46–49, 267, 268, 324, 366
Programmer  3, 4, 6, 7, 23, 24, 29, 32, 39, 137, 171, 172, 175, 215, 282, 283
Psychophysics  324
Radiographic  10, 246, 247, 250, 252–259, 261, 265–271, 350
Rat  103, 127, 129, 166
Rational  3, 5, 7
Resolution principle  7, 23
Retina  11, 208, 317, 334
Road  8, 79, 97
Rule
Conjunction  22, 23, 25, 29, 41, 45
Marginalization  23, 41, 45, 320
Normalization  22, 23
Safe travelling distance  95
Index

Vocal tract 11, 347, 349–351, 355, 361
Vocalization 11, 347, 349, 350, 357–364, 367, 368, 370
Voronoi graph 105, 106
X-ray 10, 252

servoing 53

Visuo-vestibular 10–12, 293, 296–298, 301, 303, 306, 310, 313
Authors

Abbas 12, 17
Abry 348-350, 352, 353, 361, 371–373, 376, 377
Adams 138, 158
Adelson 12, 19
Adobbati 278, 289
Aho 113, 130
Ahuactzin 46, 50, 171, 182, 225, 229, 241
Aji 13, 14, 46, 50
Alais 13, 14
Alami 217, 240
Albright 13, 14
Amavizca-Ruiz 247, 248, 272
Anastasio 12, 14
Angelaki 296, 305, 306, 309, 312–314
Arai 305, 306, 313
Arkin 53, 77
Arleo 128, 130
Arnborg 12, 14
Arras 80, 101, 106, 132, 174, 184
Arrigoni 275, 289
Arrouas 352, 376
Aslin 13, 14
Aspert 139, 140, 158
Atal 355, 376
Attias 173, 181
Augeras 217, 242
Aycard 13, 14
Badin 348, 349, 361, 372, 373, 377
Baeza-Yates 113, 130
Bailly 349, 356, 373
Banks 12–16, 335, 344
Bar-Shalom 79, 101, 217, 240
Battaglia 13, 14, 128, 130
Baum 286, 289
Bayes 12, 14
Beal 13, 18, 136, 157
Beaupreps 352, 361, 373, 376
Becker 136, 140, 157, 317, 345
Beeson 106, 130
Bekey 13, 19
Belkacem-Boussaid 12, 14
Bennewitz 165, 183
Benoit 352, 372
Benson 296, 314
Bernhardt 13, 14
Bernstein 207, 209
Berrah 348, 372, 373
Berthommier 352, 376
Berthoz 166, 169, 180, 184, 208, 209, 297, 314
Bilmes 112, 130
Bishop 83, 101, 136, 159
Biswa 68, 78
Bladon 353, 373
Bodin 296, 314
<table>
<thead>
<tr>
<th>Authors</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boë</td>
<td>348, 350–357, 361, 371–373, 376, 377</td>
</tr>
<tr>
<td>Boismain</td>
<td>13, 18</td>
</tr>
<tr>
<td>Bonasso</td>
<td>53, 77, 138, 158</td>
</tr>
<tr>
<td>Borrelly</td>
<td>13, 15</td>
</tr>
<tr>
<td>Bosma</td>
<td>353, 373</td>
</tr>
<tr>
<td>Bothorel</td>
<td>350, 373</td>
</tr>
<tr>
<td>Bourdoncle</td>
<td>327, 335, 337, 341, 342, 344</td>
</tr>
<tr>
<td>Boutillier</td>
<td>163, 171, 181</td>
</tr>
<tr>
<td>Boyen</td>
<td>195, 209</td>
</tr>
<tr>
<td>Boyle</td>
<td>298, 303, 315</td>
</tr>
<tr>
<td>Bozma</td>
<td>203, 210</td>
</tr>
<tr>
<td>Braiman</td>
<td>13, 15</td>
</tr>
<tr>
<td>Brailon</td>
<td>71, 72, 78</td>
</tr>
<tr>
<td>Breazeal</td>
<td>348, 374</td>
</tr>
<tr>
<td>Bressan</td>
<td>317, 344</td>
</tr>
<tr>
<td>Brethorst</td>
<td>12, 15</td>
</tr>
<tr>
<td>Brooks</td>
<td>44, 50, 348, 374</td>
</tr>
<tr>
<td>Bruyninckx</td>
<td>203, 210</td>
</tr>
<tr>
<td>Bücken</td>
<td>165, 183</td>
</tr>
<tr>
<td>Bühlhoff</td>
<td>12, 20</td>
</tr>
<tr>
<td>Burgard</td>
<td>13, 14, 16, 19, 77, 165, 181, 183, 190, 203, 209, 222, 241</td>
</tr>
<tr>
<td>Burgess</td>
<td>128, 131, 167, 181</td>
</tr>
<tr>
<td>Burnham</td>
<td>353, 374</td>
</tr>
<tr>
<td>Burr</td>
<td>13, 14</td>
</tr>
<tr>
<td>Cacucci</td>
<td>128, 131</td>
</tr>
<tr>
<td>Campbell</td>
<td>353, 374</td>
</tr>
<tr>
<td>Caplat</td>
<td>136, 157</td>
</tr>
<tr>
<td>Cassandra</td>
<td>13, 15, 17, 80, 101, 118, 119, 130, 162, 182, 191, 209, 210</td>
</tr>
<tr>
<td>Castellanos</td>
<td>105, 130</td>
</tr>
<tr>
<td>Castle</td>
<td>12, 17</td>
</tr>
<tr>
<td>Cathiard</td>
<td>348, 353, 373, 377</td>
</tr>
<tr>
<td>Caudek</td>
<td>331–333, 344</td>
</tr>
<tr>
<td>Chase</td>
<td>354, 375</td>
</tr>
<tr>
<td>Chatila</td>
<td>53, 78, 105, 106, 131, 132</td>
</tr>
<tr>
<td>Chaumette</td>
<td>53, 77, 78</td>
</tr>
<tr>
<td>Cheeseman</td>
<td>105, 132</td>
</tr>
<tr>
<td>Chen</td>
<td>13, 19, 82, 88, 101</td>
</tr>
<tr>
<td>Chistovich</td>
<td>352, 374</td>
</tr>
<tr>
<td>Cho</td>
<td>127, 130</td>
</tr>
<tr>
<td>Choquet</td>
<td>105, 106, 120, 130, 131</td>
</tr>
<tr>
<td>Chou</td>
<td>13, 17</td>
</tr>
<tr>
<td>Clark</td>
<td>105, 130, 168, 183</td>
</tr>
<tr>
<td>Clément</td>
<td>297, 314</td>
</tr>
<tr>
<td>Cohen</td>
<td>296, 297, 300, 303, 305, 306, 309, 314, 315</td>
</tr>
<tr>
<td>Colas</td>
<td>192, 198, 210</td>
</tr>
<tr>
<td>Conradt</td>
<td>203, 210</td>
</tr>
<tr>
<td>Cooper</td>
<td>13, 15, 46, 50</td>
</tr>
<tr>
<td>Corbillon</td>
<td>243, 272</td>
</tr>
<tr>
<td>Cornilleau-Pérès</td>
<td>318, 327, 329, 335, 337, 341, 342, 344</td>
</tr>
<tr>
<td>Coste</td>
<td>13, 15</td>
</tr>
<tr>
<td>Coué</td>
<td>79, 82, 88, 100, 101, 217, 240</td>
</tr>
<tr>
<td>Cox</td>
<td>12, 15, 162, 163, 182</td>
</tr>
<tr>
<td>Cremers</td>
<td>165, 181, 183, 190, 209</td>
</tr>
<tr>
<td>Cressant</td>
<td>128, 130</td>
</tr>
<tr>
<td>Csapó</td>
<td>105, 130</td>
</tr>
<tr>
<td>Dagum</td>
<td>46, 50</td>
</tr>
<tr>
<td>Darlot</td>
<td>12, 20, 297, 315</td>
</tr>
<tr>
<td>Darwiche</td>
<td>13, 15</td>
</tr>
<tr>
<td>Davalo</td>
<td>136, 158</td>
</tr>
<tr>
<td>Davis</td>
<td>349, 350, 357, 371, 374, 375, 377</td>
</tr>
<tr>
<td>Dayan</td>
<td>128, 131</td>
</tr>
<tr>
<td>De Boer</td>
<td>372, 374</td>
</tr>
<tr>
<td>Dean</td>
<td>163, 171, 181</td>
</tr>
<tr>
<td>Dedieu</td>
<td>14, 15, 170, 181</td>
</tr>
<tr>
<td>Dekhil</td>
<td>13, 15</td>
</tr>
<tr>
<td>Delcher</td>
<td>13, 15</td>
</tr>
<tr>
<td>Delessert</td>
<td>136, 158</td>
</tr>
<tr>
<td>Delgrutte</td>
<td>353, 374</td>
</tr>
<tr>
<td>Dellaert</td>
<td>13, 16, 165, 181, 183, 222, 241</td>
</tr>
<tr>
<td>Demoment</td>
<td>12, 18</td>
</tr>
<tr>
<td>Dempster</td>
<td>286, 289</td>
</tr>
<tr>
<td>Des Tuves</td>
<td>13, 18</td>
</tr>
<tr>
<td>Diard</td>
<td>14, 18, 21, 49, 50, 136, 158, 162–164, 172, 178, 181–183, 191, 209, 318, 324, 335, 344</td>
</tr>
<tr>
<td>Dichgans</td>
<td>303, 314</td>
</tr>
<tr>
<td>Dickman</td>
<td>306, 312, 314</td>
</tr>
<tr>
<td>Dijkstra</td>
<td>318, 327, 329, 344</td>
</tr>
<tr>
<td>Dissanayake</td>
<td>105, 130</td>
</tr>
<tr>
<td>Dobler</td>
<td>140, 159</td>
</tr>
<tr>
<td>Dodd</td>
<td>353, 374</td>
</tr>
<tr>
<td>Domini</td>
<td>331–333, 344</td>
</tr>
<tr>
<td>Donald</td>
<td>13, 15</td>
</tr>
<tr>
<td>Dose</td>
<td>12, 18</td>
</tr>
<tr>
<td>Dostrovsky</td>
<td>127, 129, 132</td>
</tr>
<tr>
<td>Douglas</td>
<td>108, 131</td>
</tr>
<tr>
<td>Douillard</td>
<td>13, 18</td>
</tr>
<tr>
<td>Drewing</td>
<td>12, 15</td>
</tr>
<tr>
<td>Duboudieu</td>
<td>13, 18</td>
</tr>
</tbody>
</table>
Dufourd 106, 131
Durrant-Whyte 105, 130, 131, 162, 163, 165, 182

Eaton 139, 158
Eichenbaum 127, 130
Elves 85, 101
Erickson 12, 15, 19
Ernst 12, 13, 15, 335, 344
Escudier 348, 352, 376, 377
Espiau 13, 15

Fant 356, 374
Fink 138, 158
Fisher 12, 18
Florez-Larrahalondo 286, 289
Fortman 79, 101
Fortmann 217, 240
Fougère 12, 15
Fox 13, 14, 16, 19, 72, 77, 165, 181, 183, 190, 203, 209, 222, 241
Fraichard 71, 77, 79, 82, 88, 100, 101, 217, 240
Franz 168–170, 181
Freeman 136, 138–140, 158, 159, 337–339, 344
Frey 13, 16
Fritsch 138, 158
Fröhlinghaus 165, 183
Fry 12, 16

Gabioud 348, 350, 355, 373, 374
Gadeyne 203, 210
Gallistel 116, 131
Garnier 71, 77
Garrido 13, 19
Geisler 12, 16
Geoffroy 13, 18
Gepshtein 12, 16
Gerstner 128, 130
Geweke 222–224, 241
Ghahramani 13, 17, 163, 183
Gielen 318, 327, 329, 344
Giese 127, 130
Giralt 53, 78
Glasauer 297, 314
Glauser 122, 131
Glotin 348, 372, 373
Goldstein 351, 374
Gondran 219, 241
Gopnik 13, 16
Gordon 302, 314
Gothard 128, 131, 166, 181
Graham 317, 344
Grandy 12, 19
Gray 294, 314
Grefenstette 225, 241
Grootenboer 13, 19
Grove 13, 15
Guedry 293, 296, 311, 314
Guenther 349, 374
Guépin 357, 373
Guérin 357, 373
Guiard Marigny 350, 374
Gusev 13, 16
Guttmann 13, 16
Hafner 105, 131
Hähnel 165, 181, 183
Hall 355, 376
Halpern 12, 16
Hans 163, 181
Hanson 12, 16
Hardcastle 353, 374
Harnad 4, 16
Hartley 128, 131, 167, 181
Hassan 245, 272
Hauskrecht 171, 181
Hayes 13, 18
Heckerman 163, 183
Heidbreder 12, 16
Henderson 13, 15
Hennig 165, 183
Herbert 168, 183
Hermosillo 53, 71, 72, 77, 78
Hess 296, 305, 306, 309, 313, 314
Highstein 298, 303, 315
Hillis 12, 16
Hofmann 165, 183
Holland 225, 241
Hoole 354, 374
Huber 138, 158
İstefanopulos 203, 210
Jaakkola 13, 16, 17, 19
Jackson 136, 157
Jacobs 12–14, 16, 19, 44, 50, 167, 182
Jacquemier 266, 272
Jakobson 349, 374
Jaynes 5, 12, 16, 17, 170, 182, 251, 272
Jazwinsky 81, 82, 101
Jeka 13, 17
Jensen 113, 131, 140, 158, 217, 241
Johnston 335, 344
Jong 106, 130
Jordan 13, 16, 17, 19, 44, 50, 163, 183
Juang 162, 163, 183
Kaelbling 13, 15, 17, 80, 101, 118, 119, 130, 162, 171, 181, 182, 191, 195, 209, 210
Kali 128, 131
Kalman 83, 101
Kaminka 278, 289
Kanade 113, 131
Kandel 351, 376
Kapellos 13, 15
Kasif 13, 15
Kavraki 171, 182
Keller 222, 241
Kent 349, 353, 374
Kippen 253, 261, 266, 272
Kersten 12, 16, 17
Kiefer 245, 272
Kiemel 13, 17
Klein 71, 77
Knill 12, 17
Knuth 12, 17
Koenderik 317, 344
Koenig 13, 17, 191, 210
Koike 71, 72, 78, 189, 193, 194, 206, 210
Koller 13, 17, 68, 78
Konolige 13, 16, 17
Koopman 13, 19
Koopmans-Beinum 349, 374
Körding 12, 17
Kortenkamp 53, 77, 105, 131, 138, 158
Krell 165, 183
Kruppa 13, 16
Kshirsargar 253, 272
Kuhl 349, 357, 363, 364, 369, 375
Kuipers 105, 106, 130, 131, 168, 170, 182
Kurien 13, 15, 118, 119, 130
Labhart 166, 168, 182
Laboissière 348, 361, 372, 373, 375
Laird 286, 289
Lakemeyer 165, 181
Lallouache 352, 376
Lambrinos 166, 168, 182
Lamiraux 53, 77
Lamon 113, 122, 131
Lamouret 318, 329, 333, 334, 345
Landgren 353, 375
Landy 12, 16, 335, 344
Lane 171, 182, 195, 210
Laplace 12, 17
Latombe 13, 15, 171, 182
Laugier 53, 71, 72, 77, 78, 82, 88, 100, 101, 217, 242
Laumond 53, 77, 78
Laurens 298, 312, 314
Lauritzen 13, 17, 140, 158
Lawton 165, 182
Le Hy 275, 289
Lebel 14, 15, 18, 21, 50, 136, 158, 170, 172, 181, 182, 198, 210, 275, 289, 318, 324, 335, 344
Lefebvre 203, 210
Leitner 245, 273
Leonard 105, 131, 162, 163, 165, 182
Lerner 68, 78
Lever 128, 131
Levitt 165, 182
Lewinnek 245, 272
Liljencrants 371, 375
Lindblom 354, 371, 375
Linden 12, 18
Lisien 106, 131
Little 105, 132
Littman 13, 17, 80, 101, 162, 182, 191, 209, 210
Loevenbruck 348, 373
Longuet-Higgins 317, 321, 344
López-de Teruel 13, 19
Lowe 105, 132
Lozano-Pérez 13, 18, 217, 241
Lubker 354, 375
Luby 46, 50
Luzeaux 106, 131
MacDonald 353, 376
MacKay 222, 241
Mackenzie Beck 351, 375
MacNeilage 349, 354, 357, 371, 374, 375
Maceda 350, 351, 373, 375
Maes 44, 50
Malis 53, 78
Mallot 168–170, 181
Maloney 335, 344
Manassian 12, 17
Marin 327, 335, 337, 341, 342, 344
Marjanovic 348, 374
Marshall 278, 289
Martin 353, 374
Martinelli 106, 132
Maskell 302, 314
Mason 13, 18
Matalon 12, 18
Matsuo 296, 297, 303, 305, 309, 314, 315
Matyasek 350, 357, 375, 377
Mayhew 317, 344
Mayne 312, 314
McAllister 354, 375
McEliece 13, 14, 46, 50
McGuire 138, 158
McGurk 353, 376
McLachlan 44, 50
McNaughton 128–132, 166, 181
Mekhnacha 14, 15, 46, 50, 80, 82, 88, 101, 170, 181, 216–218, 229, 241, 242, 253, 272
Meltzoff 349, 357, 363, 364, 369, 375, 376
Ménard 351, 376
Merkle 12, 20, 296, 297, 306, 309, 314, 315
Meuleau 171, 181
Meyer 168–170, 184
Miège 327, 335, 337, 341, 342, 344
Mihaylova 203, 210
Miller 53, 77
Mills 371, 376
Minguez 72, 78
Minoux 219, 241
Miolo 349, 374
Mohammad-Djafari 12, 18
Mollard 245, 273
Mölzer 166, 168, 182
Montano 72, 78
Moore 128, 131, 166, 181, 297, 314
Morales 106, 131
Moravec 80, 101, 217, 241
Morel 53, 78
Morris 12, 17
Moseley 266, 273
Moses 13, 15
Moutardier 105, 132
Muller 128, 130
Murphy 13, 18, 68, 78, 138–140, 158, 159, 163, 171, 183
Naïm 136, 140, 157, 158
Nadel 127, 132
Nagatani 105, 106, 120, 130
Naji 337–339, 344
Navarro 113, 130
Neal 13, 18, 223, 224, 241
Needleman 114, 132
Nefian 13, 18
Nemzov 105, 132
Neudorfer 12, 19
Newlands 306, 312, 314
Newman 105, 130
Norman 317, 344
Norvig 140, 159
Nourbakhsh 106, 113, 120, 121, 131, 133, 174, 184
O’Connell 317, 345
O’Keefe 127–129, 131, 132
Ohta 113, 131
Oie 13, 17
Olsson 353, 375
Oriolguet 352, 361, 372, 376
Overmars 171, 182, 183
Owen 105, 132, 217, 241
Paige 306, 309, 311, 315
Panerai 318, 329, 345
Pardo-Castellote 13, 19
Parr 68, 78, 195, 210
Patton 12, 14
Pavlovic 138, 158
Pearl 13, 15, 18, 140, 158
Peel 44, 50
Authors

Perrier 348, 350, 355–357, 361, 373, 376
Perzanowski 138, 158
Peterka 297, 314
Peucker 108, 131
Pfeffer 13, 17
Pfeifer 166, 168, 182
Pineau 171, 183
Piquemal 352, 376
Pissard-Gibolet 13, 15
Pizlo 12, 18
Poggio 12, 18
Pols 353, 376
Poole 13, 20
Pouget 217, 242
Rabbitt 298, 303, 315
Rabiner 136, 158, 162, 183, 286, 289
Ramel 139, 140, 152, 158, 159
Raphan 296, 297, 300, 303, 305, 306, 309, 314, 315
Recasens 354, 355, 376
Redish 127, 129, 132, 166, 183, 184
Rekleitis 106, 131
Richards 12, 17
Ritter 138, 158
Rives 168, 184
Robert 12, 18
Robert-Ribes 348, 376
Robinson 7, 18, 23, 50
Rogers 317, 318, 329, 344
Rootes 354, 375
Rosenberg 165, 183
Rosenblatt 13, 18
Röthling 138, 158
Roumeliotis 13, 19
Roweis 13, 18, 163, 183
Roy 165, 183
Rubin 138–140, 159, 286, 289
Ruiz 13, 19
Russel 140, 159
Rychert 12, 15, 19
Soyer 203, 210
Spelke 166, 168, 184
Spiegelhalter 13, 17
Stackman 168, 183
Stanton 317, 345
Steels 372, 377
Steil 138, 158
Steiner 165, 181
Sufit 353, 374
Sutherland 128, 130
Suzuki 305, 306, 313
Svestka 171, 182, 183
Tanila 127, 130
Tapus 106, 108, 116, 119, 122, 131, 132, 139, 140, 158, 159
Tarantola 12, 19
Tardos 105, 130
Taube 168, 183
Tay 82, 88, 101, 217, 242
Taylor 13, 18, 216, 217, 242
Tejada 278, 289
Telford 306, 315
Thrun 13, 14, 16, 19, 72, 77, 80, 101, 105, 106, 120, 132, 162, 163, 165, 171, 181, 183, 190, 195, 203, 208–210, 222, 241
Todd 317, 344
Tolman 103, 132, 168, 184
Torralba 138–140, 158, 159
Tourenq 13, 18
Touretzky 166, 183, 184
Trullier 168–170, 184
Turro 13, 15
Ullman 317, 345
Vallée 351, 355, 371–373, 376
Van Boxtel 329, 330, 335–337, 341, 345
Van der Kooij 13, 19
Van Der Stelt 349, 374
Veloso 278, 289
Verdot 13, 18
Vicente 248, 273
Victorino 168, 184
Vijayakumar 203, 210
Vilaín 348, 349, 373, 377
Von Helmholtz 317, 345
Wachsmuth 138, 158
Wallach 317, 345
Wang 13, 19, 166, 168, 184
Watt 12, 16
Wehner 166, 168, 182
Weiss 12, 19
Welch 83, 101, 136, 159
Wexler 318, 327, 329, 330, 333–337, 341, 342, 344, 345
Weymouth 105, 131
Wiener 168–170, 184
Williams 12, 19
Williamson 348, 374
Wioland 350, 373
Wolpert 12, 17
Wood 361, 377
Woodcock 277, 289
Wu 352, 377
Wunsch 114, 132
Yguel 82, 88, 101, 217, 242
Young 335, 344
Yuille 12, 17, 20
Zerling 350, 373
Zhao 12, 15
Zhang 13, 20, 217, 242
Zupan 12, 20, 297, 314, 315