Learning Models for Robot Navigation

Hagit Shatkay
Ph.D. Dissertation

Department of Computer Science
Brown University
Providence, Rhode Island 02912

CS-98-11
December 1998

Learning Models for Robot Navigation

by

Hagit Shatkay

B. Sc., The Hebrew University of Jerusalem, 1989

M. Sc., The Hebrew University of Jerusalem, 1992

A dissertation submitted in partial fulfillment of the
requirements for the Degree of Doctor of Philosophy
in the Department of Computer Science at Brown University

Providence, Rhode Island

May 1999

Vita

Name	Hagit Shatkay
Born	January 28, 1965 in Petah Tikva, Israel
Education	*Brown University*, Providence, RI Ph.D. in Computer Science, May 1999.
	The Hebrew University, Jerusalem, Israel M.Sc. in Computer Science, Cum Laude, May 1992.
	The Hebrew University, Jerusalem, Israel B.Sc. in Computer Science, Cum Laude, May 1989.
Honors	*Brown University*, Graduate Research Fellowship, 1997.
	The Hebrew University, Dean of the Faculty of Mathematics and Sciences, List of Academic Excellence, 1988.
	The Hebrew University, Dean of the Faculty of Mathematics and Sciences, Academic Excellence Award, 1986.
Teaching Experience	*The Israeli Open University*, Course Instructor, 1988.
	The Israeli Open University, Course Director, 1991.
Military Service	Lieutenant in the Israeli Defense Forces, 1983-1985.

Abstract

Hidden Markov models (HMMs) and partially observable Markov decision processes (POMDPs) provide a useful tool for modeling dynamical systems. They are particularly useful for representing environments such as road networks and office buildings, which are typical for robot navigation and planning. The work presented here describes a formal framework for incorporating readily available odometric information into both the models and the algorithm that learns them. By taking advantage of such information, learning HMMs/POMDPs can be made better and require fewer iterations, while being robust in the face of data reduction. That is, the performance of our algorithm does not significantly deteriorate as the training sequences provided to it become significantly shorter. Formal proofs for the convergence of the algorithm to a local maximum of the likelihood function are provided. Experimental results, obtained from both simulated and real robot data, demonstrate the effectiveness of the approach.

To my parents, Sarah (Huber) and Adam Shatkay, in memoriam

Acknowledgments

Getting a PhD is likely to be the end of my many years as a student. Throughout these years I was fortunate to be surrounded by people whose wisdom and kindness helped make this time so pleasant and interesting. Unfortunately, I can not possibly list them all here, but I hope to thank at least those who have most prominently affected the course I have taken so far.

Several teachers during elementary and high school influenced my choice of the scientific and academic direction. In particular, I thank Nira Efroni, Mr. Snook from the Lincoln primary school in New Zealand, Pearl Friedman, Pessia Michael, Zvia Solomon and Tamar Bachi.

Since starting my academic studies, both in the Hebrew University of Jerusalem and at Brown, I had the privilege to learn from and work with some of the wisest and most extraordinary people in the computer science world.

From the Hebrew University, I am especially grateful to Mordechai Tzipin, Saharon Shelah, Michael Rabin, Jeff Rosenschein and Mori Rimon. The sound basis they have provided have served me throughout my PhD work. Catriel Be'eri was my M.Sc. advisor, and introduced me to research in Computer Science. I am indebted to him for his non-compromising demands for rigor and accuracy, and hope that some of his influence is evident in this thesis.

At Brown, my deepest gratitude is extended to my advisor Leslie Pack Kaelbling. Even prior to my becoming her student, she has readily shared her deep knowledge and unique insight, and provided guidance and encouragement. Her enthusiasm, thoroughness and sheer enjoyment of the scientific work will remain with me, hopefully affecting the way I work and think.

Tom Dean's kindness, support and good advice during the early stages of my work, and his thoughtful comments as a thesis committee member were all extremely helpful. I am also most grateful to Sebastian Thrun from Carnegie Mellon University for being on my thesis committee, and for his insightful comments on my thesis, enlightening discussions, and contagious enthusiasm.

John Hughes deserves many thanks for his help throughout the years, both with graphics-oriented issues and with hard-core mathematics. I also thank Stanley Zdonik for being so up-beat and encouraging during my first years in Brown, and for complementing my theoretical background with his “where's the beef” approach. Eugene Charniak, Tom Doeppner, David Laidlaw, Philip Klein, Franco Preparata, Roberto Tamassia and Eli Upfal have all been willing to answer my questions and lend machine cycles when needed.

From the amazing technical and administrative staff, I especially thank Dorinda Multon and Max Salvas for putting up with more than their fair share of urgent and special requests, John Bazik, Kathy Kirman and Soren Spies for their helpfulness, and Susan (Platt) Hansen, Jennet Kirschenbaum, Dawn Nicholaus and Fran Palazzo for being so tolerant and accommodating to all sorts of last-minute needs. I also thank Mary Andrade, Lori Agresti, Suzy Howe and Mary Killilea for their friendliness and assistance. Trina Avery willingly answered numerous grammatical questions, and together with Leslie and Susan introduced me to the Providence Singers and the Messiah singing. This is one of the most beautiful memories I have of my time in Brown, and I thank all three of them for this.

A wonderful group of fellow graduate students has made my years in Brown so pleasurable. Lunches with the "DEC women" — Beth Phalen, Nisha Thatte-Potter, Maria Loughlin and Shamsi Moussavi — and our ongoing reunions have all been joyful events. My first officemate, Tony Cassandra, and his wife Ann Marie became good family friends. Tony has also been a partner for many valuable technical discussions. My current officemates, Bill Smart, who sustains Ramona the robot, and Kee Eung Kim, have carried on this tradition both personally and professionally. Vaso Chatzi, has been a treasured friend all these years, as well as a valuable reference source on many issues — from departmental news to geometry and algebra. This work would have literally looked very different without the relentless efforts of Dimitris Michailidis as a Tex Merc. Manos Renieris, who just replaced Dimitris in this capacity, has already performed several miracles as well. Friendship aside, discussions with Luis Ortiz have greatly contributed to my understanding of the EM algorithm. I'm sure the staff in "Montana" is going to miss our debates. Other people whose company I enjoyed both socially and professionally are Zori Atanassova, Sharon Caraballo, Jak Kirman, Jim Kurien, Sonia Leach, Michael Littman, Andrew McKeith, Laurent Michel, Marian Nodine, Bharathi Subramanian, Galina Shubina, Costas Busch, José Castaños and Shieu-Hong Lin, as well as Nicolas Meuleau and Milos Hauskrecht.

My family and friends in Israel have accompanied me through the years. My sister Michal has been an invaluable source of encouragement and advice, while my brother-in-law, Ira, has readily served as one of the better thesauri one could hope for. Shula, my aunt, went out of her way to help, and without her assistance after baby Ruth was born, it would have been very hard for me to resume my grad-school routine. Noga Amirav, David Michaeli, Gadi Solotorevski and Avinoam Kalma have kept our friendship going despite the distance. I am thankful to them, as well as to the rest of our family and friends in New England and in Israel.

All this said, the one person to whom I am most grateful and obliged is my husband, Yaron Reshef. I cannot thank him enough for his love, perseverance, optimism, excellent cooking, and the many hours he has registered as a single parent throughout these years. Finally, my son, Eadoh, and daughter, Ruth, have been the most wonderful children in the world. Their cheerful disposition helped put every experimental disaster and reluctant proof in perspective, and make the combination of motherhood and studentship so gratifying.

Contents

⋆ Parts of this thesis have been previously published in the International Joint Conference on Artificial Intelligence, 1997, and in the International Conference on Machine Learning, 1998, by Hagit Shatkay and Leslie Pack Kaelbling.

List of Tables

List of Figures

Chapter 1

Introduction

Dynamical systems provide a formal mathematical framework for describing many physical phenomena. Possible states of a physical system are represented as a set of vertices or *nodes*, and the dynamical aspect of states changing over time, as arcs or *transitions*. Since physical phenomena are seldom either fully observable or completely predictable, it is also desirable for dynamical systems to model the inherent uncertainty in observations and transitions. The work presented here is concerned with acquiring a particular family of models for dynamical systems, namely, Hidden Markov models.

1.1 HMMs and POMDP Models

Hidden Markov models (HMMs) represent a variety of *nondeterministic* dynamical systems as abstract probabilistic state-transition systems with discrete states and observations. The states of the dynamical system are naturally mapped to the states of the model. The observable aspects of each state in the dynamical system, which are often noisy and imprecise, are mapped to probability distributions or density functions over observations; each state in the model has associated with it a distribution, or a probability density function, over possible observations. The uncertain dynamics of the modeled system is represented through probabilistic transitions between the model's states; each state is assigned a probability distribution over the possible next states.

Such models are adequate for representing systems in which external entities exercise *no control* over the dynamics of the system, and the stochastic behavior is completely specified by the states, transitions and probabilities. They are widely used in a variety of areas such as natural language understanding [Cha93], speech recognition [Rab89, RJ93], handwritten text analysis [CKZ94, BG95], and protein and DNA representation [Chu89, BCH$^+$93, KBM$^+$94].

Hidden Markov models can be extended to model *decision* processes in which control is exercised, by introducing *actions* into the model. The extended models are known as *partially observable Markov decision process* (POMDP) models. Like the basic HMM, a POMDP model has a set of states corresponding to the states of the modeled system. In addition, each *action* has associated with it a set of transition probability distributions – one distribution per state. The distribution models the probabilistic transition resulting from executing the action in the state. Similarly, each action has a set of observation probability distributions, one distribution per state, modeling the probabilistic observation which can be perceived upon arrival at the state after executing the action.

POMDP models are useful for modeling processes in which the outcome is uncertain and the state is not fully observable. Such processes arise in almost all aspects of life, from financial investments to medical decision making. A variety of other applications is given in work by Littman [Lit96] and Cassandra [Cas98].

1.2 Models for Robot Navigation

POMDP models have proven particularly useful as a basis for robot navigation in buildings, providing a sound method for localization and planning [SK95, NPB95, CKK96]. Most other approaches to modeling environments for robot navigation [ME85, Asa91, LDWC91, TBF98] are concerned with obtaining a *geometrical* description of the environment, and are centered around finding *positions* and *locations* in it, trying to determine exactly where in the environment the robot is. In contrast, HMMs and POMDP models are centered around the concept of *state* rather than that of *location.*

A *state* typically corresponds to a significant landmark in the environment coupled with other important robot's attributes. Such attributes may include the robot's orientation, its arm position, or its voltage level. This more general concept, naturally captures robot behaviors and properties that do not necessarily involve a change in location, such as arm movement, picking or dropping an object, camera positioning etc. , thus providing a consistent framework for planning and acting in the environment. By being concerned with the *topology* induced by significant landmarks, rather than with the complete geometry of the space, the models also tend to be more compact and support efficient planning.

Much previous work on planning using POMDP models has required that the model be provided, through manual specification. This is a tedious process and it is often difficult to obtain correct probabilities. An ultimate goal is for an agent to be able to *learn* such models automatically, both for robustness and in order to cope with new and changing environments.

1.3 Learning the Model

From a theoretical-computational standpoint, HMMs and POMDP models, can be viewed as probabilistic finite automata (PFA) and input/output PFA, respectively. In the general case, the conjecture is that learning such models is hard, based on Abe and Warmuth's [AW92] non-approximability results with respect to probabilistic finite automata, as described in Section 2.1.2. Still, in practice, the Baum-Welch algorithm [Rab89] is frequently used to learn HMMs. Since POMDP models are a simple extension of HMMs, they can, theoretically, be learned with a simple extension to the Baum-Welch algorithm. However, in the general case, without strong prior constraint on the structure of the model, the Baum-Welch algorithm does not perform very well: it is slow to converge, requires a great deal of data, and is often stuck in local minima.

Typically, application domains in which HMM learning has proven successful provide some bias which assists in the learning process. For instance, due to the temporal nature of the speech process, it can be modeled using a specific family of HMMs, namely, *left-to-right* HMMs [Rab89]. In these models, transitions occur in one direction only, and there are no cycles other than ones caused by self-transitions. That is, the states can be indexed, such that the probability of transitions from state i to state j, where $j < i$, is 0. This constraint determines many of the model parameters, leaving fewer model parameters that actually need to be learned, thus making the learning problem significantly simpler. A similar constraint applies to handwritten text, as well as to biological structures such as proteins or DNA, due to their sequential nature. Such constraints do not usually hold in the navigation domain, since in most real environments one can move back and forth, repeatedly visiting the same states via various distinct routes.

Previous work, such as Koenig and Simmons' [KS96b] used prior knowledge of the environment to bias the learning algorithm towards the correct model. Using their approach, a human provides a correct but incomplete topological model of the environment, and the Baum-Welch algorithm is used to fill in the details. One of the central goals of the work presented here is to explore ways in which better models can be obtained, while using both less time and less data, without requiring a prior description of the learned environment.

1.4 A New Approach

The approach taken in this work is based on utilizing a different source of information which allows the Baum-Welch algorithm to learn good topological models without the use of human-provided initial model. We propose to use readily available weak *odometric information* to improve the results of the Baum-Welch algorithm.

Most robots are equipped with wheel encoders that enable an odometer to record the change in the robot's position as it moves through the environment. This data is typically very noisy and inaccurate. The floors in the environment are rarely smooth, the wheels of the robot are not always aligned and neither are the motors, a lot of the mechanics is imperfect, resulting in slippage and drift. All these effects accumulate, and if we were to mark the initial position of the robot, and try to estimate its current position based on a long sequence of odometric recordings, we would find that our estimate is typically incorrect. That is, the raw recorded odometric information is not an effective tool for determining the absolute location of the robot in the environment.

The idea underlying our approach is that this weak odometric information, despite its noise and inaccuracy, still provides geometrical cues that can help to distinguish between different states as well as to identify revisitation of the same state. Hence, such information enhances the ability to learn *topological* models. However, the use of geometrical information requires careful treatment of geometrical constraints and directional data.

We demonstrate how the existing models and algorithms can be extended in order to take advantage of the noisy odometric data and the geometrical constraints. The geometrical information is directly incorporated into the probabilistic topological framework, producing a significant improvement over the standard Baum-Welch algorithm, without the need for human-provided model. Although there are still a number of intriguing problems that need to be addressed, our experiments prove that this is a promising direction in model acquisition for robot navigation.

As a possible generalization to the problem of HMM acquisition, outside the scope of robotics, our approach demonstrates the merit of using domain-specific constraints to achieve high utilization of the data, and restrict the learning process, directing it towards acquiring better models. We believe that this approach can be put to use in other domains, such as medical decision making and biological modeling. In the medical domain, various conditions and symptoms exclude each other, and temporal constraints restrict the possible transitions in the patient's state. In the molecular biology domain, one can exploit 3-dimensional geometrical constraints over molecular structures, which are likely to be analogous to the constraints arising when modeling environments for robot navigation. We expect that by using these constraints, the space of appropriate models which may fit a data set can be reduced, and the model acquisition process can be made more accurate and efficient.

1.5 Thesis Outline

The rest of the thesis is organized as follows: Chapter 2 provides a survey of previous work in the area of learning maps and automata; Chapter 3 presents the formal framework for this work; Chapter 4 describes the basic algorithm we have developed for using odometric information in the context of the Baum-Welch algorithm; Chapter 5 discusses special issues in handling directional data within a probabilistic framework; Chapter 6 presents methods for choosing an initial model from which to start the algorithm, and introduces a new method we have developed for this purpose; Chapter 8 describes ways to overcome the problem of cumulative rotational errors, which is another facet of the problems caused by the presence of directional data and angular changes; In Chapter 10 we provide a way for enforcing complete geometrical consistency in the topological model throughout the learning process; Chapters 7, 9, and 11 present experimental results for each variant of our learning algorithm. The experiments demonstrate that our algorithm indeed converges to better models with fewer iterations than the standard Baum-Welch, and is robust in the face of data reduction. In Chapter 12 we summarize the results and conclude the work, as well as list several directions for future research.

Chapter 2

Approaches to Learning Maps and Models

The work presented in this document lies in the intersection between the theoretical area of learning computational models — in particular learning automata from data sequences — and the applied area of map acquisition for robot navigation. In the following we provide a survey of results from both of these areas. The reinforcement learning literature also addresses some aspects of learning models for Markov decision processes [Sut90, Thr92, Kae93]. The latter can be viewed as a special case of learning probabilistic automata with fully observable states, and we briefly review related work from this domain in Section 2.1.3.

2.1 Learning Automata from Data

Informally speaking, an automaton consists of a set of states, and a set of transitions which lead from one state to another. In the context of this work, the automaton states correspond to the states of the modeled environments, and the transitions, to the state changes due to actions performed in the environment. Each transition of the automaton is tagged by a symbol from an *input alphabet*, Σ, corresponding to the action or the *input* to the system, which caused the state transition. An example of an automaton with three states and input alphabet $\{a, b\}$ is shown in Figure 2.1.

Classical automata theory [HU79] distinguishes two types of special states; a single *initial* state and a set of *accepting* states. If a sequence of actions starts from an initial state and results in an accepting state, it is said that the automaton *accepts* the sequence. For

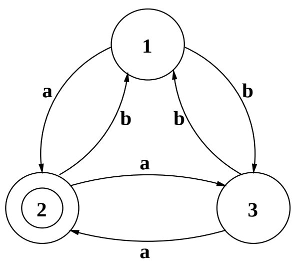

Figure 2.1:A 3-state automaton over the alphabet $\{a, b\}$.

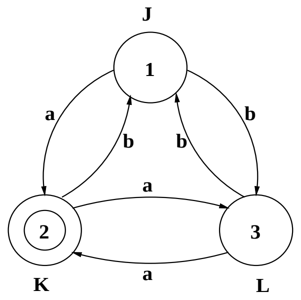

Figure 2.2:An input-output automaton; the input alphabet is $\{a, b\}$, the output alphabet is $\{J, K, L\}$.

instance, in Figure 2.1, state 2 is depicted as a double circle, denoting an accepting state. If state 1 is assigned to be the *initial* state, the sequences $\langle a\rangle$, $\langle b\,a\rangle$ and $\langle a\,b\,a\rangle$, are all *accepted* by the automaton, while the sequence $\langle a\,b\rangle$ is not.

The basic structure described above can be further extended to model the generation of output sequences [HU79]. This is done by defining an *output alphabet* Δ and assigning to each state a symbol in Δ that is emitted each time the state is reached. Such extended automata are called *input-output automata.* Figure 2.2 depicts a 3-state automaton over the input alphabet $\{a, b\}$ and the output alphabet $\{J, K, L\}$. For instance, if the input sequence is $\langle a\,b\,a\rangle$ and the initial state is 1, the generated output sequence is $\langle J\,K\,J\,K\rangle$.

There are various possible kinds of *uncertainty* about the environment as well as the interaction with it, which can be modeled through different types of automata. First, states in the environment can be either *fully observable* or *partially observable.* If the environment is fully observable, one always knows its exact state in the environment. When states are only partially observable or *hidden*, one does not know its state with certainty. In addition, the results of each action taken in the environment can be either *fully determined* or *uncertain.* At any given state (be it observable or hidden), the execution of a fully deterministic action is guaranteed to lead to a single next state. The execution of an action with uncertain results is not guaranteed to lead to a single next state and is modeled as a *stochastic transition function.* Given a pair consisting of the current state and action, the transition function assigns to each state a probability of being reached through the action, from the current state. Based on these distinctions, we can partition automata into four groups:

- Fully observable states, deterministic transitions
- Fully observable states, stochastic transitions
- Hidden states, deterministic transitions
- Hidden states, stochastic transitions

Automata with fully observable states can be viewed as input-output automata in which states are distinctly labeled, the output alphabet consists of state labels, and each state emits its own label when visited. Automata with *hidden* states do not emit their state labels, but might emit other output symbols (hence the term *partially observable*).

We can imagine an agent moving through an environment while recording its perceived observations and actions. The problem of *learning* an automaton, is informally described as the problem of constructing an automaton that *accepts* the recorded sequence of actions, and *emits* the recorded sequence of observations, if such observations exist. In a setting where the automaton does not have a distinct *accepting* state, the learning problem is similar, but merely requires that the learned automaton has a directed path through its states, corresponding to the recorded input (and/or output) sequence.

Acting in a fully observable and deterministic environment, corresponding to an automaton of the first kind, we can record the origin state in which we start the exploration, as well as each subsequently visited state. In terms of decision process models, this can be viewed as acting within the framework of *deterministic* Markov decision processes [Put94]. After visiting all the states (and executing all possible actions - if we do have a choice of action), we obtain a complete model of the environment. That is, we deterministically know how to get from each state to all the other reachable states. Hence, learning a model of such an environment is easy.

The second kind of automata corresponds a *stochastic* Markov decision process model [Put94]. Learning such a model based on a sequence of recorded visited states and executed actions, amounts to estimating transition probabilities under the executed actions. It is a fairly simple task, under the assumption that the sequence of recorded states is provided and we are not dealing with the problem of obtaining sufficient data for estimation purposes, and is discussed in Section 2.1.3.

Obtaining models of the third and the fourth kinds, correspond to the problems of learning a *deterministic* and a *probabilistic* finite automaton, respectively. These problems do not have simple solutions in the general case. The models and their respective learning problems are discussed in detail in Sections 2.1.1 and 2.1.2.

It is also possible to have a fully deterministic environment in which an agent with imperfect perception records its actions and observations. In this case the agent may record the wrong states, actions or observations resulting in a noisy sequence from which learning needs to be done. In this case, the learning procedure needs to take into account that with some probability each recorded item may be wrong. The model learned is deterministic, rather than stochastic, but it might contain errors with respect to the true model, due to the erroneous data from which it was learned. Some results under this scenario are also discussed in Section 2.1.1.

2.1.1 Deterministic Automata

A standard deterministic finite automaton consists of a finite set of states Q, a finite input alphabet Σ, a transition function $\delta : Q \times \Sigma \rightarrow Q$, and a set $F \in Q$ of accepting states. The basic problem of learning finite *deterministic* automata from given data can be roughly described as follows: *Given two sets of positive and negative example strings, S and T respectively, over alphabet Σ, and a fixed number of states k, construct a minimal deterministic finite automaton with no more than k states that accepts S and does not accept T.* This problem has been shown to be NP-*complete* [Gol78]. Pitt and Warmuth [PW89] have shown that even if we are not learning the *minimal* automaton of k states, but are willing to learn an automaton with a polynomial number of states $f(k)$ with the same language, the problem is still NP-*complete*.

Despite the hardness, positive results have been shown possible within various special settings. Angluin [Ang87] showed that if there is an oracle to answer membership queries (assuming a reset operator of the automaton to its initial state), and to provide counterexamples to conjectures about the automaton, there is a polynomial time learning algorithm from positive and negative examples. Rivest and Schapire [RS87b, RS87a] provide an effective method for learning permutation automata, using *distinguishing sequences* (called "tests") for disambiguating states. Their method is guaranteed to find an automaton that with high probability is the correct one. In later work, [RS89], the authors use *homing sequences* for the same purpose. They show that they can learn a correct permutation automaton in polynomial time assuming there is a "teacher" which provides counterexamples, while a *highly probable* automaton can be learned even without the assumption of a teacher.

All of the above work assumes deterministic, *noise-free* behavior of the learned automaton. As mentioned earlier, there are cases in which the training sequence from which the automaton is learned may be noisy. Basye, Dean and Kaelbling [BDK95] presented several algorithms that, with high probability, learn input-output *deterministic* automata when various forms of *noise* are present in the training data. They show that when the transitions (actions) are deterministic but output emissions (observations) are noisy, a polynomial time algorithm exists, that learns a correct deterministic model with high probability. The algorithm does not learn a distribution over the observations, but rather assumes that a likely observation exists for each state and this observation is the one learned. Thus the learned model is *completely deterministic* rather than probabilistic. Similar results hold when the transitions are noisy and the observations are deterministic. (Again, the automaton learned is a *deterministic* one and does not model the transitions as probabilistic). For the case where both transitions and observations are noisy, a polynomial time algorithm for learning a probably correct deterministic automaton is given under strong assumptions, which include unique labeling of states.

2.1.2 Probabilistic Automata

Probabilistic automata are ones in which a probability distribution governs the transitions between states on any given input. In addition, in the case of input-output automata, a probability distribution is defined over the output emissions as well. The basic learning problem in this context is to find an automaton that assigns the same distribution as the true one to data sequences, from training data S generated by the true automaton. Another form of a learning problem is that of finding a probabilistic automaton λ that assigns the maximum likelihood to the training data S, that is, an automaton that maximizes $\Pr(S|\lambda)$.

Abe and Warmuth [AW92] show that finding a probabilistic automaton with 2 states, even when small error with respect to the true model is allowed with some probability (the *Probably Approximately Correct* learning model), cannot be done in polynomial time with a polynomial number of examples, unless NP = RP. They also show the equivalence of the problem of learning an automaton in the PAC sense to that of approximating the maximum likelihood automaton. This means that approximating a solution to any of the two learning problems stated above, for a probabilistic automaton, is equivalently hard. From their work arises a broader conjecture, which has not yet been proven, that the general problem of learning probabilistic automata with any number of states, even under the PAC learning model, is hard. A similar broadly accepted conjecture stemming from the same work is that learning hidden Markov models (the kind of probabilistic automata formally introduced in Section 3.1) is hard even in the PAC sense.

Two ways of addressing this hardness are presented in the rest of this section. One uses restrictions on the class of probabilistic models learned, and the other learns an unrestricted hidden Markov model with good practical results but with no PAC guarantees on the quality of the result.

Restricting the Learning Problem: In their above mentioned paper, Abe and Warmuth suggest that an interesting open problem is to find subclasses of probabilistic automata that are both practically useful and polynomially PAC learnable.

Work by Ron et al. [RST94, RST95, RST98] pursues such an approach. The authors present two classes of probabilistic automata that are useful in the area of natural language understanding, in particular for cursive hand writing recognition, speech recognition and printed text analysis. One such class consists of *acyclic* probabilistic finite automata, and the other of *probabilistic finite suffix* automata. Both of these classes can be learned in polynomial time (in all the parameters) within the PAC framework.

Learning with Restricted Guarantees: Another approach, the one predominantly taken in this work, is to learn a model for the data from the complete unrestricted class of hidden Markov models. Only weak guarantees exist about the goodness of the model, but the learning procedure may be directed to obtain practically good results.

This approach is based on guessing an automaton (model), and using an iterative procedure to make the automaton fit the training data better. One algorithm commonly used for this purpose is the Baum-Welch algorithm [BE67, BS68, BPS+70], which is presented in detail by Rabiner [Rab89]. The iterative updates of the model are based on gathering sufficient statistics from the data given the current automaton, and the update procedure is guaranteed to converge to a model that locally maximizes the likelihood function $Pr(\text{data}|\text{model})$. Since the maximum is *local*, the model might not be close enough to the automaton by which the data was generated, and a challenging problem is to find ways to force the algorithm into converging to higher maxima, or at least to make it converge faster, facilitating multiple guesses of initial models, thus raising the probability of converging to higher maxima. Such an approach is the one taken in this work.

Throughout this work we assume that the *number of states* in the model we are learning is given as input. This is not a very strong assumption, since there exist methods for learning the number of states. A natural generalization of the algorithm presented here is to apply such methods to directly learn the number of states from the data. Obviously, without any bound on the number of states, one can designate a state for each data point in the input sequence, thus perfectly fitting the data. Such an approach is a trivial example of *overfitting*; the model indeed fits the data well but is not general enough for modeling other data obtained from the same modeled environment. Regularization methods are used in order to avoid overfitting, directing the learning process towards models that fit both the current training data as well as yet-unseen data. One such technique is *cross-validation* [Sto74, Sha93, ET93]. Its basic idea is to use only parts of the available data to learn models of varying number of states, while saving some of the data for testing purposes. Once several models are learned, the likelihood (or some other measure of goodness) that they assign to the part of the data not used for learning is compared. The number of states for the model that has the highest measure of goodness, is taken to be the correct number of states, and is fixed. A final model is then obtained by learning from the complete data under the fixed number of states. Other regularization methods such as the *minimum description length* principal for deciding on the number of states and other model parameters, are discussed in Vapnik's book [Vap95]. Another similar criterion suggested by Akaike is described in a book by Sakamoto et al. [SIK86]. In Section 6.2 we suggest another possible heuristic for estimating the number of states as part of an initialization algorithm.

2.1.3 Models for Markov Decision Processes

Much of the work on reinforcement learning [Kae93, Sut90, Thr92, BBS95, MB98] is concerned with acting optimally within the context of *fully observable* Markov decision processes. The Markov model consists of states and actions that transition an agent from one state to the other, where every such transition has associated with it a reward. The goal of the agent is to optimize its reward. The transitions between states are usually stochastic, and the agent does not always know either the probability distribution governing the transition or the reward associated with each state-action pair. In such cases, where the parameters are unknown to the agent, it tries to obtain knowledge about them via *exploration*. The main idea behind exploration is that by taking actions at each state, the agent obtains counts of the number of times it ended up in every state. It uses the counts to calculate sufficient statistics and estimate the transition probabilities which it does not know a priori. Given a sequence of states recorded during exploration, learning the model is a straightforward statistical estimation problem [Bil59]. The more involved issue is that of deciding on strategies to explore the environment in order to obtain the data [Mar67].

This form of model learning is different from the problem we are addressing, since in the HMM and POMDP case the state itself is *hidden* and one can not directly obtain transition counts between states and calculate statistics. Another aspect of the learning in a partially observable environment is that of learning the *observation distribution* associated with each state, as described in Chapter 3. This aspect does not exist in the fully observable case.

2.2 Learning Maps and Models for Robot Navigation

The other area which closely relates to the work presented here is that of modeling environments for robot navigation. A distinction is usually made between two principal kinds of maps: *geometric* and *topological*. Geometric maps describe the environment as a collection of *objects* or *occupied positions* in space, and the *geometric* relationships among them. The *topological* framework is less concerned with the geometry, and models the world as a collection of *states* and their *connectivity*, that is, which states are reachable from each of the other states and what actions lead from one state to the next.

We draw an additional distinction, between world-centric[1] *maps* that provide an "objective" description of the environment independent of the agent using the map, and robot-centric *models* which capture the *interaction* of a particular "subjective" agent with the environment. When learning a *map*, the learning agent needs to take into account its own noisy sensors and actuators and try to obtain an objectively correct map that other agents could use as well. Similarly, other agents using the map need to compensate for their own limitations in order to assess their position according to the map. When learning a *model* that captures *interaction* the agent acquiring the model is the one who is also using it. Hence, the noisy sensors and actuators specific to the agent are reflected in the model. A different model is likely to be needed by different agents. Most of the related work described below, especially within the geometrical framework, is centered around learning objective maps of the world rather than agent-specific models. We shall point out in this survey the work that is concerned with the latter kind of models.

Our work focuses on acquiring *purely topological models*, and is less concerned with learning geometrical relationships between locations or objects, or objective maps, although geometrical relationships do serve as an aid in our acquisition process. The concept of a *state* used in this topological framework is more general than the concept of a *geometrical location*, since a state can include information such as the battery level, the arm position etc. Such information, which is of great importance for planning, is non-geometrical in nature and therefore can not be readily captured in a purely geometrical framework. The following provide a survey of both work done within the geometrical framework and within the topological framework as well as combinations of the two approaches.

[1]I thank Sebastian Thrun for the terminology.

2.2.1 Geometric Maps

Geometric maps provide a description of the environment in terms of the objects placed in it and their positions. For example, *grid-based* maps are an instance of the geometric approach. In a grid-based map, the environment is modeled as a grid (an array), where each position in the grid can be either vacant or occupied by some object (binary values placed in the array). This approach can be further refined to reflect uncertainty about the world, by having grid cells contain occupancy probabilities rather than just binary values. A lot of work has been done on learning such grid-based maps for robot navigation, through the use of sonar readings and their interpretation, by Movarec and Elfes and others [ME85, Mor88, Elf89, Asa91].

An underlying assumption when learning such maps is that the robot can tell where it is on the grid when it obtains a sonar reading indicating an object, and therefore can place the object correctly on the grid. A similar localization assumption underlies other geometric mapping techniques [LDWC91, SSC91, TGF$^+$98], even when an explicit grid is not part of the model. This assumption can be hard to satisfy. Leonard and Cox [LDWC91] and Smith et al. [SSC91] address this issue through the use of geometrical beacons to estimate the location of the robot. A probability distribution is used to model the robot's possible current location, based on observations collected up to the current point.

Recent work by Thrun et al. [TBF98], uses a similar probabilistic approach for obtaining grid-based maps. This work is refined [TGF$^+$98] to first learn the location of significant landmarks in the environment and then fill in the details of the complete geometrical grid, based on laser range scans. The latter work extends the approach of Smith et al. , by using observations obtained both *before* and *after* a location has been visited, in order to derive a probability distribution over possible locations. To achieve this, the authors use a *forward-backward* procedure similar to the one used in the Baum-Welch algorithm [Rab89], (see Chapter 4 of this work), in order to determine possible locations from observed data. The approach resembles ours both in the use of the *forward-backward* estimation procedure, and in its probabilistic basis, aiming at obtaining a maximum likelihood map of the environment. It still significantly differs from ours both in its initial assumptions and in its final results. The data assumed to be *provided* to the learner includes both the motion model and the perceptual model of the robot. These consist of transition and observation probabilities within the grid. Both of these components are *learnt* by our algorithm, although not in a grid context but in a topological, coarser-grained, framework. The end result of their algorithm is a probabilistic grid-based map, while ours is a probabilistic topological model.

In addition to being concerned only with locations, rather than with the richer notion of *state*, a fundamental drawback of geometrical maps is their fine granularity and high accuracy. Geometrical maps, particularly grid-based ones, tend to give an accurate and detailed picture of the environment. In cases where it is necessary for a robot to know its exact location in terms of metric coordinates, metric maps are indeed the best choice. However, many planning tasks do not require such fine granularity or accurate measures, and are better facilitated through a more abstract representation of the world. For example, if a robot needs to deliver a bagel from office a to office b, all it needs to have is a map depicting the relative location of a with respect to b, the passageways between the two offices, and perhaps a few other landmarks to help it orient itself if it gets lost. If it has a reasonably well-operating low-level obstacle avoidance mechanism to help it bypass flower pots and chairs that it might encounter on its way, such objects do not need to be part of the environment map. Just as a driver traveling between cities needs to know neither its longitude and latitude coordinates on the globe, nor the location of the specific houses along the way, the robot does not need to know its exact location within the building nor the exact location of various items in the environment, in order to get from one point to another. Hence, the effort of obtaining such detailed maps is not usually justified. In addition the maps can be very large, which makes planning — even though planning is polynomial in the size of the map — be inefficient.

2.2.2 Topological Maps and Models

An alternative to the detailed geometric maps are the more abstract topological maps. Such maps specify the *topology* of important landmarks and situations (*states*), and routes or transitions (*arcs*) between them. They are less concerned with the physical location of landmarks, and more with topological relationships between situations. Typically, they are less complex and support much more efficient planning than metric maps. Topological maps are built on lower-level abstractions that allow the robot to move along arcs (perhaps by wall- or road-following), to recognize properties of locations, and to distinguish significant locations as *states*; they are flexible in allowing a more general notion of state, possibly including information about the non-geometrical aspects of the robot's situation.

There are two typical strategies for deriving topological maps: one is to learn the topological map directly; the other is to first learn a geometric map, then to derive a topological model from it through some process of analysis.

A nice example of the second approach is provided by Thrun and Bücken [TB96a, TB96b, Thr99], who use occupancy-grid techniques to build the initial map. This strategy is appropriate when the primary cues for decomposition and abstraction of the map are geometric. However, in many cases, the nodes of a topological map are defined in terms of other sensory data (e.g. labels on a door or whether or not the robot is holding a bagel). Learning a geometric map first also relies on the odometric abilities of a robot; if they are weak and the space is large, it is very difficult to derive a consistent map.

In contrast, our work concentrates on learning a topological model *directly*, assuming that abstraction of the robot's perception and action abilities has already been done. Such abstractions were manually encoded into the lower level of our robot navigational software, as described in Chapter 7. Work by Pierce and Kuipers [PK97] discusses an automatic method for extracting abstract states and features from raw perceptual information.

Kuipers and Byun [KB91] provide a strategy for learning *deterministic* topological maps. It works well in domains in which most of the noise in the robot's perception and action is abstracted away, learning from single visits to nodes and traversals of arcs. An underlying assumption for this strategy is that the current state can be reliably identified based on local information, or based on distance traversed from the previous well-identified state. It is unable to handle situations in which long sequences of actions and observations are necessary to disambiguate the robot's state.

Engelson and McDermott [EM92] learn "diktiometric" maps (topological maps with metric relations between nodes) from experience. The uncertainty model they use is interval-based rather than probabilistic, and the learned representation is deterministic. *Ad hoc* routines handle problems resulting from failures of the uncertainty representation.

We prefer to learn a combined *model* of the world and the robot's interaction with the world; this allows robust planning that takes into account likelihood of error in sensing and action. The work most closely related to ours is by Koenig and Simmons [KS96b, KS96a], who learn POMDP models (stochastic topological models) of a robot hallway environment. They also recognize the difficulty of learning a good model without initial information; they solve the problem by using a human-provided topological map, together with further constraints on the structure of the model. A modified version of the Baum-Welch algorithm learns the parameters of the model. They also developed an incremental version of Baum-Welch that can be used on-line. Their models contain very weak metric information, representing hallways as chains of one-meter segments and allowing the learning algorithm to select the most probable chain length. This method is effective, but results in large models with size proportional to the hallways length, and strongly depends on the provision of a good initial model.

The rest of the work describes our approach to learning topological models. We show that by using weak odometric information directly, we can avoid the use of human-provided *a priori* models and still learn stochastic maps efficiently and effectively.

Chapter 3

Models and Assumptions

This chapter describes the basics of the formal framework for our work. It starts by introducing the classic hidden Markov model. The model is then extended to accommodate noisy odometric information in its simplest form, ignoring information about the robot's heading and orientation. In chapters 5 and 8, the model is further extended and refined to accommodate heading information and address the problems that arise as a result.

We concentrate here on describing models and algorithms for learning HMMs, rather than POMDPs. The extension to complete POMDPs is through learning an HMM for each of the possible actions, and is straightforward although notationally more cumbersome. We briefly discuss it in Section 3.3.

3.1 HMMs – The Basics

A hidden Markov model consists of states, transitions, observations and probabilistic behavior. We provide here a more formal definition of this basic model. In the next section we elaborate the definition to account for odometric information.

A hidden Markov model is a tuple $\lambda = \langle S, O, A, B, \pi \rangle$, where

- $S = \{s_0, \ldots, s_{N-1}\}$ is a finite set of N states;
- $O = \{o_1, \ldots, o_M\}$ is a finite set of M possible observation values;
- A is a stochastic transition matrix, with $A_{i,j} = Pr(q_{t+1} = s_j | q_t = s_i)$; $0 \le i, j \le N-1$; q_t is the state at time t; for every state s_i, $\sum_{j=0}^{N-1} A_{i,j} = 1$.

$A_{i,j}$ holds the transition probability from state s_i to state s_j.

- B is a stochastic observation matrix, with $B_{j,k} = Pr(v_t = o_k | q_t = s_j)$; $0 \leq j \leq N-1$, $1 \leq k \leq M$; v_t is the observation recorded at time t; for every state s_j, $\sum_{k=1}^{M} B_{j,k} = 1$.

  $B_{j,k}$ holds the probability of observing o_k while being at state s_j.

- π is a stochastic initial distribution vector, with $\pi_i = Pr(q_0 = s_i)$; $0 \leq i \leq N-1$; $\sum_{i=0}^{N-1} \pi_i = 1$. π_i holds the probability of being in state s_i at time 0, when starting to record the observations.

This model corresponds to a world in which the actual state of matters at any given time t, $q_t \in S$, is hidden and not directly observable, but some observation, $v_t \in O$, is detected and recorded at the state when it is visited at time t. An agent moves from one hidden state to the next according to the probability distribution encoded in matrix A. The observed information in each state is governed by the probability matrix B.

Given a stochastic system with an unknown model, one can gather sequences of observations in the system. By calculating sufficient statistics from the observed data, *estimates* for the states and the observations of the system are obtained. Using these estimates, one may be able to reconstruct a plausible model of the system, as demonstrated by the following simple example.

Example 3.1 *Consider a system consisting of a single biased coin that is being tossed. It can be viewed as a system with a single state, in which one can observe, either a* head, *H, or a* tail, *T, with some* unknown *probability.*

A sequence of observations can be recorded by tossing the coin several times. For instance, $H\,T\,T\,T\,H\,T\,T$, is such a sequence. By counting the number of times H was observed (2), and the number of times T was observed (5), we obtain the estimate $\frac{2}{7}$ for the probability of observing a head, and the estimate $\frac{5}{7}$ for the probability of observing a tail. These probabilities constitute a plausible model of the tossed coin.

The learning problem for HMMs can be roughly stated as follows: *Given a sequence of observations gathered from a stochastic system, reconstruct a plausible hidden Markov model of the system.* A more accurate measure of "plausibility" will be given in Section 4.1.

3.2 Adding Odometry to Hidden Markov Models

The world is composed of a finite set of states. The states do not necessarily correspond directly to locations of the robot; they may include other state information, such as orientation or battery level. The dynamics of the world are described by state-transition distributions that specify the probability of making transitions from one state to the next. There is a finite set of observations that can be made in each state; the frequency of such observations is described by a probability distribution and depends only on the current state. In our model, observations are *multi-dimensional*; an observation is a vector of values, each chosen from a finite domain. It is assumed that these observation values are conditionally independent, given the state.

In addition to the set of possible observations, each state is assumed to be associated with a position in a metric space. Whenever a state transition is made, the robot records an *odometry vector*, which estimates the position of the current state relative to the previous state. For the time being we assume that the odometry vector consists of readings of x and y coordinates in a global coordinate system, and that these readings are corrupted with independent normal noise (extension to dependent noise is possible, and requires consideration of the complete covariance matrix). We extend the odometry vector to include information about the heading of the robot, and relax the global coordinate system assumption in Chapters 5 and 8, respectively.

There are two important assumptions underlying our treatment of odometric relations between states: First, that there is an inherent "true" odometric relation between the position of every two states in the world; Second, that when the robot moves from one state to the next, there is a normal, 0-mean noise around the correct expected odometric reading along each odometric dimension. This noise reflects two kinds of odometric error sources:

– The lack of precision in the discretization of the real world into states (e.g. there is a rather large area in which the robot can stand which can be regarded as "the doorway of the AI lab").

– The lack of precision of the odometric measures recorded by the robot, due to slippage, friction, disalignment of the wheels, imprecision of the measuring instruments, etc.

To formally introduce odometric information into the hidden Markov model framework, we define an *augmented hidden Markov model* as a tuple $\lambda = \langle S, O, A, B, R, \pi \rangle$, where

- $S = \{s_0, \ldots, s_{N-1}\}$ is a finite set of N states;

- $O = \prod_{i=1}^{l} O_i$ is a finite set of observation vectors of length l; the ith element of an observation vector is chosen from the finite set O_i;

- A is a stochastic transition matrix, with $A_{i,j} = Pr(q_{t+1} = s_j | q_t = s_i)$; $0 \leq i, j \leq N-1$; q_t is the state at time t;

  $A_{i,j}$ holds the transition probability from state s_i to state s_j.

- B is an array of l stochastic observation matrices, with $B_{i,j,o} = Pr(V_t[i] = o | q_t = s_j)$; $1 \leq i \leq l$, $0 \leq j \leq N-1$, $o \in O_j$; V_t is the observation vector at time t; $V_t[i]$ is its i^{th} component.

  $B_{i,j,k}$ holds the probability of observing o_k along the i^{th} component of the observation vector, while being at state s_j.

- R is a relation matrix, specifying for each pair of states, s_i and s_j, the mean and variance of the D-dimensional[1] odometric relation between them; $\mu(R_{i,j}[m])$ is the mean of the m^{th} component of the relation between s_i and s_j and $\sigma^2(R_{i,j}[m])$, the variance; furthermore, R is *geometrically consistent*: for each component m, the relation $\mu^m(a,b) \overset{\text{def}}{=} \mu(R_{a,b}[m])$ must be a *directed metric*, satisfying the following properties for all states a, b, and c:

  ◇ $\mu^m(a,a) = 0$;

  ◇ $\mu^m(a,b) = -\mu^m(b,a)$ *(anti-symmetry)*; and

  ◇ $\mu^m(a,c) = \mu^m(a,b) + \mu^m(b,c)$ *(additivity)* .

  This representation of odometric relations reflects the two assumptions, previously stated, regarding the nature of the odometric information. The "true" odometric relation between the position of every two states is represented as the mean. The noise around the correct expected odometric relation, accounting for both the lack of precision in the real-world discretization and the inaccuracy in measurement, is represented through the variance.

- π is a stochastic initial probability vector describing the distribution of the initial state; for simplicity it is assumed here to be of the form $\langle 0, \ldots, 0, 1, 0, \ldots, 0 \rangle$, implying that there is one designated initial state, s_i, in which the robot is always started.

[1]For the time being we consider D to be 2, corresponding to (x, y) readings.

This model extends the standard hidden Markov model, as presented in Section 3.1, in two ways:

- It allows for observations to be factored into independent components (given the state), and represented as vectors. Factoring the observations into components and assuming conditional independence between them allows for the calculation of the probability of an observation vector from the probability of its components. It therefore results in fewer probabilistic parameters in the learnt model than if we were to view each *observation vector* as a single "atomic" observation.

- It introduces the *odometric relation* matrix R and constraints over its components. The use of R and the constraints over it have proven useful for learning the other model parameters, as demonstrated in Chapters 7, 9 and 11.

3.3 Extending POMDP Models

We briefly review the definition of partially observable Markov decision process models (POMDP models), and describe their adaptation for supporting odometric information. A more detailed description of standard POMDPs can be found in work done by Cassandra, Littman and Kaelbling [CKL94, CKK96, Cas98].

Traditionally, a POMDP model consists of:

- $S = \{s_0, \ldots, s_{N-1}\}$ is a finite set of N states;

- $O = \{o_1, \ldots, o_M\}$ is a finite set of M possible observation values;

- $\mathfrak{a} = \{a_1, \ldots, a_K\}$ is a finite set of K possible actions;

- $\{A^1, \ldots, A^K\}$ are stochastic transition matrices, one for each possible action; $A^l_{i,j} = Pr(q_{t+1} = s_j | q_t = s_i, c_t = a_l)$; $0 \leq i, j \leq N-1$; $1 \leq l \leq K$; q_t is the state at time t; c_t is the action taken at time t; for every state s_i and action a_l, $\sum_{j=0}^{N-1} A^l_{i,j} = 1$.

- $\{B^1, \ldots, B^K\}$ are stochastic observation matrices, one for each possible action; $B^l_{j,k} = Pr(v_t = o_k | q_t = s_j, c_{t-1} = a_l)$; $0 \leq j \leq N-1$, $1 \leq k \leq M$, $1 \leq l \leq K$; v_t is the observation recorded at time t; c_{t-1} is the action taken at time $t-1$, which caused the transition from the previous state to state s_j; for every state s_j and action a_l, $\sum_{k=1}^{M} B^l_{j,k} = 1$.

- π is a stochastic initial probability vector describing the distribution of the initial state of the model; $\pi_i = Pr(q_0 = s_i)$; $\sum_{i=0}^{N-1} \pi_i = 1$.

The above is a straightforward extension of the basic HMM described in Section 3.1 to a decision process model that includes actions[2]. This definition implies that a POMDP model can be viewed as a collection of K HMMs, where K is the number of actions. As such, it can be learned through a simple extension to any algorithm aimed at acquiring HMMs.

We extend the definition to accommodate multi-dimensional observation vectors as follows: $O = \prod_{i=1}^{l} O_i$ is a finite set of observation vectors of length l; the ith element of an observation vector is chosen from the finite set O_i.

As in the case of HMMs, we introduce the odometric relation matrix. However, there is still only *one matrix R* that is common for the whole POMDP, as opposed to one matrix per action. The reason is that usually a single action type does not allow us to gather enough information about the odometric relation among a group of neighboring states, in order to deduce reliable mean and standard deviation. By considering all odometric transitions combined over all the executed actions we can obtain better estimates regarding the odometric relations between states. Moreover, typically, odometric measures between states are not effected by the actions, and any possible effect that a specific action, responsible for a transition, has on the odometric error is reflected in the variance around the mean odometric relation.

We have introduced the basic formal model that we use for representing environments and the robot interaction with them. The rest of the formal framework, namely, a statement of the learning problem and the basic algorithm for learning the model from data, is described in the following chapter.

[2]We do not discuss here the reward component of POMDP models since rewards are usually associated with tasks and goals that the planner has to accomplish, and is not always an "objective" part of the world in which the robot moves.

Chapter 4

Learning HMMs with Odometric Information

This chapter introduces the learning problem for HMMs, and discusses the standard learning algorithm and the basics of our odometric extension to it. Convergence proofs for the resulting algorithm are also provided. The augmented HMM learned by the algorithm is of the most restricted type, as given in Chapter 3. As we elaborate the model in the following chapters, the learning algorithms are also extended, as described in Chapters 5, 8 and 10.

4.1 The Learning Problem

The learning problem for hidden Markov models can be generally stated as follows: *Given an experience sequence* E *sampled from a model which is assumed to be a hidden Markov model, find a hidden Markov model that could have generated this sequence and is "useful" or "close to the original" according to some criterion.* Clearly this broad definition lacks a formal notion of what it means for the learned model to be close to the original model, or useful. We provide more rigorous criteria in the following paragraphs.

One common statistical approach is to look for a model λ that maximizes the *likelihood* of the data E given the model. Formally stated it maximizes: $\Pr(\mathsf{E}|\lambda)$. Another approach is to find a model that maximizes the *posterior probability* of the model given the data $\Pr(\lambda|\mathsf{E})$. This model is known as the *Maximum Aposteriori Probability* model (MAP). Note that the latter probability is typically more complicated to directly compute than the former. Moreover, by applying Bayes rule, it is easy to see that under the assumption that a priori all models are equally likely, the model that maximizes the likelihood also maximizes the posterior probability, hence the two criteria are equivalent. However, given the complicated landscape of typical likelihood functions in a multi-parameter domain, obtaining a maximum likelihood model is not feasible. All known practical methods can only guarantee a *local-maximum* likelihood model.

Another way of evaluating the quality of a learned model is by comparing it to the true model. We note that stochastic models (such as HMMs) induce a probability distribution over all observation sequences of a given length. The Kullback-Leibler [KL51] divergence of a learned distribution from a true one is a commonly used measure for estimating how good a learned model is. Obtaining a model that minimizes this measure is a possible learning goal. The culprit here is that in practice, when we learn a model from data, we do not have any ground truth to compare the learned model with. However, we can evaluate learning algorithms by measuring how well they perform on data obtained from known models. It is reasonable to expect that an algorithm that learns well from data that is generated from a model we do have, will perform well on data generated from an unknown model, assuming that the models we use indeed form a suitable representation of the true generating process. We discuss the Kullback-Leibler (KL) divergence in more detail in Section 7.2 in the context of evaluating our experimental results.

It is shown by Abe and Warmuth [AW92], that maximizing the likelihood and minimizing the KL-divergence is a related process, since a model that maximizes the likelihood of the training data also minimizes the KL-divergence of the distribution induced by the model with respect to the *training data distribution*. Ideally speaking, if the data is a faithful representative of the true model, finding a maximum likelihood model for the data and finding a minimum KL-divergence model with respect to the true model should amount to the same thing. More precisely, as the amount of training data tends to infinity, the training data distribution approaches the one induced by the true generating process, and the KL-divergence of the maximum likelihood model with respect to the true generating process tends to 0.

An evaluation scheme based on the KL-divergence, has a similar underlying idea to that of using *cross-validation* [Sto74, GHW79] for assessing how good a model is. When learning a model from given training data, we would like the model to be general enough to model data outside the training set, that is generated by the same process. When using cross-validation, parts of the available data are held out during the training process, and are only used for assessing the learned model, thus verifying that the model is indeed general enough to account for data outside the training set. The KL-divergence compares the learned model with the true one based on *newly generated* sequences of the true model that were not used during the training phase. Thus, it enables the assessment of the learned model's generality, without the need to hold-out any of the training data. In the general case, when the true model is not available, cross validation may prove useful for comparing the goodness of various learned models.

To summarize, the learning problem as we address it in this work, is that of obtaining a model by attempting to (locally) maximize the likelihood, while evaluating the results based on the KL-divergence with respect to the true underlying distribution, when such a distribution is available.

4.2 The Learning Algorithm

The learning algorithm for a hidden Markov model starts from an initial model λ' and is given an experience sequence, E; it returns a revised model λ, with the goal of maximizing the likelihood $\Pr(\mathsf{E}|\lambda)$. The experience sequence E is of length T; each element is a pair $\mathsf{E}_t = \langle r_t, V_t \rangle$, where r_t is the observed odometric relation between q_{t-1} and q_t and V_t is the observation vector at time t.

Our algorithm is a straightforward extension of the Baum-Welch algorithm to deal with the odometric information and the factored observation sets. The Baum-Welch algorithm is an expectation-maximization (EM) algorithm [DLR77]; it starts with an initial model λ_0 and alternates between

- the *E-step*: computing the state-occupation and state-transition probabilities, $\gamma_t(i) = Pr(q_t = s_i|\mathsf{E}, \lambda)$ and $\xi_t(i,j) = Pr(q_t = s_i, q_{t+1} = s_j|\mathsf{E}, \lambda)$, respectively, at each time t in the sequence, given E and the current model λ, and

- the *M-step*: finding a new model λ that maximizes $\Pr(\mathsf{E}|\lambda, \gamma, \xi)$.

An EM algorithm is guaranteed to provide monotonically increasing convergence of $\Pr(\mathsf{E}|\lambda)$. The Baum-Welch has been proven to be an EM algorithm [DLR77]; it has also been provably extended to real-valued observations [Lip82, Jua85]. Our algorithm, as described throughout the rest of this section, uses the additional matrix, R, and enforces the first two geometric consistency constraints on the M-step, but like the standard Baum-Welch it is still guaranteed to converge to a local maximum of the likelihood function. The proof is along the lines of the one presented by Juang et al. [JLS86] for the standard Baum-Welch algorithm, and is given in Section 4.3.

4.2.1 Computing State-Occupation Probabilities

Following Rabiner [Rab89], we first compute the forward (α) and backward (β) matrices. When all measurements are discrete, $\alpha_t(i)$ is the probability of observing E_0 through E_t and $q_t = s_i$, given λ; $\beta_t(i)$ is the probability of observing E_{t+1} through E_{T-1} given $q_t = s_i$ and λ. Formally:

$$\alpha_t(i) = Pr(\mathsf{E}_0, \ldots, \mathsf{E}_t, q_t = s_i|\lambda) \ ,$$

$$\beta_t(i) = Pr(\mathsf{E}_{t+1}, \ldots, \mathsf{E}_{T-1}|q_t = s_i, \lambda) \ .$$

When some of the measurements are continuous (as is the case with R), these matrices contain probability density values rather than probabilities.

The forward procedure for calculating the α matrix is initialized with

$$\alpha_0(i) = \begin{cases} {b_0}^i & \text{if } \pi_i = 1 \\ 0 & \text{otherwise} \ , \end{cases}$$

and continued for $0 < t \leq T-1$ with

$$\alpha_t(j) = \sum_{i=0}^{N-1} \alpha_{t-1}(i) A_{i,j} f(r_t|R_{i,j}) b_t^j \ . \tag{4.1}$$

$f(r_t|R_{i,j})$ denotes the *density* at point r_t according to the normal distribution represented by the means and variances in entry i, j of the relation matrix R, and b_t^j is the probability of observing vector v_t in state s_j; that is, $b_t^j = \prod_{i=0}^{l} B_{i,j,v_t[i]}$.

The backward procedure for calculating the β matrix is initialized with

$$\beta_{T-1}(j) = 1 \ ,$$

and continued for $0 \leq t < T-1$ with

$$\beta_t(i) = \sum_{j=0}^{N-1} \beta_{t+1}(j) A_{i,j} f(r_{t+1}|R_{i,j}) b_{t+1}^j \ . \tag{4.2}$$

Given α and β, we now compute the state-occupation and state-transition probabilities, γ and ξ. The state-occupation probabilities are computed as follows:

$$\gamma_t(i) = \Pr(q_t = s_i|\mathsf{E}, \lambda) \ = \ \frac{f_1(q_t = s_i, \mathsf{E}|\lambda)}{f_2(\mathsf{E}|\lambda)}$$

$$= \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=0}^{N-1} \alpha_t(j)\beta_t(j)} \quad , \tag{4.3}$$

where f_1, f_2 are density functions. Similarly, the state-transition probabilities are computed as:

$$\begin{aligned} \xi_t(i,j) &= \Pr(q_t = s_i, q_{t+1} = s_j | \mathsf{E}, \lambda) \\ &= \frac{\alpha_t(i) A_{i,j} b^j{}_{t+1} f(r_{t+1}|R_{i,j}) \beta_{t+1}(j)}{\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} \alpha_t(i) A_{i,j} b^j{}_{t+1} f(r_{t+1}|R_{i,j}) \beta_{t+1}(j)} \quad . \end{aligned} \tag{4.4}$$

We note that the numerator and the denominator in the fractions are both density functions, but the quotient is a discrete probability function. These are essentially the same formulae appearing in Rabiner's tutorial [Rab89], but they also take into account the density of the relational observation.

4.2.2 Updating Model Parameters

At this phase of the algorithm, the goal is to find a new model, $\overline{\lambda}$, that maximizes $\Pr(\mathsf{E}|\lambda, \gamma)$. Generally, this is simply done using maximum-likelihood estimation of the probability distributions in A and B by computing expected transition and observation frequencies. It is more difficult in our model than in regular HMMs, because we must also compute a new relation matrix, R, under the constraint that it remain geometrically consistent. Through the rest of this chapter we use the notation $\overline{y}$ to denote a reestimated value, and y to denote the current value.

The A and B matrices can be straightforwardly reestimated; $\overline{A}_{i,j}$ is the expected number of transitions from s_i to s_j divided by the expected number of transitions from s_i:

$$\overline{A}_{i,j} = \frac{\sum_{t=0}^{T-2} \xi_t(i,j)}{\sum_{t=0}^{T-2} \gamma_t(i)} \quad . \tag{4.5}$$

$\overline{B}_{i,j,o}$ is the expected number of times o is observed along the ith dimension when in s_j divided by the expected number of times of being in s_j:

$$\overline{B}_{i,j,o} = \frac{\sum_{t=0}^{T-1} I(V_t[i] = o)\gamma_t(j)}{\sum_{t=0}^{T-1} \gamma_t(i)} \quad , \tag{4.6}$$

where $I(c)$ is an indicator function with value 1 if c is true and 0 otherwise.

If we were not enforcing geometrical consistency, the R matrix would be reestimated by:

$$\overline{\mu}_{i,j}^{m} \overset{\text{def}}{=} \mu(\overline{R}_{i,j}[m]) = \frac{\sum_{t=0}^{T-2} r_t[m]\xi_t(i,j)}{\sum_{t=0}^{T-2} \xi_t(i,j)} \tag{4.7}$$

$$\overline{\sigma}_{i,j}^{m} \overset{\text{def}}{=} \sigma^2(\overline{R}_{i,j}[m]) = \frac{\sum_{t=0}^{T-2} (r_t[m] - \overline{\mu}_{i,j}^{m})^2 \xi_t(i,j)}{\sum_{t=0}^{T-2} \xi_t(i,j)} , \tag{4.8}$$

where $m \in \{x,\, y\}$.

However, the geometrical constraints induce interdependencies among the optimal mean estimates as well as between optimal variance estimates and mean estimates. Parameter estimation under this form of constraints is almost untreated in main-stream statistics [Bar84] and we found no previous existing solutions to the estimation problem we are facing. As an illustration consider the following constrained estimation problem of 2 normal means.

Example 4.1 *Suppose we are given two sample sets of points $P = \{p_1, p_2, \ldots, p_n\}$ and $Q = \{q_1, q_2, \ldots, q_k\}$. We are told that they were independently drawn from two distinct normal distributions with means μ_P, μ_Q and variances σ_P^2, σ_Q^2, respectively. We are asked to find maximum likelihood estimates for the two distribution parameters. Moreover, we are also told that the means of the two distributions are related, such that $\mu_Q = -\mu_P$. This setting is shown in Figure 4.1.*

If not for the latter constraint, the task is simple [DeG86], and we have:

$$\mu_P = \frac{\sum_{i=1}^{n} p_i}{n}\ ,\ \mu_Q = \frac{\sum_{j=1}^{k} q_j}{k}\ ,\ \sigma_P^2 = \frac{\sum_{i=1}^{n} (p_i - \mu_x)^2}{n}\ ,\ \sigma_Q^2 = \frac{\sum_{j=1}^{k} (q_j - \mu_y)^2}{k}\ . \tag{4.9}$$

However, the constraint $\mu_P = -\mu_Q$ forces us to find a single mean value μ and set the other one to its negated value, $-\mu$. Intuitively speaking, when choosing such a maximum likelihood single mean, the sample that is more concentrated should have more effect and the sample that varies more should be more "submissive". This way the overall sample deviation from the means would be minimized and the likelihood of the data maximized. Thus, there exists mutual dependence between the estimation of the mean and the estimation of the variance, as opposed to the estimation given in formulae 4.9, in which the optimal mean estimation depends solely on the sampled values.

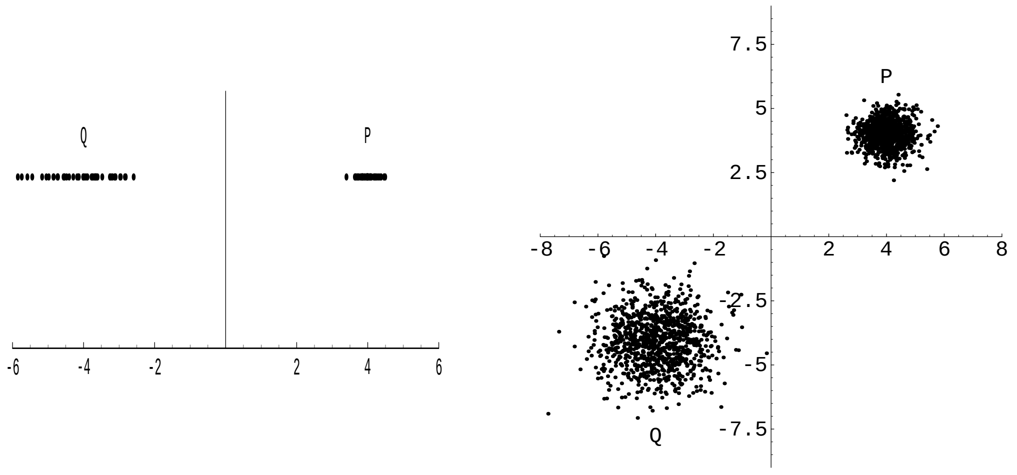

Figure 4.1: Examples of two sets of normally distributed points with constrained means, in 1 and 2 dimensions.

Since the samples are independently drawn, their joint likelihood function is:

$$f(P,Q|\mu_P,\mu_Q,\sigma_P^2,\sigma_Q^2)=\prod_{i=1}^{n}\frac{e^{\frac{-(p_i-\mu_P)^2}{2\sigma_P^2}}}{\sqrt{2\pi}\sigma_P}\cdot\prod_{j=1}^{k}\frac{e^{\frac{-(q_j-\mu_Q)^2}{2\sigma_Q^2}}}{\sqrt{2\pi}\sigma_Q}\ .$$

The log of the joint likelihood function under the constraint $\mu_Q=-\mu_P$ is therefore:

$$\sum_{i=1}^{n}\left(\frac{-(p_i-\mu_P)^2}{2\sigma_P^2}-\log(\sqrt{2\pi}\sigma_P)\right)+\sum_{j=1}^{k}\left(\frac{-(q_j+\mu_P)^2}{2\sigma_Q^2}-\log(\sqrt{2\pi}\sigma_Q)\right)\ . \qquad (4.10)$$

By taking the derivatives of expression 4.10 with respect to μ_P, σ_P and σ_Q and equating them to 0, while using the constraint $\mu_Q=-\mu_P$, we obtain the following set of mutual equations for maximum likelihood estimators:

$$\mu_P=\frac{\sigma_Q^2\sum_{i=1}^{n}p_i-\sigma_P^2\sum_{j=1}^{k}q_j}{n\sigma_Q^2+k\sigma_P^2},\quad \mu_Q=-\mu_P,\quad \sigma_P^2=\frac{\sum_{i=1}^{n}(p_i-\mu_P)^2}{n},\quad \sigma_Q^2=\frac{\sum_{j=1}^{k}(q_j+\mu_P)^2}{k}\ .$$

By substituting the expressions for σ_P and σ_Q into the expression for μ_P, we obtain a cubic equation which is cumbersome, hence is not given here, but still solvable (in this simple case). The solution provides a maximum likelihood estimate for the mean and variance under the above constraint.

In the case where the two samples are assumed to have the same variance, *the variance factors in the expression for μ_P above cancel out, and the estimate for μ_P is simply:*

$$\mu_P=-\mu_Q=\frac{\sum_{i=1}^{n}p_i-\sum_{j=1}^{k}q_j}{n+k}\ ,$$

which agrees with the intuitive solution to the problem. Under this assumption, the maximum likelihood estimate for σ_Q^2, σ_P^2 needs to take into account the equality constraint, and can be expressed as:

$$\sigma_P^2 = \sigma_Q^2 = \frac{\sum_{i=1}^{n}(p_i - \mu_P)^2 + \sum_{j=1}^{k}(q_j + \mu_P)^2}{n+k} \ .$$

We now proceed to the update of the relation matrix under constraints. For clarity, we discuss only the first two geometrical constraints at this stage. Enforcing the additivity constraint is discussed in Chapter 10.

Zero distances between states and themselves are trivially enforced, by setting all the diagonal entries in the R matrix to 0, with a small variance, along the x and y dimension. *Anti-symmetry* is enforced by using the data from s_j to s_i as well as from s_i to s_j when reestimating $\mu(R_{i,j})$. However, we note that the variance has to be taken into account, and we obtain the following set of mutual equations:

$$\overline{\mu}_{i,j}^m = \frac{\sum_{t=0}^{T-2}\left[\frac{r_t[m]\xi_t(i,j)}{(\overline{\sigma}_{i,j}^m)^2} - \frac{r_t[m]\xi_t(j,i)}{(\overline{\sigma}_{j,i}^m)^2}\right]}{\sum_{t=0}^{T-2}\left[\frac{\xi_t(i,j)}{(\overline{\sigma}_{i,j}^m)^2} + \frac{\xi_t(j,i)}{(\overline{\sigma}_{j,i}^m)^2}\right]} \ , \qquad (4.11)$$

$$(\overline{\sigma}_{i,j}^m)^2 = \frac{\sum_{t=0}^{T-2}[\xi_t(i,j)(r_t[m] - \overline{\mu}_{i,j}^m)^2]}{\sum_{t=0}^{T-2}\xi_t(i,j)} \ . \qquad (4.12)$$

For the x and y dimensions we get a complicated but still solvable equation of the 3^{rd} degree. However, for the more general cases involving information regarding the orientation of the robot (see Chapters 5, 8), as well as when complete additivity is enforced (see Chapter 10) there are no such closed form reestimation formulae.

Hence, rather than have very complicated reestimation formulae, we use a *lag-behind* update rule; the yet-unupdated estimate of the variance is used for calculating a new estimate for the mean, and the newly updated mean estimate is then used to update the variance. Thus, the mean is updated using a variance parameter that lags behind it in the update process, and the reestimation formula 4.11 needs to use $\sigma^m_{i,j}$ rather than $\overline{\sigma}^m_{i,j}$:

$$\overline{\mu}^m_{i,j} = \frac{\sum_{t=0}^{T-2} \left[\frac{r_t[m]\xi_t(i,j)}{(\sigma^m_{i,j})^2} - \frac{r_t[m]\xi_t(j,i)}{(\sigma^m_{j,i})^2} \right]}{\sum_{t=0}^{T-2} \left[\frac{\xi_t(i,j)}{(\sigma^m_{i,j})^2} + \frac{\xi_t(j,i)}{(\sigma^m_{j,i})^2} \right]} \ . \tag{4.13}$$

A possible alternative to our lag-behind approach is to update the mean as though the assumption $\sigma_{j,i} = \sigma_{i,j}$ indeed holds. Under this assumption, the variance terms in equation 4.11 cancel out, and the mean update is independent of the variance once again. Then the variances are updated as stated in equation 4.12, *without* assuming any constraints over them. This approach was taken in earlier stages of this work [SK97b, SK98]. We experimentally studied various update policies for learning constrained Gaussian parameters[1]. The experimental results under the restricted experimental settings suggest that the lag-behind strategy is superior to the others and very close to the actual maximum likelihood estimation method. Moreover, a similar approach was taken by other researchers when using EM in highly non-linear optimization problem, termed *one step late* update [MK97]. It turns out, as we also show in Section 4.3.3, that this update approach falls under the *generalized* EM family of algorithms, which have similar properties to the EM algorithms.

The complexity of the above algorithm per iteration is still $O(TN^2)$, like the standard Baum-Welch method. Note that the constants are significant here, since the calculation of formulae 4.1 and 4.2 requires the evaluation of exponential terms, which is a time consuming operation. This cost can be significantly reduced through the use of lookup-tables, although this is not currently implemented in our code.

4.2.3 Stopping Criterion

As stated in the beginning of this section, and proved in Section 4.3, our algorithm is an EM algorithm and as such it is guaranteed to converge to a local maximum of the likelihood function. Moreover, a local maximum is reached if and only if the model parameters have reached a fixed point and are no longer changed by the reestimation procedure.

To practically determine that the algorithm has indeed converged, we can compare the value of the likelihood function between consecutive iterations. When the change in the

[1] I am grateful to Luis Ortiz for lending his code and expertise, as well as for conducting these experiments with me.

likelihood value is less than a predetermined small ϵ, we can assume that convergence has been reached, and stop the algorithm.

Alternatively, we can compare the amount of change in the *model parameters* themselves between iterations, and when the change in each parameter is less than a predetermined ϵ, decide that a fixed point has been reached, which implies the convergence of the algorithm.

Note that the odometric data and the odometric relation matrix are used as an aid towards obtaining the transition and observation matrices, and therefore, a stopping criterion need only take into consideration the transition and observation matrices or the likelihood of the observation data, rather than the relation matrix or the likelihood of the odometric data.

The latter of the two stopping criteria can be viewed as more conservative. The reason is that the likelihood function expression involves a product of the model's parameters [Rab89]. Since these parameters are all probabilities (as we are only taking into account the transition and observation distribution matrices), they are all numbers between 0 and 1. Therefore, when the change in each of them is less than ϵ, the change in the likelihood value is typically much smaller.

In our implementation of the algorithm, we use this second criterion, and determine that the algorithm has converged when the change in each of the entries of the transition matrix, A, and the observation matrix, B, from one iteration to the next does not exceed a predefined ϵ. (We set $\epsilon = 0.001$ in our implementation). By comparing only the change in the transition and observation matrices, we also enable a fair comparison of the number of iterations required for convergence with and without the use of odometric information, as described in Chapters 7 and 9 of this work.

4.2.4 Extending the Algorithm for Learning POMDPs

To extend the algorithm given above to learn a complete POMDP, each item in the experience sequence E contains, in addition to the observation and the odometric relation, the *action* that caused the transition associated with the odometric relation.

For each action there is a separate pair of matrices A and B. Hence, the forward-backward procedure as described in Section 4.2.1, and in particular Equations 4.1 and 4.2, must take into account at each time t the transition probabilities A_{ij} and the observation probabilities b^j that are associated with the specific action taken at time t. Furthermore, the update procedure for the A and B matrices, (formulae 4.5 and 4.6), for a particular action a, only takes into account the estimated state transitions and observations that are a result of a in the data sequence. The update of the relation matrix R does not need to take any action information into account since R is a single common matrix for the whole model.

The time complexity of learning a POMDP compared with that of learning an HMM is not significantly different since the *forward-backward* procedure described in Section 4.2.1, which is the most computationally-intensive part of the algorithm, does not require any additional computational steps. The only difference in this procedure is the one mentioned above. Only the final update of the A and B matrices, needs to be performed separately for each action, but this stage is not a computational bottleneck. Therefore, the overall time requirements remain almost unchanged, under the assumption that the number of possible actions is typically much smaller than the number of states in the model. A factor that is likely to make learning a POMDP more time consuming, and needs to be taken into account, is the larger number of model parameters introduced due to the multiple actions. In order to facilitate the learning of useful models, longer data sequences, and therefore proportionally more computation time, may be required.

4.3 Correctness Proof of the Reestimation Formulae

For the kind of iterative reestimation algorithms that we use, proving the correctness of the reestimation formulae means proving that through repeated reestimation, the likelihood does not decrease, and that the algorithm converges to a fixed-point model $\overline{\lambda}$, which is a local maximum of the likelihood function $P(\mathsf{E}|\lambda)$, where E is the observed experience sequence. We formalize this in the following theorem:

Theorem 4.1 *Let λ' be the current model, E be the experience sequence, and $\overline{\lambda}$ be the reestimated model according to the reestimation formulae 4.5, 4.6, and either 4.7 and 4.8, or 4.13 and 4.12. Then $Pr(\mathsf{E}|\lambda') \leq Pr(\mathsf{E}|\overline{\lambda})$, and $\lambda' = \overline{\lambda}$ if and only if λ' is a local maximum of $Pr(\mathsf{E}|\lambda)$ as a function of λ.*

Proof: There are several proof techniques for the correctness of the reestimation formulae for the standard Baum-Welch algorithm (under various kinds of observation matrix B) [BPS$^+$70, DLR77, LRS83, Jua85, JLS86]. Our proof uses the same approach as the latter two. It is straightforward to show that maximization of the likelihood function with respect to each of the parameters separately is equivalent to its maximization with respect the complete model. Hence, we break up the proof, and prove that each of the reestimation formulae indeed improves the likelihood function with respect to the associated reestimated parameter. Since the likelihood function with respect to the A and B matrices is a discrete probability distribution, it is bounded from above, and the convergence of the process is guaranteed. For the relation matrix reestimation procedure, convergence is also guaranteed through a more complicated condition given by Wu [Wu83], and which holds for the exponential family of distributions [MK97]. Hence convergence is guaranteed.

4.3.1 Transitions and Observations

To prove the correctness of formulae (4.5) and (4.6), we use the central theorem of Baum and Sell [BS68], which states[2] that *for* $x = \{x_{ij}\}$ *s.t.* $x_{ij} > 0,\ 0 \leq j \leq N-1,$ *and* $\sum_{j=0}^{N-1} x_{ij} = 1$*, given a homogeneous polynomial* P *in the variable* x_{ij}*, with nonnegative coefficients, the transformation*

$$\overline{x_{ij}} = \frac{x_{ij}\frac{\partial P}{\partial x_{ij}}}{\sum\limits_{k=0}^{N-1} x_{ik}\frac{\partial P}{\partial x_{ik}}} \tag{4.14}$$

satisfies $P(x) \leq P(\overline{x})$*, and* $\overline{x} = x$ *if and only if* x *is a local maximum of* P*.*

The density expression $\Pr(\mathsf{E}|\lambda) = \sum_{i=0}^{N-1} \alpha_{T-1}(i) = \sum_{i=0}^{N-1} \alpha_{T-1}(i)\beta_{T-1}(i)$, which we want to maximize, is indeed a homogeneous polynomial in A_{ij} and in B_{ijo}. Both A_{ij} and B_{ijo} are discrete probability distributions, therefore are positive and satisfy $\sum_{j=0}^{N-1} A_{ij} = 1$ and $\sum_{o\in O_i} B_{ijo} = 1$. Hence, according to the above theorem the reestimation formula for A_{ij}, that leads to a local maximization of $P(E|A_{ij})$ is:

$$\overline{A_{ij}} = \frac{A_{ij}\frac{\partial P(\mathsf{E}|\lambda)}{\partial A_{ij}}}{\sum\limits_{k=0}^{N-1} A_{ik}\frac{\partial P(\mathsf{E}|\lambda)}{\partial A_{ik}}} \ . \tag{4.15}$$

We now need to show that the right-hand side of formula (4.15) is equal to that of (4.5). To do this we need to show that:

$$\frac{\partial P(E|\lambda)}{\partial A_{ij}} = \sum_{t=0}^{T-2} \alpha_t(i) b^j{}_{t+1} f(r_{t+1}|R_{i,j})\beta_{t+1}(j) \ . \tag{4.16}$$

By substituting the right hand side expression into (4.15), and using equations (4.3,4.4) we get the desired equality.

[2]Baum and Sell's theorems are actually somewhat stronger and the statement given here is just one consequence.

By induction on k, $0 \leq k \leq T-1$, it is easy to show that:

$$\sum_{i=0}^{N-1} \frac{\partial \alpha_k(i)}{\partial A_{ij}} \beta_k(i) = \sum_{t=0}^{k-1} \alpha_t(i) b^j{}_{t+1} f(r_{t+1}|R_{i,j}) \beta_{t+1}(j) \ .$$

For $k = T-1$ we get (4.16), which concludes the proof of formula (4.5). The proof for B_{ijo} (formula (4.6)) is almost identical.

4.3.2 Odometric Relations

We note that the density expression, $\Pr(\mathsf{E}|\lambda)$, is not a polynomial in $\mu(R_{i,j}[m])$ and $\sigma^2(R_{i,j}[m])$. Hence the theorem by Baum and Sell can not be applied here. Still, since we assume that the odometric relations along the x and y dimensions are normally distributed, in the unconstrained case, their reestimation procedure is an instance of the exponential family reestimation, discussed by Dempster et al [DLR77]. However, for the sake of completeness and for an easier discussion of the constrained case, we provide a complete proof, using the technique of maximizing Baum's auxiliary function, following Section 4 of the paper by Baum et al. [BPS$^+$70]. We denote by λ_ρ, where ρ is some relation matrix , the model whose A and B matrices are the same as those of λ, but whose relation matrix R is replaced by the matrix ρ.

We start by making the observation that if $\mathcal{S}$ is the set of all state sequences of length T, i.e. $\mathcal{S} = \{s\}$ where $s = s_0, \ldots, s_{T-1}$ is a sequence of states of length T, the density $P(\mathsf{E}|\lambda)$ can be expressed as

$$P(\mathsf{E}|\lambda) = \sum_{s \in \mathcal{S}} P(\mathsf{E}, s|\lambda) = \sum_{s \in \mathcal{S}} P(\mathsf{E}|s, \lambda) P(s|\lambda) \ .$$

We can rewrite $P(\mathsf{E}|s, \lambda)$ and $P(s|\lambda)$ as

$$P(\mathsf{E}|s,\lambda) = \prod_{t=0}^{T-1} b_t^{s_t} \prod_{t=1}^{T-1} f(r_t|R_{s_{t-1},s_t}) \quad \text{and} \quad P(s|\lambda) = \pi_{s_0} \prod_{t=1}^{T-1} A_{s_{t-1},s_t} \ .$$

Thus, $P(\mathsf{E}|\lambda)$ can be expressed as

$$\sum_{s \in \mathcal{S}} \tau(s) \prod_{t=1}^{T-1} f(r_t|R_{s_{t-1},s_t}) \ ,$$

where $\tau(s)$ is a product of initial, transition and observation probabilities. Recalling that $f(r_t|R_{i,j})$ denotes the density of r_t according to the D-variate independent normal distribution with the parameters stored in $R_{i,j}$, we rewrite it as

$$f(r_t|R_{i,j}) = \prod_{m=1}^{D} \frac{f_{ij}^m(r_t^m)}{\sqrt{2\pi}\sigma_{ij}^m}, \tag{4.17}$$

where $r_t^m \overset{\text{def}}{=} r_t[m]$ and $f_{ij}^m(r_t^m) \overset{\text{def}}{=} e^{-(r_t^m-\mu_{ij}^m)^2/2(\sigma_{ij}^m)^2}$. We also use the notation: $\overline{f}_{ij}^m(r_t^m) \overset{\text{def}}{=} e^{-(r_t^m-\overline{\mu}_{ij}^m)^2/2(\overline{\sigma}_{ij}^m)^2}$ and $f_{ij}(r_t) \overset{\text{def}}{=} \prod_{m=1}^{D} f_{ij}^m(r_t^m)$.

Baum et al. [BPS+70] introduce an auxiliary function, Q, and prove that maximizing it is the same as increasing the likelihood. More formally, the $\overline{R}$ that maximizes the auxiliary function

$$\begin{aligned} Q(R,\overline{R}) &\overset{\text{def}}{=} \sum_{s\in\mathcal{S}} P(\mathsf{E},s|\lambda)\log(\prod_{t=1}^{T-1} f(r_t|\overline{R}_{s_{t-1},s_t})) \\ &= \sum_{s\in\mathcal{S}} P(\mathsf{E},s|\lambda)\sum_{t=1}^{T-1}\log(f(r_t|\overline{R}_{s_{t-1},s_t})) \end{aligned}$$

also satisfies $P(\mathsf{E}|\lambda_{\overline{R}}) \geq P(\mathsf{E}|\lambda_R)$.

Since the $\overline{R}$ that maximizes $Q(R,\overline{R})$ also maximizes the same expression in which $\sqrt{2\pi}f_{ij}^m(r_t^m)$ is substituted for $f_{ij}^m(r_t^m)$, we can ignore the $\sqrt{2\pi}$ factor in (4.17) and rewrite Q as

$$Q(R,\overline{R}) = \sum_{s\in\mathcal{S}} P(\mathsf{E},s|\lambda)\sum_{t=1}^{T-1}\sum_{m=1}^{D}[\log(\overline{f}_{s_{t-1},s_t}^m(r_t^m)) - \log(\overline{\sigma}_{s_{t-1},s_t}^m)] \ . \tag{4.18}$$

Since the normal distribution is strictly log-concave, a slight adaptation to the proof of Theorem 4.1 in [BPS+70] is sufficient for showing that Q above has a unique global maximum as a function of $\overline{\mu}_{ij}^m$ and $\overline{\sigma}_{ij}^m$, which is the unique point in which the partial derivatives of Q according to $\overline{\mu}_{ij}^m$ and $\overline{\sigma}_{ij}^m$ are 0. We now show that the reestimation formulae (4.7) and (4.8) indeed find the maximizing $\overline{\mu}_{ij}^m$ and $\overline{\sigma}_{ij}^m$.

For a pair of states i, j and an odometry component $m \in \{x, y\}$ we can express the restriction of the auxiliary function Q to transitions from i to j and to the m^{th} odometric component, m, as

$$Q_{ij}^m(R,\overline{R}) = \sum_{s\in\mathcal{S}} P(\mathsf{E},s|\lambda) \sum_{\substack{t\text{ s.t.}\\ s_{t-1}=i\\ s_t=j}} (\log(\overline{f}_{s_{t-1},s_t}^m(r_t^m)) - \log(\overline{\sigma}_{s_{t-1},s_t}^m)) \ .$$

Observing that $\xi_{t-1}(i,j) = \sum_{\substack{s\in\mathcal{S}\text{ s.t.}\\ s_{t-1}=i\\ s_t=j}} P(\mathsf{E},s|\lambda)$ allows us to rewrite $Q_{ij}^m(R,\overline{R})$ as

$$Q_{ij}^m(R,\overline{R}) = \sum_{t=0}^{T-2}\xi_t(i,j)(\log(\overline{f}_{i,j}^m(r_{t+1}^m)) - \log(\overline{\sigma}_{ij}^m)) \ . \tag{4.19}$$

Since $\frac{\partial Q}{\partial\overline{\mu}_{ij}^m} = \frac{\partial Q_{ij}^m}{\partial\overline{\mu}_{ij}^m}$ and $\frac{\partial Q}{\partial\overline{\sigma}_{ij}^m} = \frac{\partial Q_{ij}^m}{\partial\overline{\sigma}_{ij}^m}$, showing that the partial derivatives of Q_{ij}^m with respect to $\overline{\mu}_{ij}^m$ and $\overline{\sigma}_{ij}^m$ are 0 whenever equations (4.7) and (4.8) are satisfied, concludes our proof. The differentiation is straightforward and is provided in Appendix B.1 .

4.3.3 Constrained Odometric Relations

The correctness of formulae 4.13 and 4.12, under our lag-behind update policy, is proved by showing that these update rules are instances of *generalized* EM. Dempster et al. [DLR77] introduced this notion, and it is explained in detail by McLachlan and Krishnan [MK97]. The idea is that by merely *improving* the auxiliary function Q, rather than *maximizing* it, at each iteration, we are still guaranteed to improve the likelihood function. Therefore, it is not necessary to find an update rule that maximizes Q but simply one that improves it.

The proof technique presented here uses direct enforcement of the anti-symmetry constraint in the expression for Q. This is in contrast to the use of Lagrange multipliers in earlier work [SK97a].

From equations (4.18) and (4.19) we have:

$$Q(R,\overline{R}) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\sum_{m=1}^{D} Q_{ij}^{m}(R,\overline{R}) \ . \tag{4.20}$$

By isolating all terms in which $i = j$ we can rewrite this expression as:

$$Q(R,\overline{R}) = \sum_{i=0}^{N-1}\sum_{m=1}^{D} Q_{ii}^{m}(R,\overline{R}) + \sum_{i=0}^{N-1}\sum_{j=(i+1)}^{N-1}\sum_{m=1}^{D}[Q_{ij}^{m}(R,\overline{R}) + Q_{ji}^{m}(R,\overline{R})] \ . \tag{4.21}$$

In order to increase the sum in (4.21), under the constraint $\overline{\mu}_{ij}^{m} = -\overline{\mu}_{ji}^{m}$, with respect to the parameters $\overline{\mu}_{ij}^{m}$ and $\overline{\sigma}_{ij}^{m}$, it is sufficient to increase the sum of each pair $(Q_{ij}^{m}(R,\overline{R}) + Q_{ji}^{m}(R,\overline{R}))$ with respect to these parameters, since all other pairs do not contain them. Thus, we are left with the following expression:

$$L_{ij}^{m}(\overline{\mu}_{ij}^{m}, \overline{\mu}_{ji}^{m}, \overline{\sigma}_{ij}^{m}, \overline{\sigma}_{ji}^{m})$$
$$= \sum_{t=0}^{T-2}\left[\xi_t(i,j)\left(\log(e^{-\frac{(r_t^m-\overline{\mu}_{ij}^{m})^2}{2(\overline{\sigma}_{ij}^{m})^2}}) - \log(\overline{\sigma}_{ij}^{m})\right) + \xi_t(j,i)\left(\log(e^{-\frac{(r_t^m-\overline{\mu}_{ji}^{m})^2}{2(\overline{\sigma}_{ji}^{m})^2}}) - \log(\overline{\sigma}_{ji}^{m})\right)\right] \ .$$

We enforce the anti-symmetry constraint by substituting μ_{ji}^{m} by $-\mu_{ij}^{m}$ in the above expression, and obtain the following expression which we need to increase with respect to $\overline{\mu}_{ij}^{m}$, $\overline{\sigma}_{ij}^{m}$ and $\overline{\sigma}_{ji}^{m}$:

$$
\begin{aligned}
&L^m_{ij}(\overline{\mu}^m_{ij}, \overline{\sigma}^m_{ij}, \overline{\sigma}^m_{ji}) \\
&= \sum_{t=0}^{T-2} \left[\xi_t(i,j) \left(\log(e^{-\frac{(r^m_t - \overline{\mu}^m_{ij})^2}{2(\overline{\sigma}^m_{ij})^2}}) - \log(\overline{\sigma}^m_{ij}) \right) + \xi_t(j,i) \left(\log(e^{-\frac{(r^m_t + \overline{\mu}^m_{ij})^2}{2(\overline{\sigma}^m_{ji})^2}}) - \log(\overline{\sigma}^m_{ji}) \right) \right] \\
&= \sum_{t=0}^{T-2} \left[\xi_t(i,j) \left(-\frac{(r^m_t - \overline{\mu}^m_{ij})^2}{2(\overline{\sigma}^m_{ij})^2} - \log(\overline{\sigma}^m_{ij}) \right) + \xi_t(j,i) \left(-\frac{(r^m_t + \overline{\mu}^m_{ij})^2}{2(\overline{\sigma}^m_{ji})^2} - \log(\overline{\sigma}^m_{ji}) \right) \right] \quad .
\end{aligned}
$$

Let μ^m_{ij}, σ^m_{ij} and σ^m_{ji} be the current values of the parameters. To increase L^m_{ij} we do the following:

1. Temporarily fix $\overline{\sigma}^m_{ij}, \overline{\sigma}^m_{ji}$ to be the current $\sigma^m_{ij}, \sigma^m_{ji}$. Denote by $\widehat{L}^m_{ij}(\overline{\mu}^m_{ij})$ the function obtained from L^m_{ij} through this instantiation of the $\overline{\sigma}$ parameters. Formally stated:
$$\widehat{L}^m_{ij}(\overline{\mu}^m_{ij}) \stackrel{\mathrm{def}}{=} L^m_{ij}(\overline{\mu}^m_{ij}, \sigma^m_{ij}, \sigma^m_{ji}) \quad .$$

2. Find the value $\widehat{\mu}^m_{ij}$ that maximizes $\widehat{L}^m_{ij}(\overline{\mu}^m_{ij})$.

3. Set $\overline{\mu}^m_{ij}$ in L^m_{ij} to the value $\widehat{\mu}^m_{ij}$ found in step 2. Denote by $\widehat{\widehat{L}}^m_{ij}(\overline{\sigma}^m_{ij}, \overline{\sigma}^m_{ji})$ the function obtained from L^m_{ij} through this instantiation of $\overline{\mu}^m_{ij}$. Formally stated:
$$\widehat{\widehat{L}}^m_{ij}(\overline{\sigma}^m_{ij}, \overline{\sigma}^m_{ji}) \stackrel{\mathrm{def}}{=} L^m_{ij}(\widehat{\mu}^m_{ij}, \overline{\sigma}^m_{ij}, \overline{\sigma}^m_{ji}) \quad .$$

4. Find values $\widehat{\sigma}^m_{ij}, \widehat{\sigma}^m_{ji}$ that maximize $\widehat{\widehat{L}}^m_{ij}$.

5. Set $\overline{\sigma}^m_{ij}, \overline{\sigma}^m_{ji}$ to be $\widehat{\sigma}^m_{ij}, \widehat{\sigma}^m_{ji}$.

Since $\widehat{\mu}^m_{ij}$ maximizes $\widehat{L}^m_{ij}(\overline{\mu}^m_{ij})$ we have:

$$L^m_{ij}(\widehat{\mu}^m_{ij}, \sigma^m_{ij}, \sigma^m_{ji}) \geq L^m_{ij}(\mu^m_{ij}, \sigma^m_{ij}, \sigma^m_{ji}) \quad .$$

Since $\widehat{\sigma}^m_{ij}, \widehat{\sigma}^m_{ji}$ maximize $\widehat{\widehat{L}}^m_{ij}(\overline{\sigma}^m_{ij}, \overline{\sigma}^m_{ji})$ we have:

$$L^m_{ij}(\widehat{\mu}^m_{ij}, \widehat{\sigma}^m_{ij}, \widehat{\sigma}^m_{ji}) \geq L^m_{ij}(\widehat{\mu}^m_{ij}, \sigma^m_{ij}, \sigma^m_{ji}) \quad .$$

By transitivity:

$$L^m_{ij}(\widehat{\mu}^m_{ij}, \widehat{\sigma}^m_{ij}, \widehat{\sigma}^m_{ji}) \geq L^m_{ij}(\mu^m_{ij}, \sigma^m_{ij}, \sigma^m_{ji}) \quad .$$

Hence, setting $\overline{\mu}^m_{ij}, \overline{\sigma}^m_{ij}, \overline{\sigma}^m_{ji}$ according to the above procedure *does not decrease* L^m_{ij}. If it does not *strictly increase* L^m_{ij}, then according to the generalized EM algorithm a maximum of the likelihood function is reached.

It is now left to obtain expressions for $\widehat{\mu}^m_{ij}$, $\widehat{\sigma}^m_{ij}$, and $\widehat{\sigma}^m_{ji}$.

To find $\widehat{\mu}_{ij}^m$, we take the derivative of $\widehat{L}_{ij}^m(\overline{\mu}_{ij}^m)$ with respect to $\overline{\mu}_{ij}^m$ and equate it to 0.

$$\frac{\partial \widehat{L}_{ij}^m}{\partial \overline{\mu}_{ij}^m} = \sum_{t=0}^{T-2}[\xi_t(i,j)\frac{(r_t^m - \overline{\mu}_{ij}^m)}{(\sigma_{ij}^m)^2} - \xi_t(j,i)\frac{(r_t^m + \overline{\mu}_{ij}^m)}{(\sigma_{ji}^m)^2}] \ .$$

By equating to 0 we get:

$$\overline{\mu}_{i,j}^m = \frac{\sum_{t=0}^{T-2}[\frac{r_t^m \xi_t(i,j)}{(\sigma_{i,j}^m)^2} - \frac{r_t^m \xi_t(j,i)}{(\sigma_{j,i}^m)^2}]}{\sum_{t=0}^{T-2}[\frac{\xi_t(i,j)}{(\sigma_{i,j}^m)^2} + \frac{\xi_t(j,i)}{(\sigma_{j,i}^m)^2}]} \ ,$$

which is identical the reestimation expression for $\overline{\mu}_{i,j}^m$ given in 4.13. It is easy to check that the second derivative of $\widehat{L}_{ij}^m(\overline{\mu}_{ij}^m)$ with respect to $\overline{\mu}_{ij}^m$ is negative, and therefore this is indeed a maximum rather than a minimum.

Similarly, to obtain $\widehat{\sigma}_{ij}^m$, we take the derivative of $\widehat{\widehat{L}}_{ij}^m(\overline{\sigma}_{ij}^m, \overline{\sigma}_{ji}^m)$ with respect to $\overline{\sigma}_{ij}^m$, and equate it to 0, and a similar process is done for finding $\widehat{\sigma}_{ji}^m$. The obtained expressions are:

$$(\overline{\sigma}_{i,j}^m)^2 = \frac{\sum_{t=0}^{T-2}[\xi_t(i,j)(r_t[m] - \overline{\mu}_{i,j}^m)^2]}{\sum_{t=0}^{T-2}\xi_t(i,j)} \quad (\overline{\sigma}_{j,i}^m)^2 = \frac{\sum_{t=0}^{T-2}[\xi_t(j,i)(r_t[m] + \overline{\mu}_{i,j}^m)^2]}{\sum_{t=0}^{T-2}\xi_t(j,i)} \ .$$

which agree with the update formulae 4.12. Again, the second derivative is negative, which ensures that this is indeed a maximum point. This concludes our proof. □

For the special case where $i = j$, the value $\overline{\mu}_{ij}^m$ is 0, which shows that the update formula indeed satisfies the first two of the three geometrical consistency constraints.

Chapter 5

Directional Data and Distributions

5.1 Motivation

Throughout our discussion so far, we have considered only two components of the odometric information gathered by the robot, namely, the x and y coordinates. However, an additional measure that is usually recorded is the change of the robot's heading, θ, as it moves from one state to the next. If a robot is standing in one position and takes the action of turning left or right, a respective change of heading of approximately $\pm 90°$ is recorded between the state prior to the turn and the state following the turn. Obviously, there is noise around this measure, as is the case for the x and y measures. In previous work [SK97b, SK97a], we did treat the change in heading as though it were simply normally distributed. However, the change in heading is different from that in x and y, in the sense that angular measurements are *cyclic*. That is, a change in heading of $90°$ is the same as a change of $450°$ or of $-270°$.

If we knew in advance, for every two states, the approximate change in heading ($\Delta\theta$) that the robot goes through when moving from one of them to the other, we could still have modeled it as though it were approximately normal with a mean $\Delta\theta$, and some small variance σ^2 [AC82]. We could adopt a convention of having all angles normalized to be within a cyclic range, e.g. $[-180°, 180°]$, (similarly we may use radians or other units), and always choose to take as the angular change between two points $min(|\theta|, 360° - |\theta|)$, and assign it the correct sign. However, *we do not know in advance the angular change between every two states.* We have a sequence of angular measurements and we estimate the probabilities of the states in which they were recorded, and take a *weighted mean* of the measurements in order to estimate the angular change between every two states. Thus, we are facing the following problem: *What is the interpretation of a "mean angle"?*

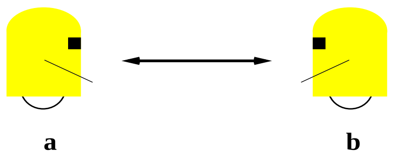

Figure 5.1: Robot changes heading from state a to state b.

As an example, consider the transition from state a to state b, as depicted in Figure 5.1. Suppose that we adopt the convention that angles are expressed as numbers between $-180°$ and $180°$. Also, suppose we have two (noisy) measurements of the angular distance from state a to state b: $-169°$ and $185°$. The simple average between these two measurements gives us an estimate of the mean heading change of $8°$. Obviously this is not the value that reflects even remotely the change of heading between the two states. A similar problem arises if we use a convention for expressing angles between $0°$ and $360°$. The problem lies in the fact that angles that are about $180°$ away from the mean angle greatly deviate from this mean, while angles that are about $360°$ away from the mean are actually very close to it. To capture this idea, the concept of *circular distribution* is required. Angular data plays a significant role in various aspects of both theory and mechanics of robotics, as well as other areas of computer science (e.g computer graphics). Since distributions over such data are not widely known to researchers in this area, (although the problematic aspect of such data has long been realized by statisticians), we provide here a brief introduction to the basic concepts and techniques used for handling circular data. In particular we concentrate on the *von Mises distribution*, which is a circular version of the normal distribution. Further discussion can be found in several statistical publications [GGD53, Mar72, KJ82a, KJ82b]. Section 5.4 returns to show how the theory is applied in our model and learning algorithm.

5.2 Statistics of Directional Data

Directional data in the 2-dimensional space can be represented as a collection of 2-dimensional vectors, $\{\langle x_1, y_1\rangle, \ldots \langle x_n, y_n\rangle\}$, on the unit circle, as shown in Figure 5.2. The 2-dimensional points can also be represented as the corresponding angles between the radii from the center of the unit circle and the x axis, $(\theta_1, \ldots, \theta_n)$, respectively. The relationship between the two representations is:

$$x_i = \cos(\theta_i), \qquad y_i = \sin(\theta_i), \qquad (1 \leq i \leq n) \ .$$

The vector mean of the n points, $\langle \overline{x}, \overline{y}\rangle$, is calculated as:

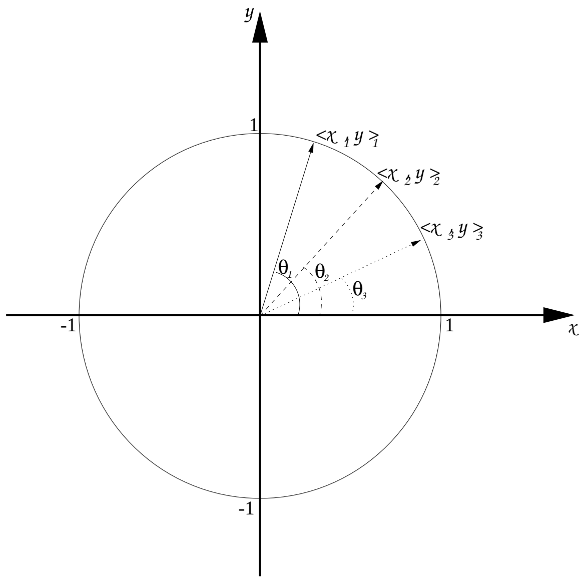

Figure 5.2: Directional data represented as angles and as vectors on the unit circle.

$$\overline{x} = \frac{\sum_{i=1}^{n} x_i}{n} = \frac{\sum_{i=1}^{n} \cos(\theta_i)}{n}, \qquad \overline{y} = \frac{\sum_{i=1}^{n} y_i}{n} = \frac{\sum_{i=1}^{n} \sin(\theta_i)}{n} \quad . \tag{5.1}$$

Using polar coordinates, we can express the mean vector in terms of angle, $\overline{\theta}$, and length, $\overline{a}$, where (except for the case $\overline{x} = \overline{y} = 0$):

$$\overline{\theta} = \arctan(\frac{\overline{y}}{\overline{x}}), \qquad \overline{a} = (\overline{x}^2 + \overline{y}^2)^{\frac{1}{2}} \quad . \tag{5.2}$$

The angle $\overline{\theta}$ is the mean angle, while the length $\overline{a}$ is a measure (between 0 and 1) of how concentrated the sample angles are around $\overline{\theta}$. The closer $\overline{a}$ is to 1, the more concentrated the sample is around the mean, which corresponds to a smaller sample variance.

Distributions that generate directional data are called *directional* or *circular* distributions. A function f is a density function of a continuous circular distribution if and only if:

$$f(x) \geq 0, \qquad \int_0^{2\pi} f(x)dx = 1 \quad .$$

A simple example of a circular distribution is the *uniform circular distribution*, whose density function is $f(\theta) = \frac{1}{2\pi}$ (where θ is measured in radians). One way of deriving a circular version of an unlimited linear distribution is through "wrapping" it around the circumference of the unit circle. If x is a random variable on the line with probability density function $f(x)$, the wrapped random variable $x_w = [x \bmod 2\pi]$ is distributed according to a wrapped distribution with the probability density function: $f_w(\theta) = \sum_{-\infty}^{\infty} f(\theta + 2\pi k)$.

Applying this derivation to the normal distribution results in a circular version of the normal distribution, but estimating its parameters from sample data can be problematic [GGD53, Mar72].

An easier-to-estimate circular version of the normal distribution was derived by von Mises [GGD53, Mar72], in a way analogous to the way Gauss derived the linear normal distribution — whose maximum-likelihood parameter estimates are the *sample mean and variance*. This circular distribution is the one we are using to model the robot heading in this work, and is described below.

5.3 The von Mises Distribution

A circular random variable, θ, $0 \leq \theta \leq 2\pi$, is said to have the *von Mises distribution* with parameters μ and κ, where $0 \leq \mu \leq 2\pi$ and $\kappa > 0$, if its probability density function is:

$$f_{\mu,\kappa}(\theta) = \frac{1}{2\pi I_0(\kappa)} e^{\kappa \cos(\theta - \mu)} ,$$

where $I_0(\kappa)$ is the modified Bessel function of the first kind and order 0:

$$I_0(\kappa) = \sum_{r=0}^{\infty} \frac{1}{r!^2} (\frac{1}{2}\kappa)^{2r} .$$

Similar to the linear normal distribution, this is a unimodal distribution, symmetrical around μ. The mode is at $\theta = \mu$ while the antimode is at $\theta = \mu + \pi$. We observe that the ratio of the density at the mode to the density at the antimode is $e^{2\kappa}$, which indicates that the larger κ is, the more concentrated the density is about the mode. Figure 5.3 shows an "unwrapped" plot of the von Mises distribution for various values of κ where $\mu = 0$.

We now describe how to estimate the parameters μ and κ given a set of heading samples, angles $\theta_1, \ldots \theta_n$, from a von Mises distribution. We are looking for maximum likelihood estimates for μ and κ. The likelihood function for the data generated by a von Mises distribution with parameters μ and κ can be expressed as:

$$L_{\mu,\kappa} = \prod_{i=1}^{n} f_{\mu,\kappa}(\theta_i) = \frac{e^{\left(\kappa \sum_{i=1}^{n} \cos(\mu - \theta_i)\right)}}{(2\pi)^n I_0(\kappa)^n} .$$

Hence the log likelihood is

$$\log(L_{\mu,\kappa}) = \kappa \sum_{i=1}^{n} \cos(\mu - \theta_i) - \log((2\pi)^n I_0(\kappa)^n) .$$

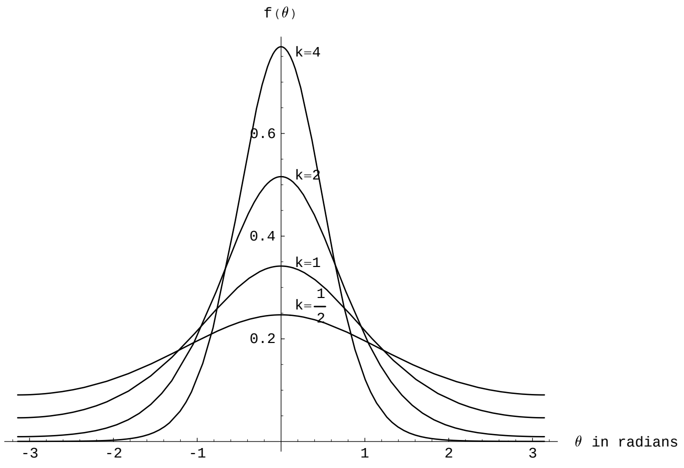

Figure 5.3: The von Mises distribution with mode 0 and various κ values.

To find the maximum likelihood estimate for μ we take $\frac{\partial \log(L_{\mu,\kappa})}{\partial \mu}$, equate it to 0, and obtain the estimate $\overline{\mu}$ for μ:

$$\overline{\mu} = \arctan(\frac{\overline{y}}{\overline{x}}) \ , \tag{5.3}$$

where $\overline{y}$, $\overline{x}$ are as defined in equation 5.1.

The maximum likelihood estimate for κ is obtained by taking $\frac{\partial \log(L_{\mu,\kappa})}{\partial \kappa}$ and equating it to 0. We note that $\frac{dI_0(\kappa)}{d\kappa} = I_1(\kappa)$, where $I_1(\kappa)$ is the modified Bessel function of the first kind and order 1:

$$I_1(\kappa) = \sum_{r=0}^{\infty} \frac{1}{r!(r+1)!}(\frac{1}{2}\kappa)^{2r+1} \ .$$

Hence the maximum likelihood estimate for κ is the $\overline{\kappa}$ that solves the equation:

$$\frac{I_1(\overline{\kappa})}{I_0(\overline{\kappa})} = \frac{1}{n}\sum_{i=1}^{n}\cos(\theta_i - \mu) \ \ . \tag{5.4}$$

If we do not know μ and are only interested in estimating κ with respect to the *estimated* μ, $\overline{\mu}$ (as defined in 5.3), we can use the identity:

$$\frac{1}{n}\sum_{i=1}^{n}\cos(\theta_i - \overline{\mu}) = \frac{1}{n}\sqrt{(\sum_{i=1}^{n}\cos(\theta_i))^2 + (\sum_{i=1}^{n}\sin(\theta_i))^2} \ \ , \tag{5.5}$$

and the definition of $\overline{a}$, as given in Equation 5.2, to deduce that the maximum likelihood estimate for κ is the $\overline{\kappa}$ that satisfies:

$$\frac{I_1(\overline{\kappa})}{I_0(\overline{\kappa})} = \overline{a} \ .$$

However, if we do have a given μ and we want to find a maximum likelihood estimate for the concentration κ of the sample data around that specified μ, the identity 5.5 cannot be used (see also Upton [Upt73]). We need to use as a maximum likelihood estimate for κ, the $\overline{\kappa}$ that satisfies:

$$\frac{I_1(\overline{\kappa})}{I_0(\overline{\kappa})} = \max[\frac{1}{n}\sum_{i=1}^{n}\cos(\theta_i - \mu),\ 0] = \max\left[\frac{1}{n}\sqrt{(\sum_{i=1}^{n}\cos(\theta_i))^2 + (\sum_{i=1}^{n}\sin(\theta_i))^2 - (\sum_{i=1}^{n}\sin(\mu - \theta_i)),\ 0}\right].$$

The above estimation formulae agree with the intuition that the sample is more concentrated ($\overline{\kappa}$ is larger) about the sample mean, $\overline{\mu}$, than about the true distribution mean, μ.

5.4 Handling Angular Odometric Readings

It is now left to explain how we use the von Mises distribution to model the heading readings obtained by the robot as part of its odometric information. Through the rest of this section we explain how the parameters of the von Mises distribution are incorporated into the hidden Markov model and how the learning algorithm described in Chapter 4 is adapted to learn these parameters.

To model the heading difference between each pair of states, the relation matrix R, described in Section 3.2, becomes 3-dimensional rather than 2-dimensional, consisting of the components $\langle x,\ y,\ \theta\rangle$ rather than just $\langle x,\ y\rangle$. The component $R_{i,j}[\theta]$ represents the heading change when moving from state s_i to s_j, and is assumed to consist of the two parameters of the von Mises distribution governing this change. The notation $\mu^{\theta}_{i,j} \overset{\text{def}}{=} \mu(R_{i,j}[\theta])$ represents the mean of the distribution for this heading change, while $\kappa^{\theta}_{i,j} \overset{\text{def}}{=} \kappa(R_{i,j}[\theta])$ represents the concentration parameter around the mean. The three constraints stated in Section 3.2 for the components of R, hold for the θ component as well.

Similarly, every observed relation item, r_t, in the experience sequence E, has a heading-change component, θ, which records the change in heading of the robot between the state at time t, q_t, and the state q_{t+1}.

The reestimation formula for the von Mises mean and concentration parameters of the heading change between states s_i and s_j is the solution to the equations:

$$\overline{\mu}^{\theta}_{i,j} \quad = \arctan\left(\frac{\sum_{t=0}^{T-2}[\sin(r_t[\theta])(\xi_t(i,j)\overline{\kappa}_{i,j} - \xi_t(j,i)\overline{\kappa}_{j,i})]}{\sum_{t=0}^{T-2}[\cos(r_t[\theta])(\xi_t(i,j)\overline{\kappa}_{i,j} + \xi_t(j,i)\overline{\kappa}_{j,i})]}\right)$$

$$\frac{I_1[\overline{\kappa}^{\theta}_{i,j}]}{I_0[\overline{\kappa}^{\theta}_{i,j}]} = \max\left[\frac{\sum_{t=0}^{T-2}[\xi_t(i,j)\cos(r_t[\theta] - \overline{\mu}^{\theta}_{i,j})]}{\sum_{t=0}^{T-2}\xi_t(i,j)},\ 0\right] . \tag{5.6}$$

Note here that the larger $\overline{\kappa}_{ij}$ is, the more concentrated the sample is around the mean, and the more weight we give to the estimated counts $\xi_t(i,j)$ of observing this mean. Also note that the denominator contains a sum rather than a difference, since cos is an even function and $\cos(-r_t[\theta]) = \cos(r_t[\theta])$.

Rather than try to solve these hard mutual equations, we take advantage of generalized EM as we did in Section 4.3.2, and use the "lag-behind" update. We update the mean using the current estimates of the concentration parameters, $\kappa_{i,j}$, $\kappa_{j,i}$, as follows:

$$\overline{\mu}^{\theta}_{i,j} = \arctan\left(\frac{\sum_{t=0}^{T-2}[\sin(r_t[\theta])(\xi_t(i,j)\kappa_{i,j} - \xi_t(j,i)\kappa_{j,i})]}{\sum_{t=0}^{T-2}[\cos(r_t[\theta])(\xi_t(i,j)\kappa_{i,j} + \xi_t(j,i)\kappa_{j,i})]}\right) ,$$

and then calculate the new concentration parameters based on the newly updated mean, as the solution to equation 5.6. Finding $\overline{\kappa}^{\theta}_{i,j}$ that satisfies this equation is done through the use of a lookup table, listing values of the quotient $\frac{I_1[x]}{I_0[x]}$.

The above reestimation formulae agree with the maximum likelihood estimator formulae given in equations (5.3, 5.4). The convergence of the estimation process is guaranteed due to the von Mises being a member of the exponential family [Mar72], and the monotone improvement of the likelihood through their use can be proved along the lines of the proof provided in Section 4.3.3.

Chapter 6

Choosing an Initial Model

It is typical in instances of the Baum-Welch algorithm to simply initialize the model at random, perhaps trying multiple initial models to find different local likelihood maxima. Concretely, to choose an N-state model uniformly at random, we first choose N discrete probability distributions over N events, from the set of all possible N-dimensional probability distributions, uniformly at random[1], in order to construct the transition-distribution matrix A; similarly, we choose N distributions over observations for the observation matrix B; finally we populate the relation matrix with random means, while enforcing consistency constraints.

It is important to note that when continuous distributions are estimated, arbitrary initialization of means and variances can cause numerical instability throughout the algorithm. For instance, if an odometric relation between state i and state j is initially assigned a distribution with an arbitrary mean that is far from any of the actual odometric readings, with only a small variance around it, the density mass assigned to any odometric data according to this distribution is very small, and can exceed the machine's precision, causing an underflow to occur. It is especially severe in a multi-dimensional odometric readings setting, like the one we are dealing with, since when numbers are multiplied by one another, even plausible density values rapidly decrease. Moreover, very small probability values exclude plausible state transitions between states i and j from being considered since they are assigned a density mass which is arbitrarily close to 0.

We may try to choose an initial large variance for the odometric distribution, thus

[1]See Cassandra's Ph.D. thesis [Cas98] for details on choosing distributions at random.

avoiding the underflow problem. Such an approach results in highly inaccurate mean reestimation; if the odometry from state i to state j is initially estimated at a value with high variance, many transitions that do not go from i to j still get a reasonably high density under the distribution from i to j. When the mean is reestimated, all these transitions would participate in the calculation. Thus the model reduces to having arbitrary means that roughly accommodate all the data points due to the large variance around them.

This problem has long been realized by Rabiner et al. [RJL$^+$85], and they make two suggestions for a solution:

- When events have very low density mass, add some small constant to it.
- Start from a good initial model.

The first suggestion is close in practice to the Bayesian approach, with a uniform prior over all events. This approach is adopted in our implementation, as we add a small constant to the estimated transition and occupation probabilities ξ and γ, to avoid 0 probabilities. However, as Rabiner et al. point out, it is not sufficient, since without good estimates for most of the parameters, all the data has low probability which is compensated for by the added constants, resulting in a very flat model. The overall effect is similar to that of having a large variance around arbitrary means.

The second suggestion is not easily met, since we rarely have enough prior knowledge to provide a good initial model, and as stated earlier, our goal is to automate as much as possible the learning process, avoiding the hard work of manually obtaining a good initial model.

We have come up with three initialization strategies. One is to use random initial distributions for the transition and observation matrix, and random relation means from within the range of odometric readings, assigning large initial variance. All three geometrical constraints are enforced on the relation means through dependency propagation. For instance, once we pick the mean μ^x_{ij} at random, the mean μ^x_{ji} is set to $-\mu^x_{ij}$ rather than chosen at random itself. This method still often leads to numerical instability of the algorithm, and results in very flat models, both in the sense of being close to uniform transition and observation distributions and in terms of inaccurate odometric mean estimates with high variance around them.

The two other initialization strategies are biased by the odometric data. One of them was briefly described in earlier publications [SK97b, SK97a], and is based on clustering states according to their odometric locations in a global coordinate system. This method is not robust in the face of cumulative rotational errors, which are discussed in Chapter 8.

We also developed another initialization algorithm, based on clustering odometric *relations* rather than *positions*, and on tagging each odometric relation by its estimated origin and destination states. The algorithm is only concerned with odometric relations *between* states, and therefore is not sensitive to cumulative rotational errors. It also has the potential advantage that it may assist in determining the number of states in the model when all we have is a rough overestimate for this number.

In the following we present both algorithms. They both take as input the sequences E of observations recorded at states and odometric relations recorded *between* states, as well as the number of states N.

6.1 K-Means-Based Initialization

Given a sequence of observations and odometric readings E, we begin by assigning global metric coordinates to each element in the sequence. This is done by accumulating the observed odometric relations between consecutive pairs of odometric readings, as demonstrated in the following example.

Example 6.1 *Suppose we have the following 8 consecutive odometric readings:*

$$\langle 2\ 94\ 92\rangle,\ \langle 1994\ 0\ 88\rangle,\ \langle 3\ \ {-93}\ 86\rangle,\ \langle {-1999}\ \ {-1}\ 94\rangle,\ \langle {-4}\ 102\ 91\rangle,$$
$$\langle 1998\ \ {-5}\ 90\rangle,\ \langle {-2}\ \ {-106}\ 91\rangle,\ \langle {-2003}\ 7\ 87\rangle\ .$$

These can be viewed as the change in heading from one state to the next measured within a global coordinate system.

By accumulating these measures, assuming that the initial position is $\langle 0\ \ 0\ \ 0\rangle$ *we get the following sequence of global position assignments:*

$$\langle 0\ 0\ 0\rangle,\ \langle 2\ 94\ 92\rangle,\ \langle 1996\ 94\ 180\rangle,\ \langle 1999\ 1\ \ {-94}\rangle,\ \langle 0\ 0\ 0\rangle, \qquad (6.1)$$
$$\langle {-4}\ 102\ 91\rangle,\ \langle 1994\ 97\ \ {-179}\rangle,\ \langle 1992\ \ {-9}\ \ {-88}\rangle,\ \langle {-11}\ \ {-2}\ \ {-1}\rangle\ .$$

The angles here are taken to be in the interval $[-180°, 180°]$. The set of global coordinates, as demonstrated above, ignoring any other observation information, is fed into a k-means clustering algorithm (see, for instance, the discussion on simple isodata in Duda and Hart's book [DH73]) , yielding a partition of the data into N clusters.

The k-means algorithm is an iterative procedure that starts by arbitrarily choosing k cluster seeds, where a *seed* is just a point of the same dimension as the data. The number of clusters that we are looking for, k, corresponds in our case to the number of states in the model, N. Each data point is assigned to the cluster whose seed is closest to it. Then the mean of each cluster is calculated from its points, and these means are used as the cluster seeds for the next iteration. The algorithm halts when a fixed-point is reached, and there is no more transition of elements between the clusters.

To use this algorithm, we need to define a procedure for calculating the means as well as a distance metric between elements, in order to find the elements closest to each mean. Since our odometric locations are of the form $\langle x, y, \theta\rangle$ where θ is an angle, we use the mean angle as defined in Section 5.2, and define the mean of a set of odometric locations $\{\langle x_i, y_i, \theta_i\rangle\}$ as:

$$\langle \mu_x, \mu_y, \mu_\theta\rangle \;=\; \left\langle \frac{\sum_i xi}{n}, \frac{\sum_i y_i}{n}, \arctan\left(\frac{\sum_i \sin(\theta_i)}{\sum_i \cos(\theta_i)}\right)\right\rangle \quad .$$

A simple way to overcome the cyclicity of the heading component when calculating the distance between an odometric location $\langle x, y, \theta\rangle$ and a cluster mean $\langle \mu_x, \mu_y, \mu_\theta\rangle$ is to take the Euclidean distance between the vectors $\langle x, y, \sin(\theta), \cos(\theta)\rangle$ and $\langle \mu_x, \mu_y, \sin(\mu_\theta), \cos(\mu_\theta)\rangle$. However, since x, y, μ_x, μ_y are expressed in centimeters, and *sine* and *cosine* are always numbers between -1 and 1, the latter's effect on the distance is negligible, even when the actual heading difference between the mean and the point is very large. To overcome this phenomenon, we choose a constant, C, to scale the *sine* and *cosine* components of the Euclidean distance to the same order of magnitude as the x and y components. In our case, where x and y are measured in centimeters, the scaling constant $C = 200$ proved to be a good choice throughout all our experiments. It is however possible to calculate C from the order of magnitude of the typical changes in x and y rather than provide it as a constant to the program.

Example 6.1 (Cont.)*If the k-means algorithm works well, and the number of states is known to be* 4*, we would expect to obtain the following clustering assignment to the sequence 6.1 (perhaps with a different choice of distinct cluster numbers):*

$$\langle 0,\ 1,\ 2,\ 3,\ 0,\ 1,\ 2,\ 3,\ 0\rangle \ .$$

Once clustering is done, each distinct cluster corresponds to a *state*. The experience sequence can be mapped directly onto a state sequence, such as the one shown in the example above. The state sequence, in turn, is used as though it were the actual state transition sequence through which the experience sequence, E, was generated. We then compute occupation and transition probabilities, (γ and ξ values, respectively), with the γ values being 0 or 1, due to the deterministic nature of the clusters (it might be beneficial to use stochastic clusters, using an algorithm such as Autoclass [CKS+90] as an alternative to k-means). The A, B, and R matrices are all estimated from γ and ξ as described in Section 4.2.2. Finally, an *ad hoc* process is used to adjust R to satisfy the additivity constraint. The main drawback of this algorithm is that in the presence of cumulative rotational error (to be discussed in Chapter 8), the odometric location assignment within a global coordinate system can be highly inaccurate, resulting in clustering together unrelated state positions and in separation of positions that are physically close together.

6.2 Tag-Based Initialization

We have developed an alternative initialization algorithm which assigns states to the recorded observations, based *directly* on the recorded relations between states – rather than on states location within a global coordinate system. This algorithm is much more robust to changes in the coordinate system and can accommodate rotational errors. For the sake of clarity, the description given here still assumes that the relation between states is recorded with respect to a global coordinate system. In Chapter 8 we show how this assumption is relaxed.

Given a sequence of observations and odometric readings E, we begin by clustering the odometric readings into buckets. The number of buckets is bounded from above by the possible number of distinct state transitions that are accounted for in the sequence, which is $\min(N^2, T-1)$, where T is the length of the observation sequence (hence containing $T-1$ odometric records of state transitions). The idea in the bucketing process is that each bucket will contain all the odometric readings that are close to each other along all three dimension.

To achieve this, we start by fixing a small standard deviation value (again, a predetermined constant), along the x, y, and θ dimensions. Denote these standard deviation values $\sigma_x, \sigma_y, \sigma_\theta$ respectively, where typically $\sigma_x = \sigma_y$. The process is initialized by assigning the first odometric reading to bucket 0 and setting the mean of this bucket to be the value of this reading. The rest of the process consists of examining the next odometric reading. If it is within 1.5 standard deviations along each of the three dimensions from the mean of some existing non-empty bucket, add it to the bucket and update the bucket mean accordingly. If not, assign it to an empty bucket and set the mean of this bucket to be this reading. Note

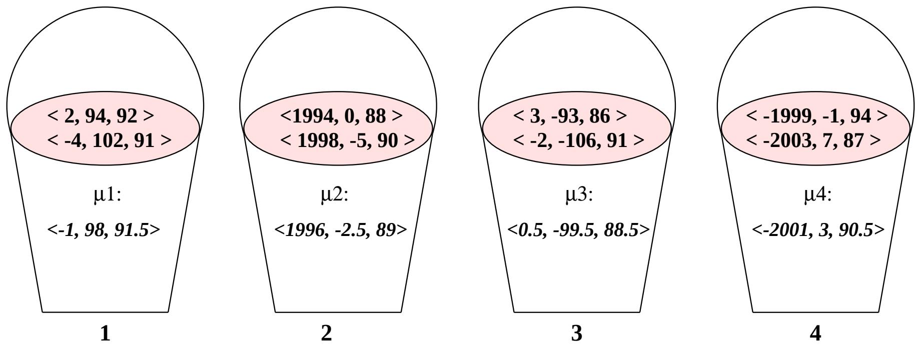

Figure 6.1: The bucket assignment of the example sequence.

that the number of buckets is sufficient to account for all state transitions or elements of the sequence (the largest of the two). Therefore, under the assumption that the relations are indeed normally distributed, with a reasonable choice of standard deviations for populating the buckets, there is always a bucket found for placing each of the readings.

This algorithm guarantees that all the odometric readings in each bucket are within a range of $1.5\langle\sigma_x, \sigma_y, \sigma_\theta\rangle$ from the bucket mean. Since the actual *sample* standard deviation of each bucket is guaranteed to be no larger than the predetermined standard deviation used during the bucketing process, intuitively each bucket is tightly concentrated around its mean. Obviously, this does not guarantee that *all* readings that are within this predetermined range from each other are indeed placed in the same bucket (although this is what we ideally hope to achieve when applying the algorithm.) We note that other clustering algorithms could be used at the bucketing stage. (See, for instance, Duda and Hart's book for a variety of such algorithms.)

Since here each bucket holds only readings close to its mean, it is reasonable to use a standard deviation even for the heading information and treat it in this context as though it were normal, as long as we keep the representation consistent.

Example 6.2 *Suppose we want to learn a 4-state model from a sequence whose odometric component is as given in the previous example:*

$\langle 2\ 94\ 92\rangle$, $\langle 1994\ 0\ 88\rangle$, $\langle 3\ -93\ 86\rangle$, $\langle -1999\ 1\ 94\rangle$, $\langle -4\ 102\ 91\rangle$, $\langle 1998\ -5\ 90\rangle$, $\langle -2\ -106\ 91\rangle$, $\langle -2003\ 7\ 87\rangle$.

As a first stage we place these readings into buckets. Suppose the standard deviation constant is 20*. Then the placement into buckets is as shown in Figure 6.1. The mean value associated with each bucket is shown as well.*

Once bucketing is done, each odometric reading has a bucket to which it has been assigned, and the next phase of the algorithm starts. It consists of *state-tagging* each odometric reading. Each odometric reading, r_t, is assigned a pair of states, s_i, s_j, corresponding to the origin state (from which the transition took place) and the destination state (to which the transition led), respectively. In conjunction with this process the mean entries, μ_{ij}, of the relation matrix R are populated.

Example 6.3 *Returning to the sequence used in example 6.2, the process is demonstrated in Figure 6.2. We assume that the recording starts at state* 0*, and that the odometric change*

A

	0	**1**	**2**	**3**
0	<0,0,0>			
1		<0,0,0>		
2			<0,0,0>	
3				<0,0,0>

S: 0

B

	0	**1**	**2**	**3**
0	<0,0,0>	<-1, 98, 91.5>		
1	< 1, -98, -91.5>	<0,0,0>		
2			<0,0,0>	
3				<0,0,0>

S: 0. 1

Bucket(R[0][1]) = µ1

C

	0	**1**	**2**	**3**
0	<0,0,0>	<-1, 98, 91.5>	<1995, 95.5, -179.5>	
1	< 1, -98, -91.5>	<0,0,0>	<1996, -2.5, 89>	
2	<-1995, -95.5, 179.5>	<-1996, 2.5, -89>	<0,0,0>	
3				<0,0,0>

S: 0, 1, 2

Bucket(R[1][2]) = µ2

D

	0	**1**	**2**	**3**
0	<0,0,0>	<-1, 98, 91.5>	<1995, 95.5, -179.5>	<1995.5, -4, -91>
1	< 1, -98, -91.5>	<0,0,0>	<1996, -2.5, 89>	<1996.5, -102, 177.5>
2	<-1995, -95.5, 179.5>	<-1996, 2.5, -89>	<0,0,0>	< 0.5, -99.5, 88.5>
3	<-1995.5, 4, 91>	<-1996.5, 102, -177.5>	<-0.5, 99.5, -88.5>	<0,0,0>

S: 0,1,2,3

Bucket(R[2][3]) = µ3

S: 0,1,2,3,0

Bucket(R[3][0]) = µ4

,..., S:0, 1, 2, 3, 0, 1, 2, 3, 0

Figure 6.2: Populating the odometric relation matrix and creating a state tagging sequence.

through self transitions is 0*, with some small standard deviation (we use* 20 *here as well).*

This is shown on part A of the figure.

Since the first element in the sequence, $\langle 2\ 94\ 92 \rangle$ is more than two standard deviations away from the mean $\mu[0][0]$ and no other entry in the relation row of state 0 is populated we pick 1 as the next state and populate the mean $\mu[0][1]$ to be the same as the mean of bucket 1, to which $\langle 2\ 94\ 92 \rangle$ belongs. To maintain geometrical consistency the mean $\mu[1][0]$ is set to be $-\mu[0][1]$, as shown in part B of the figure. We now have 2 non-diagonal entries populated in the matrix and the state sequence is $\langle 0, 1 \rangle$. The entry $[0][1]$ in the matrix becomes associated with bucket 1, and this information is kept for helping with tagging future odometric readings which were assigned the same bucket.

The next odometric reading, $\langle 1994\ 0\ 88 \rangle$, is also several standard deviations from any populated mean in row 1 (where 1 is the current believed state) of the relation matrix. Hence, we pick a new state 2, and set the mean $\mu[1][2]$ to be $\mu 2$ — the mean of the bucket 2 — to which the odometric reading belongs. For symmetry preservation, $\mu[2][1]$ is set to be $-\mu[1][2]$. We record that entry $[1][2]$ in the matrix becomes associated with bucket 2. For preserving additivity we also set $\mu[0][2]$ to be the sum of $\mu[0][1]$ and $\mu[1][2]$. $\mu[2][0]$ is set to $-\mu[0][2]$. Similarly, $\mu[2][3]$ is updated to be the mean of bucket 3, causing the setting of $\mu[3][2]$, $\mu[1][3]$, $\mu[0][3]$, $\mu[3][1]$, and $\mu[3][0]$. Bucket 3 is associated with $\mu[2][3]$.

At this stage the odometric table is fully populated, as shown in part D of Figure 6.2. Since the mean calculation is based on accumulation, the standard deviations grow as well, as the square root of the accumulated variances, although this is not shown in the figure.

The state sequence at this point is: $\langle 0, 1, 2, 3 \rangle$. The next reading, $\langle -1999\ \ -1\ 94 \rangle$, is within one standard deviation from $\mu[3][0]$ and therefore the next state is 0. Entry $[3][0]$ is associated with bucket 4, (the bucket to which the reading was assigned), and the state sequence becomes: $\langle 0, 1, 2, 3, 0 \rangle$.

The next reading, being from bucket 1 is associated with the relation from state 0 that is tagged by bucket 1, namely, state 1. By repeating this for the last two readings the final state transition sequence becomes $\langle 0, 1, 2, 3, 0, 1, 2, 3, 0 \rangle$.

Figure 6.3 provides a pseudo-code version of this algorithm. The input to the algorithm is the odometric reading sequence, E_o, of length $T-1$, together with the bucketing information. For $0 \leq i \leq T-1$, $Buckets[i]$ contains the number of the bucket to which the i^{th} odometric reading in the sequence is assigned. The algorithm produces a sequence S of the states believed to have been traversed when the data sequence E was produced, as well as an initial estimate for the mean of each entry in the relation matrix R.

```
POPULATE-AND-TAG($E_0, T, Buckets$)
for $i \leftarrow 0$ to $n$
do
    $\mu[i][i] \leftarrow 0$
    $Dev[i][i] \leftarrow \epsilon$
    for $j \leftarrow 0$ to $n, j \neq i$
    do
        $\mu[i][j] \leftarrow$ $EMPTY$

$S[0] \leftarrow 0$
Current_tag $\leftarrow 0$
for $i \leftarrow 0$ to $T-1$
do  Find $j$ s.t. $\mu[$ Current_tag $][j].Bucket = Buckets[i]$
    if not found
      then  Find $j$ s.t. $\mu[$ Current_tag $][j]$ is within 1.5 standard deviations from $E_o[i]$.
            if not found
              then  Find $j$ s.t. $\mu[$ Current_tag $][j]$ is still undefined
                    if found
                      then $\mu[$ Current_tag $][j] \leftarrow Buckets[i].mean.$
                           $\mu[$ Current_tag $][j].Bucket \leftarrow Buckets[i]$
                           Propagate update to maintain symmetry and additivity.

                      else  Find $j$ s.t. $j \neq$ Current_tag
                            and $\mu[$ Current_tag $][j]$ is closest to $E_o[i]$.



    Current_tag $= j.$
    $S[i+1] = j.$

return $S$
```

Figure 6.3: The state tagging algorithm.

It is possible that by the end of the tagging algorithm, some rows or columns of the relation matrix are still unpopulated. This happens when there is too little data to learn from or when the number of states provided to the algorithm is too large with respect to the actual model. In such cases we can either "trim" the model, using the number of populated rows as the number of states, or pick random odometric readings to populate the rest of the table, improving these estimates later. Note that the first approach suggests a method for *learning the number of states* in the model when this is not given, starting from a gross over-estimate of the number, and truncating it to the number of populated rows in the odometric table after initialization is performed.

Once the state-transition sequence is obtained, the rest of the algorithm is the same as before, deriving state-transition counts from the state-transition sequence, assigning the observations to the states under the assumption that the state sequence is correct, and obtaining the state-transition and observation statistics.

The complexity of this algorithm is worst-case bounded by the complexity of a single Baum-Welch iteration, namely TN^2. To roughly estimate its run time, we observe that the algorithm has to find for each of the $T-1$ odometric readings in the sequence E_o, an assignment of a destination state from within N possible assignment and update the rest of the odometric relationships that are affected by this assignment. If with each assignment we would have had to update N^2 entries, we would have had a complexity of TN^2. Note that the relation matrix can only be fully populated once, hence altogether the complexity for the tagging stage is $TN + N^2$.

The preliminary bucketing stage requires an assignment of a bucket to each of the $T-1$ odometric readings. We take as the maximal number of buckets $\min(N^2, T-1)$, that is, at most every reading has its own bucket. In this case, however, no two readings reflect the same state-transition. The maximal number of possible distinct state transitions over N states is N^2. Therefore, in any case, the complexity of bucketing is bounded by $\min(TN^2, T^2)$.

Thus, the initialization phase does not incur much overhead on the algorithm, and is equivalent to performing a single additional iteration of the Baum-Welch procedure. Judging by the improvement in the results due to the initialization, it is well justified.

Chapter 7

Experiments within a Global Framework

The goal of the work described so far is to use odometry to improve the learning of topological models, while using fewer iterations and less data. We tested our algorithm in a simple robot-navigation world. Our experiments consist of running the algorithm both on data obtained from a simulated model and on data gathered by our mobile robot, Ramona. The amount of data gathered by Ramona is used here as a proof of concept but is not sufficient for statistical analysis. For the latter, we use data obtained from the simulated model.

Significant assumptions underlying all the experiments described in this chapter are that the corridors in the environment are all perpendicular to each other, and that the agent — both in the real robot case and in the simulated case — is aware of this[1]. Hence, after each turn the agent assumes that its new heading is almost perpendicular to its previous heading. This assumption is used when the agent is gathering its data sequence, E. The assumption is satisfied in most office buildings, but is violated in a lot of other environments. We relax the perpendicularity assumption starting in Chapter 8.

7.1 Robot Domain

The robot used in our experiments, Ramona, is a modified RWI B21 robot. It has a cylindrical synchro-drive base, 24 ultrasonic sensors and 24 infrared sensors, situated evenly around its circumference. The infrared sensors are used mostly for short-range obstacle

[1]Thanks to Sebastian Thrun for explicitly expressing this assumption

avoidance. The ultrasonic sensors are longer ranged, and are used for obtaining (noisy) observations of the environment. In the experiments described here, the robot follows a *prescribed* path through the corridors in the office environment of our department.

Low-level software[2] provides a level of abstraction that allows the robot to move through hallways from intersection to intersection and to turn ninety degrees to the left or right. The software uses sonar data to distinguish *doors, openings,* and *intersections* along the path, and to stop the robot's current action whenever such a landmark is detected. Each stop — either due to the natural termination of an action or due to a landmark detection — is considered by the robot to be a "state".

At each stop, ultrasonic data interpretation allows the robot to perceive, in each of the three cardinal directions, (front, left and right), whether there is an open space, a door, a wall, or something unknown.

The robot also has encoders on its wheels that allow it to estimate its pose (position and orientation) with respect to its pose at the previous intersection. After recording both the sonar-based observations and the odometric information, the robot goes on to execute the following prescribed action. The next action command is issued manually by a human being. Of course, both the action and perception routines are subject to error. The path Ramona followed consists of 4 connected corridors in our building, which include 17 states, as shown in Figure 7.1.

In our simulation, we manually generated an HMM representing a prescribed path of the robot through the complete office environment of our department, consisting of 44 states, and the associated transition, observation, and odometric distributions. The transition probabilities reflect action failure rate of about $5-10\%$. That is, the probability of moving from the current state to the correct next state in the environment, under the predetermined action is between 0.85 and 0.95. The probability of self transition is typically between 0.05 and 0.15. Some small probability (typically smaller than 0.02) is sometimes assigned to other transitions. Our experience with the real robot proves that this is a reasonable transition model, since typically the robot moves to the next state correctly, and the only error that occurs with some significant frequency is when it does not move at all, due to sonar interpretation indicating a barrier when there is actually none. Once the action command is repeated the robot usually performs the action correctly, moving to the expected next state. The observation distribution typically assign probabilities of $0.85-0.95$ to the true observation that should be perceived by the robot at each state, and probabilities of

[2]The low-level software was written by James Kurien.

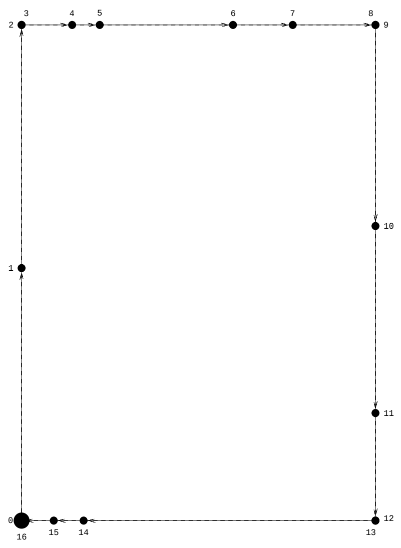

Figure 7.1:True model of the corridors Ramona traversed. Arrows represent the prescribed path direction.

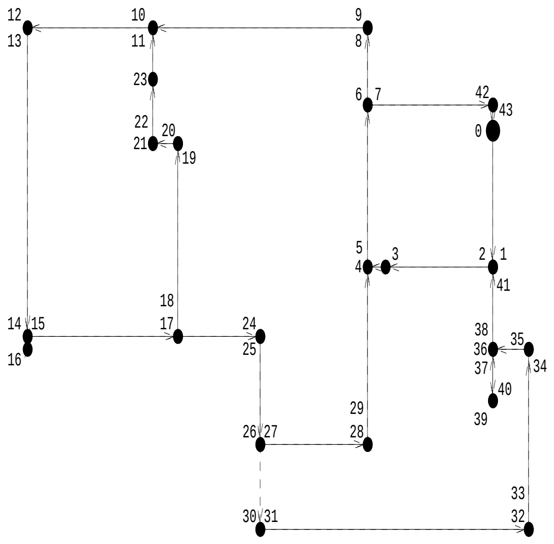

Figure 7.2:True model of a prescribed path through the simulated hallway environment.

0.05 – 0.15 to other observations that might be perceived. For example, if a door should actually be perceived, a door is typically assigned a probability of 0.85 – 0.9, a wall is assigned a probability of 0.09 – 0.1 and an open space is assigned a probability of about 0.01 to be perceived. The standard deviation around odometric readings is about 5% of the mean.

Figure 7.2 shows the HMM corresponding to the simulated hallway environment. Observations and orientation are omitted for clarity. Nodes correspond to states in the environment, while edges correspond to the corridors, drawn according to the direction in which they were traversed. Further interpretation of the figures is provided in the following section.

7.2 Evaluation Method

There are a number of different ways of evaluating the results of a model-learning algorithm. None is completely satisfactory, but they all give some insight into the utility of the results.

In this domain, there are transitions and observations that usually take place, and are therefore more likely than the others. Furthermore, the relational information gives us a rough estimate of the metric locations of the states. To get a *qualitative* sense of the plausibility of a learned model, we can extract an *essential* map from the learned model, consisting of the *states*, the most *likely transitions* and the *metric measures* associated with

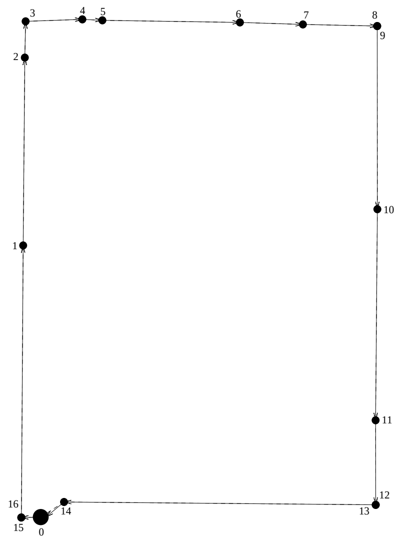

Figure 7.3:Learned model of the corridors Ramona traversed. Tag-based initialization.

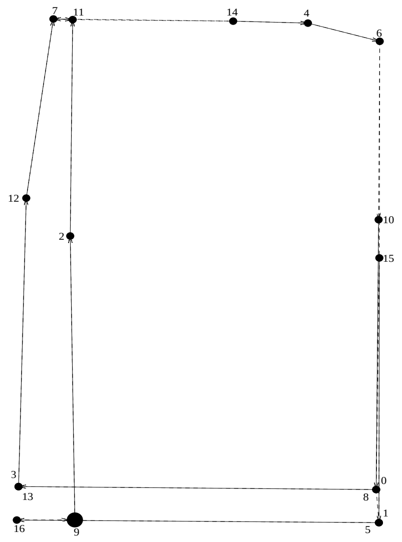

Figure 7.4:Learned model of the corridors Ramona traversed. K-means initialization.

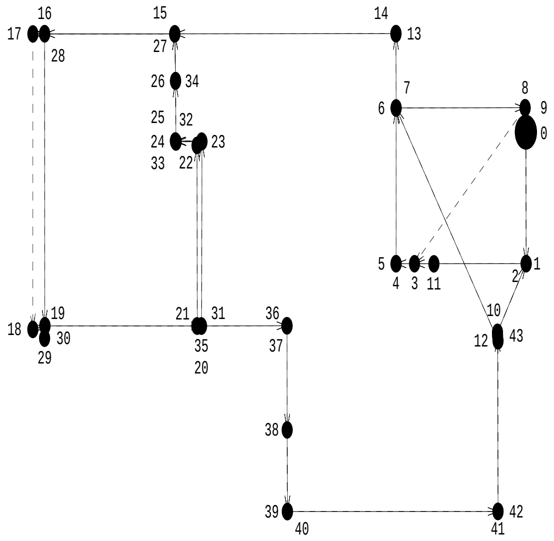

Figure 7.5:Learned model of the simulated hallway environment. Tag-based initialization.

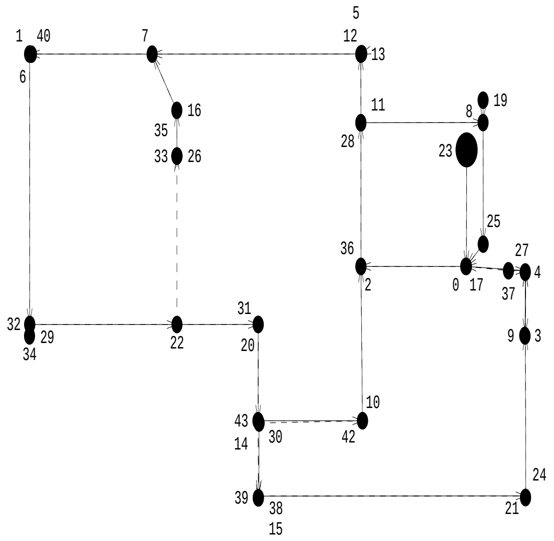

Figure 7.6:Learned model of the simulated hallway environment. K-means initialization.

them, and ask whether this map corresponds to the *essential* map underlying the true world.

Figures 7.1 and 7.2 are such essential versions of the true models, while Figure 7.3, 7.4, 7.5 and 7.6 are essential versions of representative learned ones, (obtained using the two biased initialization methods). Black dots represent the physical locations of states. Multiple states (depicted as numbers in the plot) associated with a single location typically correspond to different orientations of the robot at that location. The larger black circle represents the *initial* state. Solid arrows represent the most likely non-self transitions between the states. Dashed arrows represent the other transitions when their probability is 0.2 or higher. Typically, due to the predetermined path we have taken, the connectivity of the modeled environment is low, and therefore the transitions represented by dashed arrows are *almost* as likely as the most likely ones. Note that the length of the arrows, within each plot, is significant and represents the length of the corridors, drawn to scale.

It is important to note that the figures do not give the complete picture of the models. First, they lack observation distribution and orientation information. Second, in the figures we can only position each state once, and geometrical inconsistencies are not visible. For instance, state 16 in Figure 7.4 is placed according to its geometric relationship to state 5, which places it to the left of the initial state, 9. However, its geometric relationship with respect to state 9 is simply perpendicular, that is, according to our odometric model one merely needs to turn right to reach from state 16 to state 9, which agrees well with the true model. We also omitted states 18 and 41 in figure 7.6 since their relation to the rest of the model was not learned correctly; they are unreachable from all other states and have a uniform transition distribution into all the other states, and therefore have no well defined position in the model. We stress the fact that the figures serve more as a visual aid than as a plot of the true model. We are looking for a good *topological* model rather than a *geometrical* model. The figures provide a geometrical embedding of the topological model. However, even when the geometry, as described by the relation matrix, is different, the topology, as described by the transition and observation matrices, can still be valid.

Traditionally, in simulation experiments, the learned model is *quantitatively* compared to the actual model that generated the data. Each of the models induces a probability distribution on strings of observations; the asymmetric Kullback-Leibler divergence [KL51] between the two distributions is a measure of how good the learned model is with respect to the true model. Given a true probability distribution $P = \{p_1, ..., p_n\}$ and a learned one $Q = \{q_1, ..., q_n\}$, the KL divergence of Q with respect to P is:

$$D(P||Q) \overset{\text{def}}{=} \sum_{i=1}^{n} p_i \log_2 \frac{p_i}{q_i} \quad .$$

We report our results in terms of a sampled version of the KL divergence, as described by Juang and Rabiner [JR85]. It is based on generating sequences of sufficient length (5 sequences of 1000 observations in our case) according to the distribution induced by the true model, and comparing their log-likelihood according to the learned model with the true model log-likelihood. The total difference in log-likelihood is then divided by the total number of observations, accumulated over all the sequences, giving a number that roughly

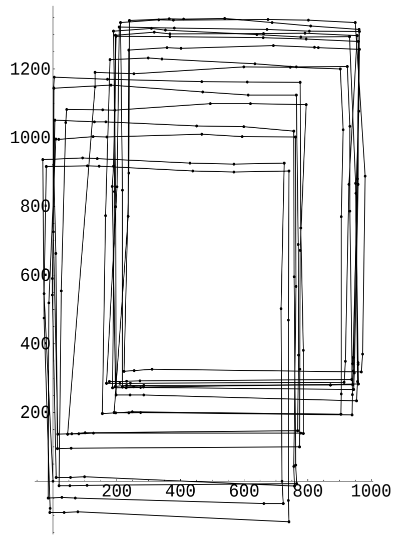

Figure 7.7:A data sequence gathered by Ramona.

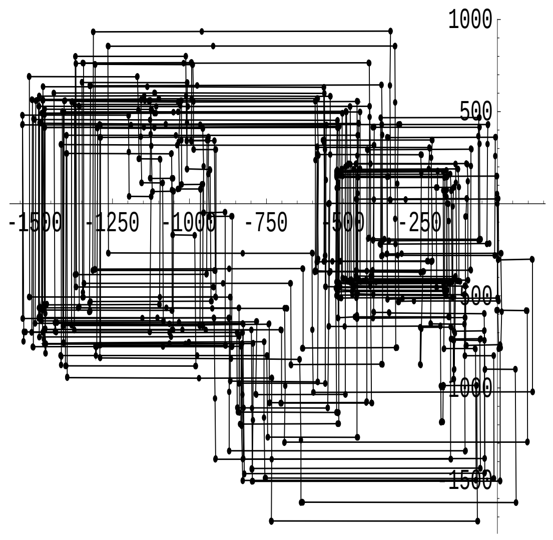

Figure 7.8:A data sequence generated by our simulator.

measures the difference in log-likelihood per observation. Formally stated, let M_1 be the true model and M_2 a learned one. By generating K sequences $S_1, \ldots, S_K$, each of length T, from the true model, M_1, the sampled KL-divergence, D_s is:

$$D_s(M_1 || M_2) = \frac{\sum_{i=1}^{K} [\log(\Pr(S_i|M_1)) - \log(\Pr(S_i|M_2))]}{KT} .$$

We ignore the odometric information when applying the KL measure, thus allowing comparison between purely topological models that are learned with and without odometry.

7.3 Results

We let Ramona go around the path depicted in Figure 7.1 and collect a sequence of about 300 observations. Figure 7.7 plots the sequence of metric coordinates, projected on $\langle x, y \rangle$, obtained by accumulating consecutive odometric readings (as described in Section 6.1). We applied the learning algorithm to the data 30 times. 10 of these runs were started from a k-means-based initial model, 10 started from a tag-based initial model, and 10 started from a random initial model. In addition we also ran the standard Baum-Welch algorithm, ignoring the odometric information, 10 times. (Note that there is non-determinism even when using biased initial models, since the k-means clustering starts from random seeds, and low random noise is added to the data in all algorithms to avoid numerical instabilities, thus multiple runs give multiple results).

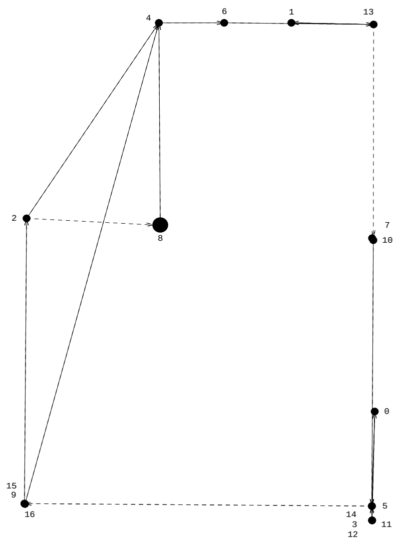

Figure 7.9:Learned model of the corridors Ramona traversed. Random initialization.

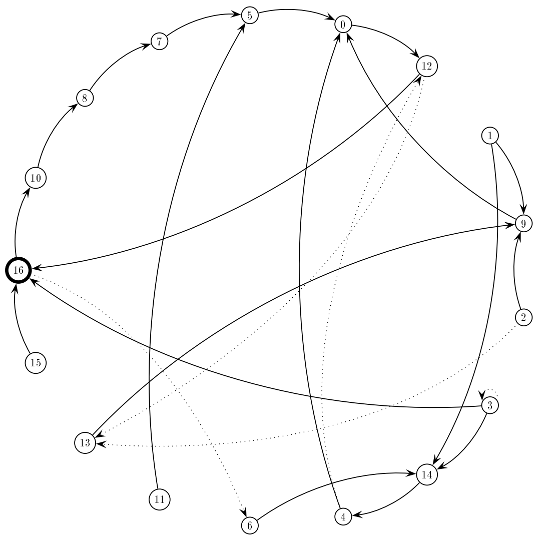

Figure 7.10:The topology of a model learned without the use of odometry.

Figures 7.3 and 7.4 show the essential representations of typical learned models starting from tag-based and k-means-based models, respectively. The geometry of the learned model strongly corresponds to that of the true environment, and most of the states positions were learned correctly. Although the figure does not show it, the learned observation distributions at each state usually match well with the true observations. Starting from a tag-based initialization, the geometry obtained is better in this case than when starting from a k-means-based model. Typically, when the number of states is relatively small, the tag-based initialization performs very well, and outperforms k-means-based initialization. Enforcing geometrical consistency during the tag-based initialization process involves accumulating variances in the relation matrix. When the number of states is large, the variances in the initial relation matrix grow large as well, and the tagging process is not as accurate any more. Figure 7.9 depicts a typical model learned when starting from an arbitrary random model. The geometrical relationships learned under this setting are not as accurate as when using either one of the biased initialization methods.

To demonstrate the effect of odometry on the quality of the learned topological model, we contrast the plotted models learned using odometry with a representative *topological* model learned *without the use of odometric information.* Figure 7.10 shows the topology of a typical model learned without the use of odometric information. In this case, the arcs represent only topological relationships, and their length is not meaningful. The initial state is shown as a bold circle. It is clear that the topology does not match the characteristic ring topology of the true environment.

For obtaining statistically sufficient information, we generated 5 data sequences, each of length 1000, using Monte Carlo sampling from the hidden Markov model whose projection is shown in Figure 7.2. One of these sequences is depicted in Figure 7.8. The figure demonstrates that the noise model used in the simulation is indeed compatible with the noise pattern associated with real robot data.

We used four different settings of the learning algorithm:

- starting from a biased, tag-based, initial model and using odometric information;
- starting from a biased, k-means-based, initial model and using odometric information;
- starting from an initial model picked uniformly at random, while using odometric information;
- starting from a random initial model *without* using odometric information (standard Baum-Welch).

For each sequence and each of the four algorithmic settings we ran the algorithm 10 times. In all the experiments, N was set to be 44, which is the "correct" number of states; for generalization, it will be necessary to use cross-validation or regularization methods to select model complexity. Section 6.2 also suggests one possible heuristic for obtaining an estimate of the number of states.

Figures 7.5 and 7.6 show essential versions of a learned model (obtained from the sequence of Figure 7.8) for a representative run using each of the biased initialization methods. Due to the perpendicularity assumption applied when collecting the data, the K-means algorithm performs well enough in this context. We note that some of the states whose locations overlap in the true model (e.g. 12,13) become separated in the learned model (e.g. states 16,17,28 at the top left corner of Figure 7.5). This is even more noticeable for states 17,18 in the original model which correspond to states 21,31,35, and 20 in Figure 7.5. Similarly, separated states (e.g. 34,35,36,37,38,39,40) that are geometrically close together in the true model are clustered together and overlap in the learned model (e.g. 10,12,43 of Figure 7.5), due to noise in the odometry readings and observations, combined with the limitations of the initialization techniques. However, there is an obvious correspondence between groups of states in the learned and true models, and most of the transitions (as well as the observations, which are not shown) were learned correctly.

Table 7.1 lists the KL divergence between the true and learned model, as well as the number of runs until convergence was reached, for each of the 5 sequences under each

Seq.	**Tag-based**		**k-means**		**Random**		**No Odo**	
#	KL	**Iter.#**	KL	**Iter. #**	KL	**Iter. #**	KL	**Iter. #**
1	1.027	23.7	1.023	30.0	0.954	23.4	6.351	124.1
2	1.006	41.3	0.994	31.4	0.953	25.8	4.863	126.0
3	0.984	16.9	1.029	21.9	1.035	26.6	5.927	113.0
4	1.124	15.8	1.021	25.9	1.017	23.6	6.261	107.4
5	1.029	8.8	1.142	27.2	1.035	32.4	4.802	122.9

Table 7.1: Average results of four learning settings with five training sequences.

of the 4 learning settings, averaged over 10 runs per sequence. We stress that each KL divergence measure is calculated based on *new data sequences* that are generated from the true model, as described in Section 7.2. The 5 sequences from which the models were learned *do not* participate in the testing process. From the table it is clear that the KL divergence with respect to the true model for models learned using odometry, starting from either a biased or a random initial model, is about *5 times smaller* than for models learned without odometry data. The standard deviation around the KL-divergence means was about 1.5 for the no-odometry setting and about 0.1 (often lower) for the odometric settings. To check the significance of our results we used the simple two-sample t-test. The models learned using odometric information have statistically significantly ($p \gg 0.9995$) lower average KL divergence than the others.

The models learned using random initialization seem slightly better in terms of the KL measure than the ones learned with biased initialization and this difference is statistically significant. A close look at the obtained models reveals that the models based on biased initialization are more peaked (that is, have more probabilities very close to 0 and to 1) than the ones based on random initialization[3]. Since the *true model* is slightly less peaked than the ones learned using biased initialization, it occasionally generates, when we execute the KL routine, sequences that are quite unlikely according to its own distribution and *highly* unlikely according to the learned model distribution. This results in a larger KL-divergence for the models learned from biased initialization with respect to the true model. To avoid this difference in peakedness it might be desirable, during the learning stage, to use the Bayesian approach, biasing the learning process towards models that do not have as sharp a distribution over transitions and observations. This will ensure that the models would accommodate even the least likely events in the true environment. It is also possible to post-process learned models, adding a fixed small constant to near-0 probabilities and

[3]The relative flatness of odometric models learned from a random starting point was discussed in Chapter 6.

subtracting a small constant from near-1 probabilities.

The number of iterations required for convergence when learning using odometric information is roughly 1/4 of that required when ignoring such information. Again, the t-test verifies the significance ($p > 0.9995$) of this result. The standard deviation around the mean iteration number is about 10 for models learned from biased initial points (lower than that when the number of iterations is low and higher than that when the number of iterations is high). In the case of random initialization the standard deviation is also typically around 10. When no odometric information is used the standard deviation is about 30. The number of iterations is statistically significantly lower ($p > 0.95$) when using the tag-based initialization than when using any other initialization strategy.

It is important to point out that the number of iterations, although much lower, does not automatically imply that the algorithm works faster or runs in less time. The major bottleneck is the fact that we need to calculate within the forward-backward calculations, as described in Section 4.2.1, the values of the normal and the von-Mises densities. These require the calculation of exponent terms rather than simple multiplications, slowing down the runs considerably, under the current naïve implementation. However, we can solve this by augmenting the program with look-up tables for obtaining the relevant values rather than calculating them. In addition, we can take advantage of the symmetry in the relations table to cut down on the amount of calculation required. It is also possible to use the fact that many odometric relations remain unchanged (particularly in the later iterations of the algorithm) from one iteration to the next, and therefore values can be cached and shared between iterations rather than be recalculated at each iteration.

The initial clustering strongly biases the outcome of learning; it is important to understand whether this bias is useful. When the entire model is initialized at random, the resulting models are usually close, in terms of the KL-divergence, to the true model. This is due to two factors; first, by being typically flatter than the other models, they give reasonable likelihood to any data sequence, and second, by starting from an odometric model that is typically bad with respect to the data, the algorithm ends up not learning much of the geometric setting of the environment. Therefore it can learn topologies that may account for the probabilistic distribution of the data but do not agree with the true topology. This is demonstrated in more detail in Section 11.3.

When starting from a tag-based initial model, the number of iterations is typically lower than when using any other setting. The models obtained when starting either from a

Seq. length	Tag-based		k-means		Random		No Odo	
	Mean KL	Std. Dev.	Mean KL	Std. Dev.	Mean KL	Std. Dev.	Mean KL	Std. Dev.
1000	0.984	0.049	1.029	0.111	1.035	0.063	5.927	1.956
900	1.173	0.351	1.176	0.162	1.027	0.077	7.852	1.446
800	1.108	0.094	1.220	0.102	1.068	0.045	9.624	1.755
700	1.185	0.160	1.329	0.111	1.116	0.104	10.504	2.774
600	1.346	0.249	2.575	1.922	1.250	0.237	14.576	3.498
500	1.205	0.066	2.049	0.717	1.270	0.137	19.649	4.975
400	1.279	0.050	2.558	1.827	1.285	0.214	26.341	4.357
300	1.737	0.428	2.447	0.369	1.599	0.508	33.252	4.444
200	14.047	11.635	2.781	0.406	3.946	2.704	52.780	4.147
100	63.438	1.482	20.606	10.807	12.770	11.987	78.982	6.394

Table 7.2: Average results of three learning settings with 10 incrementally longer sequences.

k-means based or from a tag-based initialization are about equivalent in quality, both topologically and geometrically. However, since the tag-based algorithm is almost deterministic the results obtained by using it are more consistent, and are less prone to change due to varying initial conditions. Typically, when the initialization is good, most of the work is already done and the EM algorithm quickly fills in the details. However, if the initial k-means clustering is bad, it is often close to a poor local maximum and the algorithm is unable to adjust it well. It may be best to run the algorithm multiple times, taking the model with the highest likelihood as the final result.

To examine the influence of the amount of data on the quality of the learned models, we took one of the 5 sequences (Seq. #3) and used its prefixes of length 100 to 1000 (the complete sequence), in increments of 100, as individual sequences. We ran each of the four algorithmic settings over each of the 10 prefix sequences, 10 times repeatedly. We then used the KL-divergence as described above to evaluate each of the resulting models with respect to the true model. For each prefix length we averaged the KL-divergence over the 10 runs.

Table 7.2 summarizes the results of this experiment. It lists the mean KL-divergence over the 10 runs for each of the prefixes, as well as the standard deviation around this mean. The plot in Figure 7.11 depicts the KL-divergence as a function of the sequence length for each of the three settings. Both the table and the plot demonstrate that, in terms of the KL-divergence, our algorithm, which uses odometric information, is robust in the face of data reduction. In contrast, learning without the use of odometry is much more sensitive to reduction in the amount of data.

Again, we applied the two-sample t-test to verify the statistical significance of these

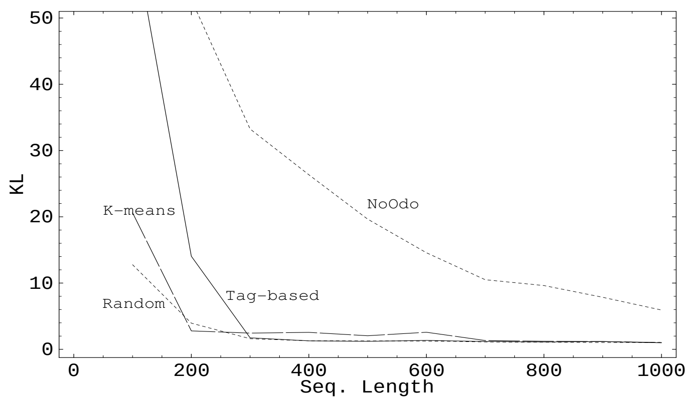

Figure 7.11: Average KL-divergence as a function of the sequence length.

results. For example, the KL-divergence being greater for models learned from a sequence of length 800 than from a sequence of length 1000, *without* the use of odometry, is highly statistically significant ($p > 0.999$). In contrast, the KL-divergence is usually not highly statistically significantly greater when the odometry is used, for either biased or random models. The KL-divergence of the learned biased model fluctuates somewhat since shorter sequences tend to have less accumulated error on their readings, thus clustering may perform better, resulting in better learned models despite the fewer data points available (see for instance the change from 600 to 500 observations).

We note that the data sequence is twice as "wide" when odometry is used than when it is not; that is, there is more information in each element of the sequence when odometry data is recorded. However, the effort of recording this additional odometric information is negligible, and is well rewarded by the fact that fewer observations and less exploration are required for obtaining a data sequence sufficient for adequate learning.

Chapter 8

State-Relative Coordinate Systems

Throughout the discussion so far, we have assumed that there is a single global coordinate system within which the robot operates. Moreover, we assumed that the robot collects its data within a perpendicular corridor framework and that it is taking advantage of this perpendicularity and the single framework while recording odometric information. This assumption can be troublesome in practice. This chapter discusses the potential problems, and presents our way for relaxing the assumption and addressing the problems. A demonstration of the effectiveness of our solution is presented in Chapter 9.

8.1 Motivation

We tend to think about an environment as consisting of landmarks fixed in a global coordinate system and corridors or transitions connecting these landmarks. However, this view of the environment may be problematic when robots are involved.

Conceptually, a robot has two levels in which it operates; the *abstract* level, in which it centers itself through corridors, follows walls and avoids obstacles, and the *physical* level in which motors turn the wheels as the robot moves. In the physical level many inaccuracies can manifest themselves: wheels can be unaligned with each other resulting in a drift to the right or to the left, one motor can be slightly faster than another resulting in similar drifts, an obstacle under one of the wheels can cause the robot to rotate around itself slightly, or uneven floors may cause the robot to slip in a certain direction. In addition, the measuring instrumentation for odometric information may not be accurate in and of itself. At the abstract level, corrective actions are constantly executed to overcome the physical drift and drag. For example, if the left wheel is misaligned and drags the robot leftwards, a corrective

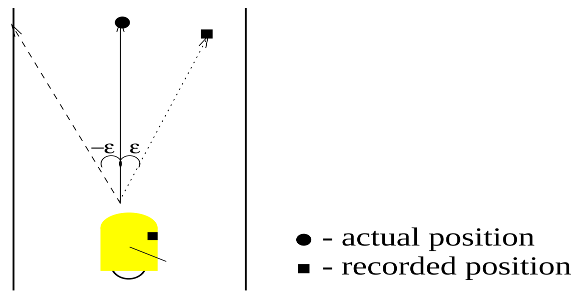

Figure 8.1: A robot moving along the solid arrow, while correcting for drift in the direction of the dashed arrow. The dotted arrow marks its *recorded* change in position.

action of moving to the right is constantly taken in the higher level to keep the robot away from the left wall and centered in the corridor.

The phenomena described above have a significant effect on the odometry recorded by the robot, if it is interpreted with respect to one global framework. For example, consider the robot depicted in Figure 8.1. It drifts to the left $-\epsilon^\circ$ when moving from one state to the next, and corrects for it by moving ϵ° to the right in order to maintain itself centered in the corridor. Let us assume that states are 5 meters apart along the center of the corridor, and that the center of the corridor is aligned with the y axis of the global coordinate system. The robot steps back and forth in the corridor from one state to the next. Whenever the robot reaches a state, its odometry reading changes by $\langle x, y, \theta\rangle$ along the $\langle X, Y,$ heading $\rangle$ dimensions, respectively. As the robot proceeds, the deviation with respect to the x axis becomes more and more severe. Thus, after going through several transitions, the odometric changes recorded between every pair of states, if taken with respect to a global coordinate system, become larger and larger (especially along the X dimension). Similar problems of inconsistent odometric changes recorded between pairs of states can arise along any of the odometric dimensions. It is especially severe when such inconsistencies arise with respect to the heading, since this can lead to mistakenly switching movement along the X and the Y axes, as well as confusion between forwards and backwards movement (when the deviation in the heading is around 90° or 180° respectively). Figure 8.2 demonstrates Ramona's view of a path through the perfectly perpendicular department corridors, depicted in Figure 7.1, based on its odometric recording, with respect to a global coordinate system.

A solution to such a situation is to model the odometric relations of moving from state s_i to state s_j using a changing coordinate system which is *relative* to state s_i, as opposed to a global coordinate system anchored at the initial state. This way, the learned relationship

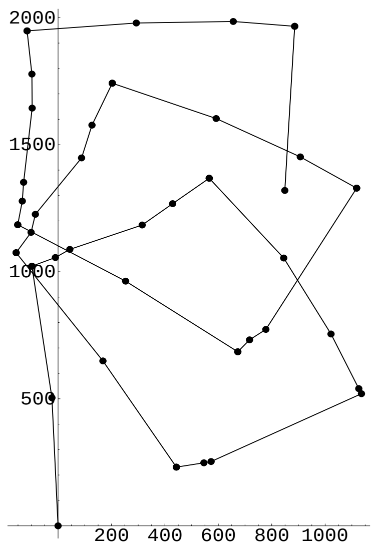

Figure 8.2: A path in a perpendicular environment plotted based on odometric readings taken by Ramona.

between each pair of states, $\langle s_i, s_j \rangle$, is reliable despite the fact that it is based on multiple transitions recorded from s_i to s_j. Within the global coordinate system the relations recorded may vary greatly between one transition from s_i to the next. We formalize this idea and provide the update rules for the odometric information based on this approach in the rest of this chapter.

8.2 Learning Odometric Relations with Relative Coordinates

As before, our experience sequence, E, consists of T pairs $\langle r_t, V_t \rangle$ of recorded odometric relations and observation vectors. The odometric relations are recorded as before, with respect to the robot's global coordinate system. However, when learning the *relation matrix* from the odometric readings, we interpret the entry $R_{i,j}$ in the relation matrix R as encoding the information with respect to a coordinate system whose origin is anchored at the state s_i; the y axis is aligned with the robot's current heading and the x axis is perpendicular to it. This is depicted in figure 8.3. The robot is in state s_i facing in the direction pointed to by the y axis, and its relationship to the state s_j is described in terms of the coordinate system shown in the figure.

To support this interpretation of the relation matrix we need to revisit the formulation of the geometrical-consistency constraints stated in Section 3.2, as well as the initialization procedure and the update formulae used when learning the model.

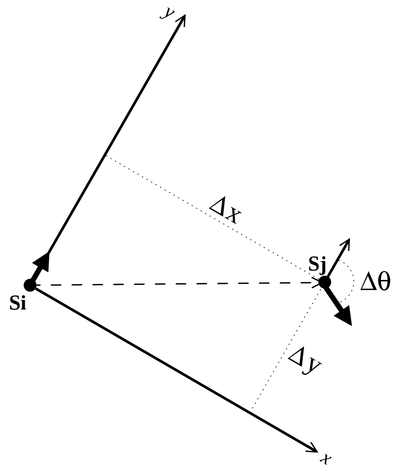

Figure 8.3: The robot is in state S_i facing in the direction of the y axis; the relation between S_i and S_j is measured according to S_i's coordinate framework.

8.2.1 Geometrical Consistency in a Relative Framework

The consistency constraints have to reflect the coordinate system with respect to which the odometry is represented. Note that the change in heading from one state to the next is independent of any specific coordinate system. Hence, only the constraints over the x and y components of the odometric relation need to be redefined.

Given a pair of states a and b, we denote by $\mu^{\langle x,y\rangle}(a,b)$ the vector $\langle \mu(R_{a,b}[x]), \mu(R_{a,b}[y])\rangle$. Let us define $\mathcal{T}_{ab}$ to be the transformation that maps an $\langle x_a, y_a\rangle$ pair represented with respect to the coordinate system of state a, to the same pair represented with respect to the coordinate system of state b, $\langle x_b, y_b\rangle$ (note that $\mathcal{T}_{ab} = \mathcal{T}_{ba}^{-1}$, and $\mu^\theta(a,b) = -\mu^\theta(b,a)$).

More explicitly, as before, let $\mu^\theta(a,b)$ be the mean change in heading from state a to state b. Then the transformation $\mathcal{T}_{ab}$ is defined as follows:

$$\left\langle \begin{array}{c} x_b \\ y_b \end{array} \right\rangle = \mathcal{T}_{ab} \left\langle \begin{array}{c} x_a \\ y_a \end{array} \right\rangle = \left\langle \begin{array}{c} x_a \cos(\mu^\theta(a,b)) - y_a \sin(\mu^\theta(a,b)) \\ x_a \sin(\mu^\theta(a,b)) + y_a \cos(\mu^\theta(a,b)) \end{array} \right\rangle .$$

$\mathcal{T}_{ab}$ can also be expressed using the matrix notation:

$$\mathcal{T}_{ab} = \left[\begin{array}{cc} \cos(\mu^\theta(a,b)) & -\sin(\mu^\theta(a,b)) \\ \sin(\mu^\theta(a,b)) & \cos(\mu^\theta(a,b)) \end{array} \right] .$$

We can now redefine the consistency constraints given in Section 3.2, for the x and y components of the odometric relation:

- $\diamond$ $\mu^{\langle x,y\rangle}(a,a) = \langle 0,0\rangle$;
- $\diamond$ $\mu^{\langle x,y\rangle}(a,b) = -\mathcal{T}_{ba}[\mu^{\langle x,y\rangle}(b,a)]$ *(anti-symmetry)*;
- $\diamond$ $\mu^{\langle x,y\rangle}(a,c) = \mu^{\langle x,y\rangle}(a,b) + \mathcal{T}_{ba}[\mu^{\langle x,y\rangle}(b,c)]$ *(additivity)* .

8.2.2 Initialization

Recall that the tag-based initialization algorithm, described in Section 6.2, maintains geometrical consistency while populating the relation matrix. When the matrix represents relations in a relative coordinate system, the above constraints need to be taken into account when maintaining the consistency of the data. Explicitly, when μ_{ij} is set to $\langle x, y, \theta\rangle$, μ_{ji} is set to

$$\langle -(x\cos(\theta) - y\sin(\theta)),\ -(x\sin(\theta) + y\cos(\theta)),\ -\theta\rangle \ .$$

Similarly, if μ_{ik} is already set to $\langle x_1, y_1, \theta_1\rangle$ and μ_{kj} is being set to $\langle x_2, y_2, \theta_2\rangle$ then μ_{ij} needs to be set to

$$\langle x_1 + (x_2\cos(\theta_1) + y_2\sin(\theta_1)),\ y_1 - (x_2\sin(\theta_1) - y_2\cos(\theta_1)),\ \theta_1 + \theta_2\rangle\rangle \ .$$

8.2.3 Reestimation

The reestimation formulae for all the parameters except for the x and y components of the relation matrix R, remain as before. However, the reestimation formulae for the x and y parameters are changed to reflect the relative coordinate systems used. We follow a similar process to the one used when deriving the reestimation formulae in Section 4.3.3. Again, we are looking to improve expression 4.21 which we repeat here for the sake of clarity:

$$Q(R,\overline{R}) = \sum_{i=0}^{N-1}\sum_{m=1}^{D} Q_{ii}^{m}(R,\overline{R}) + \sum_{i=0}^{N-1}\sum_{j=(i+1)}^{N-1}\sum_{m=1}^{D}[Q_{ij}^{m}(R,\overline{R}) + Q_{ji}^{m}(R,\overline{R})] \ ,$$

only this time the symmetry constraints are:

$$\overline{\mu}_{ji}^{x} = -\cos(\overline{\mu}_{ij}^{\theta})\overline{\mu}_{ij}^{x} + \sin(\overline{\mu}_{ij}^{\theta})\overline{\mu}_{ij}^{y} \ , \tag{8.1}$$

$$\overline{\mu}_{ji}^{y} = -\sin(\overline{\mu}_{ij}^{\theta})\overline{\mu}_{ij}^{x} - \cos(\overline{\mu}_{ij}^{\theta})\overline{\mu}_{ij}^{y} \ . \tag{8.2}$$

By substituting these expressions for $\overline{\mu}_{ji}^{x}$ and $\overline{\mu}_{ji}^{y}$, taking the derivatives of equation 4.21 with respect to $\overline{\mu}_{ij}^{x}$ and $\overline{\mu}_{ij}^{y}$, equating them to 0, and using the "lag-behind" policy with respect to the update of standard deviations, as described in Section 4.2.2, we obtain the following pair of equations:

$$\frac{\mathsf{a}}{\mathsf{v_1}} - \frac{\mathsf{u}\cos(\theta)}{\mathsf{v_2}} - \frac{\mathsf{w}\sin(\theta)}{\mathsf{v_3}} \\ -\overline{\mu}^x_{i,j}\left(\frac{\mathsf{l}}{\mathsf{v_1}} + \frac{\mathsf{k}(\cos(\theta))^2}{\mathsf{v_2}} + \frac{\mathsf{k}(\sin(\theta))^2}{\mathsf{v_3}}\right) + \overline{\mu}^y_{i,j}\mathsf{k}\cos(\theta)\sin(\theta)\left(\frac{1}{\mathsf{v_2}} - \frac{1}{\mathsf{v_3}}\right) = 0\ , \quad (8.3)$$

$$\frac{\mathsf{b}}{\mathsf{v_4}} + \frac{\mathsf{u}\sin(\theta)}{\mathsf{v_2}} - \frac{\mathsf{w}\cos(\theta)}{\mathsf{v_3}} \\ +\overline{\mu}^x_{i,j}\mathsf{k}\cos(\theta)\sin(\theta)\left(\frac{1}{\mathsf{v_2}} - \frac{1}{\mathsf{v_3}}\right) - \overline{\mu}^y_{i,j}\left(\frac{\mathsf{l}}{\mathsf{v_4}} + \frac{\mathsf{k}(\sin(\theta))^2}{\mathsf{v_2}} + \frac{\mathsf{k}(\cos(\theta))^2}{\mathsf{v_3}}\right) = 0\ , \quad (8.4)$$

where

$\mathsf{a} = \sum_t \xi^t_{ij} r^x_t$, $\mathsf{b} = \sum_t \xi^t_{ij} r^y_t$, $\mathsf{u} = \sum_t \xi^t_{ji} r^x_t$, $\mathsf{w} = \sum_t \xi^t_{ji} r^y_t$, $\mathsf{l} = \sum_t \xi^t_{ij}$, $\mathsf{k} = \sum_t \xi^t_{ji}$, $\mathsf{v_1} = (\sigma^x_{ij})^2$, $\mathsf{v_2} = (\sigma^x_{ji})^2$, $\mathsf{v_3} = (\sigma^y_{ji})^2$, $\mathsf{v_4} = (\sigma^y_{ij})^2$, and $\theta = \mu^\theta_{ij}$.

The expressions for $\overline{\mu}^x_{ij}$ and $\overline{\mu}^y_{ij}$ that solve this set of equations constitute the update formulae in the x and y dimensions. As before, these update formulae are guaranteed to improve Q, and are therefore an instance of the generalized EM algorithm.

Note that in earlier work [SK98] we used an update heuristic of assuming that all variances are the same when updating the means, obtaining the following reestimation formulae:

$$\overline{\mu}^x_{i,j} = \frac{\sum_{t=0}^{T-2}(\xi_t(i,j)r_t[x]) - \sum_{t=0}^{T-2}(\xi_t(j,i)\left[\cos(\overline{\mu}^\theta_{i,j}),\quad \sin(\overline{\mu}^\theta_{i,j})\right]\begin{bmatrix} r_t[x] \\ r_t[y]\end{bmatrix}}{\sum_{t=0}^{T-2}[\xi_t(i,j)+\xi_t(j,i)]}\ ;$$

$$\overline{\mu}^y_{i,j} = \frac{\sum_{t=0}^{T-2}(\xi_t(i,j)r_t[y]) - \sum_{t=0}^{T-2}(\xi_t(j,i)\left[-\sin(\overline{\mu}^\theta_{i,j}),\quad \cos(\overline{\mu}^\theta_{i,j})\right]\begin{bmatrix} r_t[x] \\ r_t[y]\end{bmatrix}}{\sum_{t=0}^{T-2}[\xi_t(i,j)+\xi_t(j,i)]}\ .$$

These reestimation rules are easily obtained from the more general ones by setting $\mathsf{v_1}, \mathsf{v_2}, \mathsf{v_3}, \mathsf{v_4}$ to all be the same.

This chapter has introduced an approach for learning models from data that is corrupted by cumulative rotational error. To demonstrate the effectiveness of this approach we ran experiments on data that was collected without applying the perpendicularity assumption, thus indeed suffers from the phenomena described in the beginning of this chapter. The experiments and their results are presented in the next chapter.

Chapter 9

Experiments Using Relative Coordinates

Similar to the experiments presented in Chapter 7, we test our algorithm in a simple robot-navigation world. Again, we use both real robot data and data obtained from the same simulated model as before, as shown in Figure 9.2, with two distinctions:

- The data is generated without the perpendicularity assumption. This means that the x and y coordinates are not realigned after each turn with the global x and y axes, but rather, recorded as is. This is true for both robot data and simulated data.
- The algorithm used for learning the model from the data is as described in Section 8.2.

9.1 Experimental Setting

As before, we provide qualitative results from applying the algorithm to the data obtained from the robot. For statistically evaluating our results we used the sampled Kullback-Leibler divergence of the distribution induced by the learned model with respect to that induced by the true model. The sampled sequences according to which the KL-divergence is calculated are generated and compared as before, ignoring the odometric data.

Figure 9.1 depicts the directed path through which Ramona moved. This is the same true environment as the one described in Chapter 7. Figure 9.3 shows the projection of the odometric readings that Ramona recorded along the x and y dimensions, while traversing this environment. For obtaining statistically sufficient information, we generated 5 data sequences, each of length 800, using Monte Carlo sampling from the hidden Markov model

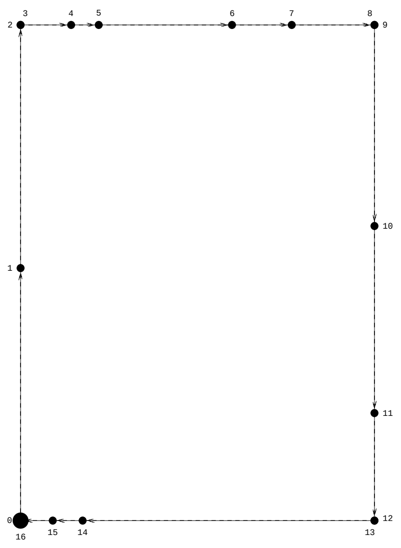

Figure 9.1:True model of the corridors Ramona traversed. Arrows represent the prescribed path direction.

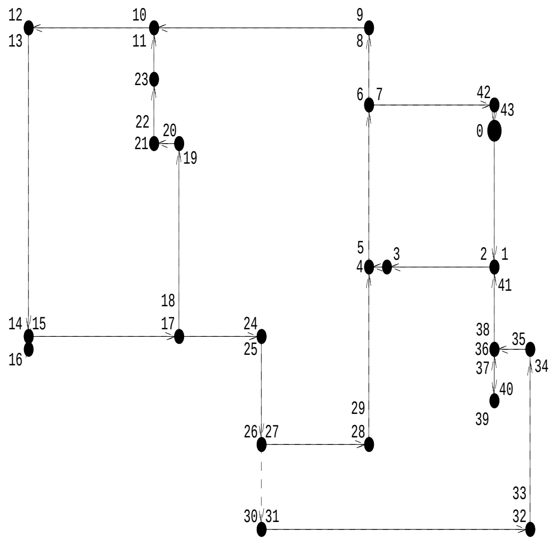

Figure 9.2:True model of a prescribed path through the simulated hallway environment.

whose projection is shown in Figure 9.2. One of these sequences is depicted in Figure 9.4. The figures of both real and simulated data demonstrate that in addition to the noise along the x and y measurements, a cumulative rotational error is present once the perpendicularity assumption is dropped.

We use the same four settings of the learning algorithm as before:

- starting from a biased, tag-based, initial model and using odometric information;
- starting from a biased, k-means based, initial model and using odometric information;
- starting from a random initial model and using odometric information;
- starting from a random initial model without using odometric information (standard Baum-Welch).

For each sequence and each of the four algorithmic settings we ran the algorithm repeatedly 10 times. In all the experiments based on simulated data, N was set to be 44, while in the experiments using real robot data the number of states was set to 17. In both cases this number of states is the "correct" one.

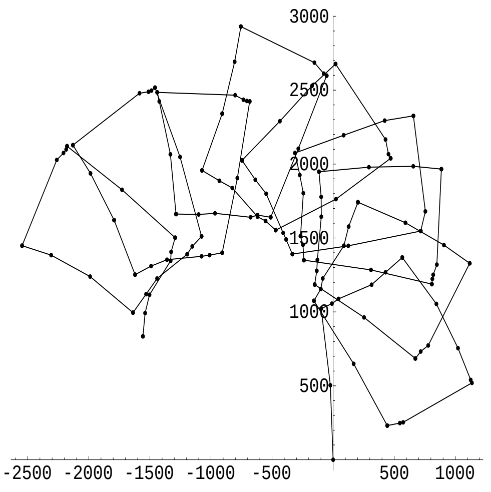

Figure 9.3:A data sequence collected by Ramona in a perpendicular hallway environment.

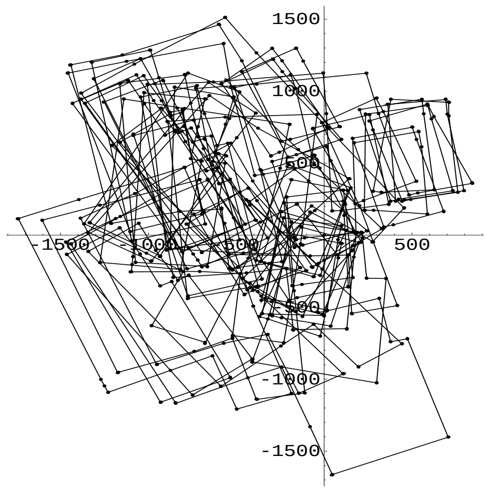

Figure 9.4:A data sequence generated by our simulator, without perpendicularity assumption.

9.2 Results

Figure 9.5 depicts a typical model learned from data obtained by the robot, using odometry, starting from a tag-based initial model. The models learned using k-means initialization and uninformed initialization do not typically reflect the clear rectangular geometry of the true environment, and hence are not satisfactory geometrically. Note that the k-means initialization, as described in Section 6.1, was expected not to perform well in the presence of cumulative rotational error.

As before, the geometrical plot used in Figure 9.5 prevents us from observing the geometrical inconsistencies in the learned model. For example, State 16, when drawn with respect to state 15 is at the same x and y coordinates as state 15, but the heading is perpendicular to it, that is, the robot needs to turn to the right in order to move from state 15 to state 16. However, if we were to draw state 16 with respect to state 1, it should have been placed very close to where state 0 is, thus corresponding much better to the rectangular geometry of the true environment. Since state 1 was drawn with respect to state 0 and state 16 with respect to state 15, while lacking geometrical consistency throughout the model, the geometry is somewhat distorted.

We contrast this model with the one shown in Figure 9.6 which is the topological layout of a model learned without the use of odometric information, from the same data sequence. It is obvious from the figure that the characteristic loop topology of the traversed environment was not learnt when odometric information was not used.

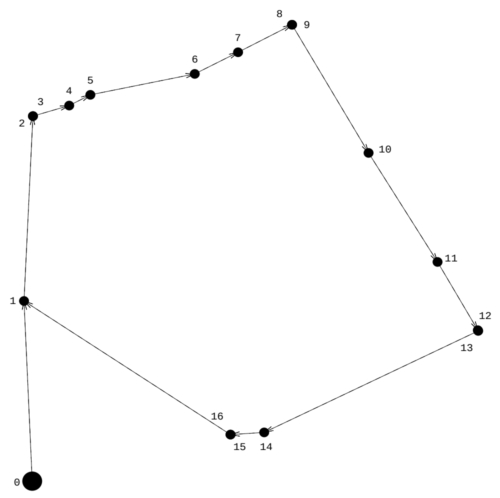

Figure 9.5:Learned model of the corridors Ramona traversed. Initialization is Tag-based.

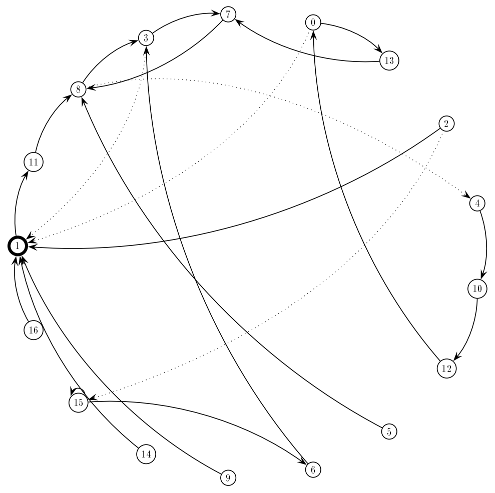

Figure 9.6:Model learned without the use of odometric information, from the same data sequence.

Figures 9.7 and 9.8 depict the geometrical layout of typical models learned from the simulated data, starting from a tag-based initial model. These models are not geometrically good, although some of the rectangular characteristics of the environment are visible. The models learned using k-means initialization and uninformed initialization are, obviously, geometrically worse. The tag-based initialization performs much better when the number of states is small. This can be explained by the fact that when it populates the initial relation matrix and tags odometric readings, it checks the proximity of readings to the table entries, using standard deviations that are accumulated when consistency is enforced. The larger the number of states, the bigger the deviation becomes, and the less accurate the tagging process is. It is an interesting possible research direction to try and learn well small portions of the environment and then combine them into a complete map of the environment.

Table 9.1 lists the KL divergence between the true and learned models, and the number of iterations the algorithm took to converge, for each of the 5 sequences under each of the 4 learning settings, averaged over 10 runs per sequence. The KL divergence with respect to the true model for models learned using odometry, starting from either a biased or a random initial model, is about *8-9 times smaller* than for models learned without odometric data. Note that in these experiments learning was done from training sequences of 800 observations rather than 1000. Therefore the KL measure for the non-odometric case is higher than that given in Table 7.1. The standard deviation around the means is about 0.1 for KL distances learned with odometry, and about 2.5 for models learned without

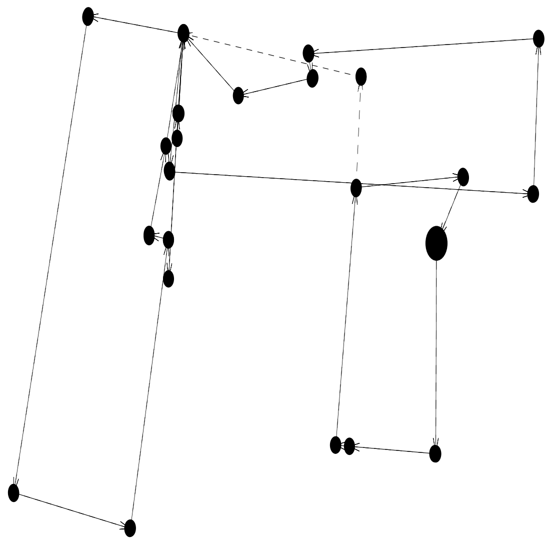
Figure 9.7:Learned model of the simulated environment. Initialization is Tag-based.

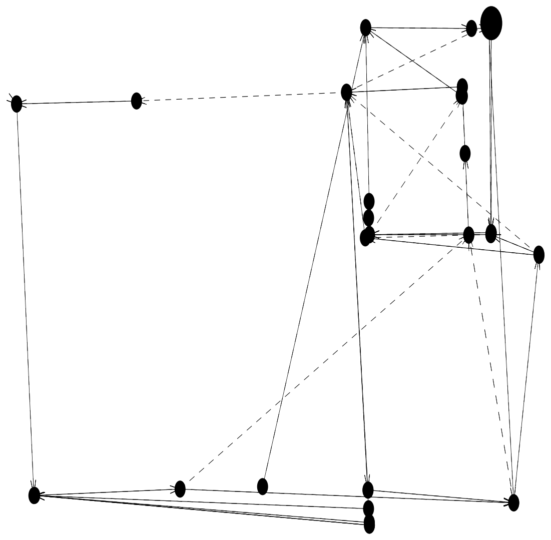
Figure 9.8:Learned model of the simulated environment. Initialization is Tag-based.

Seq. #	**Tag-based**		**k-means**		**Random**		**No Odo**	
	KL	**Iter.#**	KL	**Iter. #**	KL	**Iter. #**	KL	**Iter. #**
1	1.540	9.90	0.983	19.80	1.110	27.00	6.919	113.30
2	1.029	15.10	0.984	24.70	1.065	32.60	9.926	113.10
3	1.301	26.40	1.236	22.70	1.119	27.10	10.030	102.00
4	1.028	33.50	1.100	36.00	1.020	36.00	9.539	104.20
5	1.029	28.30	1.107	25.30	1.060	31.90	12.431	112.50

Table 9.1: Average results of the four learning settings with five training sequences.

odometry. To check the significance of our results we again applied the two-sample t-test. The models learned using odometric information have statistically significantly ($p \gg 0.9995$) lower average KL divergence than the others. We can conclude that despite the cumulative rotational error in the odometric data, the topological models learned using it are much better than those learned without any odometric information.

We notice that models learned with random initialization and the ones based on k-means have a slightly lower KL-divergence than the models whose learning used the tag-based initialization. The explanation to this is again the fact that the tag-based initialization, which is more rigorous than the others, results in more peaked models, typically assigning very low probabilities to the less likely sequences of the original model, even when the learned distributions are similar to the true ones. Thus, the divergence between the learned and the true model is larger when this initialization is used than when the other methods, which result in flatter models, are used.

Seq.	Tag-based	k-means	Random
1	1.280	0.854	1.081
2	0.876	0.904	1.025
3	1.173	1.056	1.058
4	0.837	0.968	0.954
5	0.880	0.949	0.956

Table 9.2: Average results of three learning settings with five training sequences. Distributions are flattened.

To illustrate this point, we applied a simple "flattening" procedure to the models, adding a small constant, (0.001) to all the small probabilities (< 0.0001) and subtracting a proportional portion from the larger probabilities. This procedure was applied to models that were learned using all three initialization procedure, and not just the ones that were learned from tag-based initialization. We compared the resulting less peaked models to the true model, and the results are summarized in Table 9.2. The values of the KL-divergence for the flatter models are indeed smaller than those of Table 9.1. The model learned from sequence 1 is still much worse when using tag-based initialization, since the initial model which is bad in this case, strongly biases the learning algorithm towards a peaked model that is quite different from the true one. Sequence 3 also seems to be associated with worse models when starting from tag-based initialization, although the difference here is not highly statistically significant, due to large standard deviations. For sequences 2, 4, and 5, once the learned models are flattened the KL-divergence of the models learned starting from tag-based initialization is smaller than that of the flattened models learned using the other initialization method. The differences in these cases are highly statistically significant.

The number of iterations required for convergence when learning using odometric information is roughly 3-5 times smaller than that required when ignoring such information. Again, the t-test verifies the significance ($p > 0.9995$) of this result. The typical standard deviation around the mean number of iterations is about 10 when odometry is used and about 35 when odometry is not used. Again, among the methods that use odometric information, the tag-based initialization method results in the smallest iterations number, while the random initialization results in the largest one. This ordering is statistically significant ($p > 0.95$).

To examine the influence of the amount of data on the quality of the learned models, we took one of the 5 sequences (Seq. #1) and used its prefixes of length 100 to 800 (the complete sequence), in increments of 100, as individual sequences. We ran each of the four

Seq. length	Tag-based Mean KL	Tag-based Std. Dev.	k-means Mean KL	k-means Std. Dev.	Random Mean KL	Random Std. Dev.	No Odo Mean KL	No Odo Std. Dev.
800	1.029	0.046	1.107	0.058	1.060	0.105	12.431	2.869
700	1.147	0.044	1.102	0.039	1.129	0.080	15.102	3.578
600	1.361	0.080	1.276	0.171	1.129	0.142	16.832	2.854
500	1.377	0.131	1.267	0.110	1.271	0.118	22.721	4.560
400	1.324	0.102	1.216	0.067	1.267	0.085	28.570	4.755
300	1.475	0.229	1.930	0.698	2.046	1.024	37.111	6.690
200	1.630	0.806	6.493	3.751	3.025	1.776	55.387	3.548
100	16.780	8.572	38.722	7.464	11.396	14.796	85.945	4.054

Table 9.3: Average results of four learning settings with 8 incrementally longer sequences.

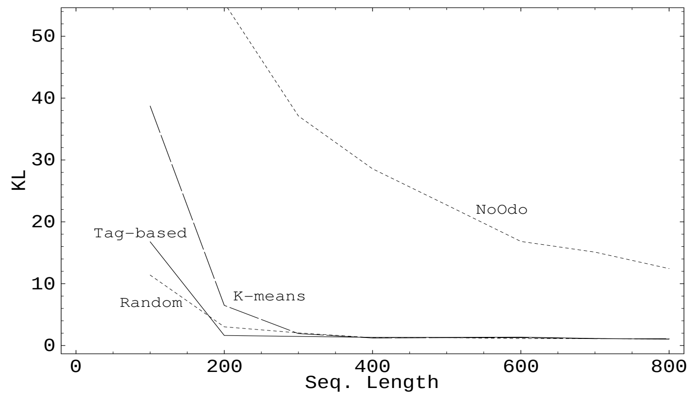

Figure 9.9: Average KL-divergence as a function of the sequence length.

algorithmic settings over each of the 8 prefix sequences, 10 times repeatedly. We then used the KL-divergence as described above to evaluate each of the resulting models with respect to the true model. For each prefix length we averaged the KL-divergence over the 10 runs.

Table 9.3 summarizes the results of this experiment. It lists the mean KL-divergence over the 10 runs for each of the prefixes, as well as the standard deviation around this mean. The plot in Figure 9.9 depicts the KL-divergence as a function of the sequence length for each of the four settings. Both the table and the plot demonstrate the robustness of the algorithm in the face of data reduction. In contrast, learning without the use of odometry is much more sensitive to reduction in the amount of data.

The conclusion from these experiments is that using odometric information, even in the presence of cumulative rotational error can be, when treated correctly, highly beneficial for learning topological models. The results also demonstrate the usefulness of the tag-based algorithm for reducing the number of iterations required for convergence, as well as for obtaining good geometrical models when the number of states is small. However, they also demonstrate a weekness of the initialization algorithm in handling large models. This limitation mostly seem to effect the quality of the geometrical aspects of the learned model. It would be interesting to see if there is a way to address this limitation directly, and also to find ways for learning well small portions of the environment and later combining them into a complete model.

Chapter 10

Enforcing Additivity

In Chapter 3, we augmented the standard HMM with an odometric relation matrix, stating that the relation matrix should satisfy the three geometrical-consistency conditions, for all states a, b, and c:

- ◇ $\mu^m(a,a) = 0$;
- ◇ $\mu^m(a,b) = -\mu^m(b,a)$ *(anti-symmetry)*; and
- ◇ $\mu^m(a,c) = \mu^m(a,b) + \mu^m(b,c)$ *(additivity)* .

In Chapter 8 we have adapted the statement of these conditions to accommodate relative coordinate systems. However, our discussion and experiments up to this point have only dealt with the first two constraints. Although the results are typically topologically satisfactory, it is of interest to know if better results can be obtained by completely satisfying geometrical consistency. Intuitively, there are cases in which enforcing additivity is crucial in order to identify that a state that is reached through two distinct routes is still the very same state. There are several ways we have explored to enforce the full geometrical consistency.

As a first step, we tried to use the iterative procedure described in the previous chapters, augmenting each iteration (which provides a symmetrical but non-additive model), with a procedure for deriving an additive model from the symmetric one. Our first attempt at doing this was through the use of a spring-system model, solving a set of equilibrium equations. The idea is to model each pair of states as though they are connected by a spring, where the length of the spring corresponds to the mean of the odometric relation between the states (as obtained from the symmetric estimation procedure), and the spring constant corresponds to the expected number of times we have transitioned between these two states. By solving the equilibrium equations we obtain the state coordinates that minimize the energy in the system, and due to the geometrical nature of the model, we obtain a geometrically consistent model.

This approach was also taken in other work on geometrical consistency in the context of metric maps. See, for instance, work by Lu and Milios on alignment of laser scans [LM97] or by Golfarelli et al. [GMR98]. However, embedding this approach in the EM setting proved unsatisfactory. At the end of each iteration, solving the equations and changing the relation estimates based on the obtained solution did not preserve monotonic improvement of the likelihood function, and did not guarantee convergence to any kind of solution at any stage.

The approach we take here is that of directly enforcing the additivity constraints through the reestimation procedure. We start by explaining the approach in the case of a global coordinate system for the x and y dimensions only. We then extend the solution to the case of relative coordinate systems. Finally we describe the way in which we maintain geometrical consistency over headings, where the direct approach is problematic due to the special nature of the von Mises distribution.

10.1 Additivity within a Global Framework

The main observation underlying our approach is that the additivity constraint is a result of the fact that states can be embedded in a geometrical space. That is, assuming we have N states, $s_0, \ldots, s_{N-1}$, there are points on some global X and Y and θ axes, $x_0, \ldots, x_{N-1}$, $y_0, \ldots, y_{N-1}$, $\theta_0, \ldots, \theta_{N-1}$ respectively, such that each state s_i is associated with the coordinates $\langle x_i, y_i, \theta_i \rangle$.

Assuming that the use of one global coordinate system is feasible, the mean odometric relation from state s_i to state s_j can be expressed as: $\langle x_j - x_i, y_j - y_i, \theta_j - \theta_i \rangle$. If cumulative rotational error is to be taken into account, the global mean relations $x_j - x_i$, $y_j - y_i$ need to be rotated to be expressed with respect to the heading θ_i at state s_i, as shown in Section 10.2.

When learning the model, rather than look for N^2 *odometric relation values* along the X, Y and θ dimensions that maximize the log-likelihood function with respect to odometric relations while satisfying additivity, we can *reparameterize* the problem. Specifically, we can express each odometric relation as a function of two of the N *state positions*, and optimize an *unconstrained* log-likelihood function with respect to the N *state positions.* Then we can calculate the relations from the state positions, and obtain a geometrically consistent optimal estimate.

Recall that in Sections 4.3.2 and 4.3.3 we introduced the function Q that we need to optimize (or at least improve) in every iteration of reestimating the parameters, as follows:

$$Q(R,\overline{R}) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\sum_{m=1}^{D} Q_{ij}^{m}(R,\overline{R}) \ , \tag{10.1}$$

where

$$Q_{ij}^{m}(R,\overline{R}) = \sum_{t=0}^{T-2} \xi_t(i,j)(\log(\overline{f}_{i,j}^{m}(r_{t+1}^{m})) - \log(\overline{\sigma}_{ij}^{m})) \ ,$$

and $m \in \{x, y, \theta\}$. Due to the independence assumption and the form of the likelihood function $P(\mathsf{E}|\lambda)$ we can separate the optimization procedure into optimizing with respect to each of x, y and θ independently. In this section we discuss the reestimation of the state positions along the x dimension only. For the y dimension the estimation is identical, while θ is treated separately in Section 10.3. We therefore concern ourselves only with optimizing:

$$\begin{aligned} Q^{x}(R,\overline{R}) &\overset{\text{def}}{=} \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} Q_{ij}^{x}(R,\overline{R}) \\ &= \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\sum_{t=0}^{T-2} \xi_t(i,j)(\log(e^{-(r_{t+1}^{x}-\mu_{ij}^{x})^2/2(\sigma_{ij}^{x})^2}) - \log(\overline{\sigma}_{ij}^{x})) \ . \end{aligned}$$

In order to satisfy geometrical consistency of the means μ_{ij}^{x} for all i and j, we need to find N 1-dimensional points, $x_0, \ldots, x_{N-1}$, such that $\mu_{ij}^{x} = x_j - x_i$. These points should maximize (or improve – for generalized EM) the expression:

$$Q^{x}(R,\overline{R}) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\sum_{t=0}^{T-2} \xi_t(i,j)[\log(e^{-(r_{t+1}^{x}-(x_j-x_i))^2/2(\sigma_{ij}^{x})^2}) - \log(\overline{\sigma}_{ij}^{x})] \tag{10.2}$$

with respect to $x_0, \ldots, x_{N-1}$. As in Section 4.2.2 we fix the standard deviation terms to their current values, σ_{ij}^{x}, when reestimating the values $\overline{x_i}$ and $\overline{x_j}$ and then reestimate $\overline{\sigma}_{ij}^{x}$ based on the reestimated $\overline{x_j}$, rather than simultaneously reestimating both terms. This is again an instance of the generalized EM algorithm.

Since all we are interested in is finding the best relationships between x_i and x_j, we can fix one of the x_i's at 0, and only find optimal estimates for the other $N-1$ state positions. Hence, we fix $x_0 = 0$. By taking the derivative of equation 10.2 with respect to each of the other x_j, $(1 \leq j \leq N-1)$ and setting it to 0 we obtain a set of $N-1$ equations of the form:

$$\sum_{\substack{i=0\\i\neq j}}^{N-1} x_i \left(\sum_{t=0}^{T-2}\left(\frac{\xi_t(i,j)}{(\sigma^x_{ij})^2}+\frac{\xi_t(j,i)}{(\sigma^x_{ji})^2}\right)\right) - x_j\left(\sum_{\substack{i=0\\i\neq j}}^{N-1}\sum_{t=0}^{T-2}\left(\frac{\xi_t(i,j)}{(\sigma^x_{ij})^2}+\frac{\xi_t(j,i)}{(\sigma^x_{ji})^2}\right)\right)$$
$$+\sum_{\substack{i=0\\i\neq j}}^{N-1}\sum_{t=0}^{T-2} r^x_t\left(\frac{\xi_t(i,j)}{(\sigma^x_{ij})^2}-\frac{\xi_t(j,i)}{(\sigma^x_{ji})^2}\right)=0 \ . \qquad (10.3)$$

The solution to these equations is an estimate for $x_1,\dots,x_{N-1}$ that maximizes Q^x, where σ^x_{ij} are fixed at their current values. To show that this is indeed a *maximum* and not a minimum, we invoke the following theorems[1]

Definition 10.1 *A matrix $\mathcal{A}=(a_{ij})$ is said to be diagonally dominant if $|a_{ii}|>\sum_{j\neq i}|a_{ij}|$.*

Definition 10.2 *Given a matrix $\mathcal{A}=(a_{ij})$, the matrix $M[\mathcal{A}]=(m_{ij})$ is a matrix whose entries are:*

$$m_{ii}=|a_{ii}|, \quad m_{ij}=-|a_{ij}|,\ for\ j\neq i \ .$$

Theorem 10.1 *[Fie86] Let $\mathcal{A}=(a_{ij})$ be a real square matrix. The following conditions are equivalent:*

- *$\mathcal{A}$ is diagonally dominant.*
- *All the off-diagonal elements of $M[\mathcal{A}]$ are non-positive ($m_{ij}\leq 0$, for $j\neq i$), and every real eigenvalue of $M[\mathcal{A}]$ is* positive*.*

Theorem 10.2 *[Apo69] Let f be a scalar field with continuous second-order partial derivatives $D_{ij}(f)$ in an n-ball $B(a)$, and let $H(a)$ denote the Hessian matrix at a stationary point a. Then if all the eigenvalues of $H(a)$ are negative, f has a relative maximum at a.*

We now examine the Hessian matrix $H(Q^x)$, which is the matrix of the second derivatives of Q^x (after setting x_0 to 0). It is an $(N-1)\times(N-1)$ matrix of the following form:

$$H(Q^x)=\begin{bmatrix}
-\sum\limits_{\substack{i=0\\i\neq 1}}^{N-1}\sum\limits_{t=0}^{T-2}(\frac{\xi_t(i,1)}{(\sigma^x_{i,1})^2}+\frac{\xi_t(1,i)}{(\sigma^x_{1,i})^2}), & \sum\limits_{t=0}^{T-2}(\frac{\xi_t(2,1)}{(\sigma^x_{2,1})^2}+\frac{\xi_t(1,2)}{(\sigma^x_{1,2})^2}, & \dots, & \sum\limits_{t=0}^{T-2}(\frac{\xi_t(N-1,1)}{(\sigma^x_{N-1,1})^2}+\frac{\xi_t(1,N-1)}{(\sigma^x_{1,N-1})^2}) \\
\sum\limits_{t=0}^{T-2}(\frac{\xi_t(2,1)}{(\sigma^x_{2,1})^2}+\frac{\xi_t(1,2)}{(\sigma^x_{1,2})^2}), & -\sum\limits_{\substack{i=0\\i\neq 2}}^{N-1}\sum\limits_{t=0}^{T-2}(\frac{\xi_t(i,2)}{(\sigma^x_{i,2})^2}+\frac{\xi_t(2,i)}{(\sigma^x_{2,i})^2}), & \dots, & \sum\limits_{t=0}^{T-2}(\frac{\xi_t(N-1,2)}{(\sigma^x_{N-1,2})^2}+\frac{\xi_t(2,N-1)}{(\sigma^x_{2,N-1})^2}) \\
\vdots & & & \\
\sum\limits_{t=0}^{T-2}(\frac{\xi_t(N-1,1)}{(\sigma^x_{N-1,1})^2}+\frac{\xi_t(1,N-1)}{(\sigma^x_{1,N-1})^2}), & \dots, & & -\sum\limits_{\substack{i=0\\i\neq(N-1)}}^{N-1}\sum\limits_{t=0}^{T-2}(\frac{\xi_t(i,N-1)}{(\sigma^x_{i,N-1})^2}+\frac{\xi_t(N-1,i)}{(\sigma^x_{N-1,i})^2})
\end{bmatrix} .$$

[1]Thanks to Vasiliki Chatzi for pointing in the direction of diagonally dominant matrices.

Under the assumption that *all ξ_t's are strictly positive*, we note that $H(Q^x)$ is *diagonally dominant*, since all its diagonal elements are strictly larger in magnitude than the sum of the other elements in their respective rows. Hence, by theorem 10.1 all the eigenvalues of the matrix $M[H(Q^x)]$ are positive. Note that $M[H(Q^x)] = -H(Q^x)$, and that from the definition of eigenvalues, k is an eigenvalue of $-H(Q^x)$ if and only if $-k$ is an eigenvalue of $H(Q^x)$. Therefore all the eigenvalues of the Hessian $H(Q^x)$ itself are *negative* and therefore by theorem 10.2, Q^x is indeed maximized when the locations for the states are chosen as solutions to the above equations.

An estimate for each mean relation μ^x_{ij} is simply obtained as

$$\mu^x_{ij} = x_j - x_i \quad , \tag{10.4}$$

and all the geometrical constraints are met. The procedure for y is identical. The process for reestimating the variance remains as described in Section 4.2.2.

10.2 Additivity within a Relative Framework

In Section 8.2 we expressed the geometrical consistency constraints in the context of relative coordinate systems. In this scenario, each state has associated with it a coordinate system whose origin is at the state, its y axis is aligned with the heading associated with the state, and its x axis is perpendicular to it (see, for instance, Figure 8.3).

To enforce the geometrical consistency constraints directly, we observe again that it is sufficient to associate points $\langle x_0, y_0, \theta_0\rangle, \ldots, \langle x_{N-1}, y_{N-1}, \theta_{N-1}\rangle$ with the states $s_0, \ldots, s_{N-1}$, respectively, only that this time the relationships between states along the x and y dimensions are interdependent through the change in headings between the states.

We denote by $(\mu^x_{ij})^0 = x^0_j - x^0_i$ and $(\mu^y_{ij})^0 = y^0_j - y^0_i$ the mean odometric relation from state s_i to state s_j with respect to the *global* coordinate system whose origin is at state s_0, along the *global* x and y axes, respectively. If state s_i has mean heading $\mu^\theta_{0,i}$ with respect to the heading at state s_0, then with respect to state s_i,

$$(\mu^x_{ij})^i = \cos(\mu^\theta_{0,i})(\mu^x_{ij})^0 - \sin(\mu^\theta_{0,i})(\mu^y_{ij})^0 = \cos(\mu^\theta_{0,i})(x^0_j - x^0_i) - \sin(\mu^\theta_{0,i})(y^0_j - y^0_i) \quad .$$

Similarly,

$$(\mu^y_{ij})^i = \sin(\mu^\theta_{0,i})(x^0_j - x^0_i) + \cos(\mu^\theta_{0,i})(y^0_j - y^0_i) \quad .$$

We note that the recorded relations r_t^x, r_t^y are also expressed with respect to the heading at time t. Thus, we are left with the following expression for maximization:

$$\begin{aligned} Q^{x,y}(R,\overline{R}) &= \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\sum_{t=0}^{T-2}\xi_t(i,j)\left(\frac{-(r_{t+1}^x-\mu_{ij}^x)^2}{2(\sigma_{ij}^x)^2}-\log(\sigma_{ij}^x)-\frac{(r_{t+1}^y-\mu_{ij}^y)^2}{2(\sigma_{ij}^y)^2}-\log(\sigma_{ij}^y)\right) \\ &= \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\sum_{t=0}^{T-2}\xi_t(i,j)\left(\frac{-(r_{t+1}^x-(\cos(\mu_{0,i}^\theta)(x_j^0-x_i^0)-\sin(\mu_{0,i}^\theta)(y_j^0-y_i^0)))^2}{2(\sigma_{ij}^x)^2}\right. \\ &\left. -\log(\sigma_{ij}^x)-\frac{(r_{t+1}^y-(\sin(\mu_{0,i}^\theta)(x_j^0-x_i^0)+\cos(\mu_{0,i}^\theta)(y_j^0-y_i^0)))^2}{2(\sigma_{ij}^y)^2}-\log(\sigma_{ij}^y)\right) \ . \end{aligned} \tag{10.5}$$

Differentiating equation 10.5 according to each x_j^0 and each y_j^0 where $j \neq 0$, and equating each partial derivative to 0, results in a set of $2N$–2 linear equations in $2N$–2 unknowns. (The derivatives and the equations are given in Appendix B.2.) The solution, obtained as part of each EM iteration, is a set of coordinates $x_1, \ldots, x_{N-1}$, $y_1, \ldots, y_{N-1}$, for the states $s_1, \ldots, s_{N-1}$, respectively. As before, s_0 is set to be at $\langle 0, 0\rangle$.

The mean state relationships, μ_{ij}^x, μ_{ij}^y are recovered through the equations:

$$(\mu_{ij}^x)^i = \cos(\mu_{0,i}^\theta)(x_j^0-x_i^0)-\sin(\mu_{0,i}^\theta)(y_j^0-y_i^0) \ , \tag{10.6}$$

$$(\mu_{ij}^y)^i = \sin(\mu_{0,i}^\theta)(x_j^0-x_i^0)+\cos(\mu_{0,i}^\theta)(y_j^0-y_i^0) \ . \tag{10.7}$$

Note that the underlying assumption used here is that an estimate for the mean angle $\mu_{0,i}^\theta$ is already calculated. Obtaining geometrically consistent heading angles is discussed in the following section.

10.3 Additive Heading Estimation

The method demonstrated so far suggested that finding optimal *state coordinates* and deducing the relationships between them rather than directly finding optimal relationships is a good way to address the additivity constraint. Unfortunately, this approach is hard to follow in the case of heading change estimation, due to the von Mises distribution assumption of the heading measures.

Recall that the von Mises density function has the form:

$$f_{\mu,\kappa}(\theta) = \frac{1}{2\pi I_0(\kappa)}e^{\kappa\cos(\theta-\mu)} \ ,$$

where $I_0(\kappa)$ is the modified Bessel function of the first kind and order 0, $I_0(\kappa) = \sum_{r=0}^{\infty}\frac{1}{r!^2}(\frac{1}{2}\kappa)^{2r}$. Hence, by substituting μ_{ij} by $\theta_j - \theta_i$ and applying a procedure similar to the one discussed in Section 10.1 we obtain a set of $N-1$ trigonometric equations with terms of the form $\cos(\theta_j)\sin(\theta_i)$ which do not lend themselves to simple solution.

One possible way to address the hardness of solving these equations, suggested by Michael Sever [GSD98], is through the direct optimization of the auxiliary function, Q, using an iterative method such as gradient descent. This is an interesting approach to pursue as future research. However, we preferred to look for a stricter method that is not iterative, given that the EM algorithm itself is already an iterative method. It seems preferable, if possible, to have a (small) fixed bound on the amount of work performed within each EM iteration.

Hence, rather than solve the equations or try to iteratively optimize the auxiliary function, we use the anti-symmetric reestimation procedure presented in Section 5.4, and follow it by a perpendicular *projection* operator, which maps the headings vector of length $(N\text{–}1)^2$, $\langle\mu^{\theta}_{ij}\rangle$, $1 \leq i, j \leq N\text{–}1$, $i \neq j$, which does not satisfy additivity, onto a vector of headings within an additive *linear vector space.* (Note that all entries in which $i = j$ are fixed to 0 already, and therefore we do not need to project them.)

The projection operator $\mathcal{P}$ maps each current mean estimate, μ^{θ}_{0j}, to a real number $\mathcal{P}(\mu^{\theta}_{0j})$. Each μ^{θ}_{ij} where $i \neq 0$ is mapped to $\mathcal{P}(\mu^{\theta}_{0j}) - \mathcal{P}(\mu^{\theta}_{0i})$. An orthogonal projection operator $\mathcal{P}$ is constructed as described by Saad [Saa95], by taking $\mathcal{P} = VV^T$ where V is a matrix whose columns are an orthonormal basis of the *linear* space of vectors that satisfy additivity. Obtaining such an orthonormal basis is done through the Gram-Schmidt procedure.

Our experience shows that this form of projection is still not satisfactory within our setting, since it simply looks for the additive vector closest to the non-additive one, ignoring the fact that some of the entries in the non-additive vector are based on a lot of observations, while others are based on hardly any data at all. Intuitively, we would like to keep the estimates that are well accounted for intact, and adapt the less accounted for estimates in order to meet the additivity constraint. More precisely, we would like to project the non-additive heading estimates vector onto a subspace of the additive vector space, in which the vectors have the same values as the non-additive vector in all entries that are well-accounted for. Unfortunately, this set of vectors is *not* a *linear* vector space (for instance, it does not satisfy closure under scalar multiplication), and the projection operator as defined above can not be applied directly. However, this set of vectors does form an *affine* vector space, and we can project onto it using a special technique from linear algebra. We describe the complete procedure below.

10.3.1 Selecting Fixed Entries

We start by picking the maximum number, n, of heading estimates that we would like to preserve. A typical choice we have made is for n to be the number of states in the model.

Using the heap data structure over all expected counts of transitions between states, we select a list of the n ordered pairs of states (s_i, s_j) that have the largest expected transition counts, $\sum_t(\xi_{ij})$, from s_i to s_j. The list is kept sorted according to the number of the expected counts. Note that we can't just fix these values and proceed to the projection stage, since:

1. There may be inconsistencies between these topmost values themselves;
2. By fixing certain relationships, other relationships are also enforced through the geometrical consistency requirement.

Therefore fixing the entries that should not be projected requires examining and propagating dependencies within the heading relationship vector. To allow for easy expression of the interdependencies, we choose to take $\mu^\theta_{0,1}, \ldots, \mu^\theta_{0,N-1}$ as independent variables, and express the rest of the relationships $\mu^\theta_{i,j}$ as $\mu^\theta_{0,j} - \mu^\theta_{0,i}$.

Building the fixed-values vector proceeds as follows. We pick the most accounted for relationship $\mu^\theta_{i_1,j_1}$, and assume its estimated value is θ_1. Fixing it implies that the vector entry for $\mu^\theta_{j_1,i_1}$ is set to $-\theta_1$. Note that once the entry is fixed it does not change any more. That is, if later in the list of sorted state pairs there is a contradicting assignment to $\mu^\theta_{j_1,i_1}$, we discard this item in the sorted list without using it to fix any of the values, and move down to the next item in the list. However, we do not add any more items to the list; thus we may, in practice, use less than the n most accounted for values.

We also note that fixing $\mu^\theta_{j_1,i_1}$ implicitly forces the relationship between μ^θ_{0,i_1} and μ^θ_{0,j_1} to be $\mu^\theta_{0,i_1} = \mu^\theta_{0,j_1} + \theta_1$.

Once all the implicit relationships are determined, the next most accounted for item on the list is examined. As said before, if it is inconsistent with the already fixed values, it is discarded. Else, it is used for fixing all the implicit dependents. We proceed until all n items in the sorted list are treated.

At the end of this phase we have entries that are completely determined, with a numerical value assigned to them, as well as entries that are inter-related such as μ^θ_{0,i_1} above which depends on μ^θ_{0,j_1} through the equation: $\mu^\theta_{0,i_1} = \mu^\theta_{0,j_1} + \theta_1$. Since, obviously, μ^θ_{0,i_1} and μ^θ_{0,j_1} depend on one another, we adopt the convention that the item with the higher index depends on the the item with the lower index, and the item with the lower index is viewed as a free variable. The following example demonstrates the process.

Example 10.1 *Suppose our model has 3 states. Hence our estimated symmetric vector of heading relationships, excluding self-transitions values which are always* 0*, has* 6 *entries.*

$$\langle \mu^\theta_{0,1}, \mu^\theta_{0,2}, \mu^\theta_{1,0}, \mu^\theta_{1,2}, \mu^\theta_{2,0}, \mu^\theta_{2,1} \rangle \;=\; \langle 90, 110, -90, 5, -110, -5 \rangle \;.$$

We note that $\mu^\theta_{0,1} + \mu^\theta_{1,2}$ *is currently not equal to* $\mu^\theta_{0,2}$*. Now, suppose that our expected supporting counts of transitions between every pair of states are as follows:*

	0	*1*	*2*
0	*0.01*	*0.01*	*0.01*
1	*0.02*	*0.01*	*3.97*
2	*0.01*	*2.94*	*0.03*

Suppose we are only going to preserve the relationships for the 2 *most accounted for state transitions. Hence our sorted list contains is* $\mu^\theta_{1,2}, \mu^\theta_{2,1}$*. By fixing* $\mu^\theta_{1,2}$ *to* 5 *we fix* $\mu^\theta_{2,1}$ *to* -5*, and* $\mu^\theta_{0,2} = \mu^\theta_{0,1} + 5$*. (Through symmetry we have* $\mu^\theta_{1,0} = -\mu^\theta_{0,1}$ *and* $\mu^\theta_{2,0} = -\mu^\theta_{0,1} - 5$*.) The second most-supported value is* $\mu^\theta_{2,1}$*, but since it was already fixed by the previous step — we are done. The geometrically consistent partially-fixed vector is now:*

$$\langle \mu^\theta_{0,1},\ \mu^\theta_{0,1} + 5,\ -\mu^\theta_{0,1},\ 5,\ -\mu^\theta_{0,1} - 5,\ -5 \rangle \;.$$

The final step left is to project the inconsistent values currently assigned to the unfixed entries $\langle 90, 110, -90, -110 \rangle$ *onto a space of the form* $\langle x, x+5, -x, -x-5 \rangle$*.*

We emphasize again, that vectors of the form $\langle x, x+5, -x, -x-5 \rangle$ *do not* constitute a *linear* vector space. In contrast, vectors of the form $\langle x, x, -x, -x \rangle$ do constitute a *linear* space. An affine transformation maps between these two spaces; adding/subtracting the vector $\langle 0, 5, -0, -5 \rangle$ is the appropriate transformation. Hence, vectors of the form $\langle x, x+5, -x, -x-5 \rangle$ constitute an *affine* space of vectors. The following section explains how to project onto such a space.

10.3.2 Projection onto an Affine Space [2]

Perpendicular projection onto a *linear* space was described earlier in this chapter. An *affine* space A is defined as follows:

[2]This section is almost completely based on material I learned through conversations with John Hughes, to whom I am most grateful.

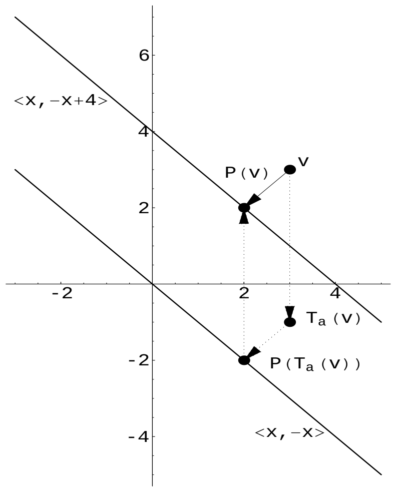

Figure 10.1: Projecting v onto an affine space

Definition 10.3 $\mathsf{A} \subseteq \mathcal{R}^n$ *is an n-dimensional* affine *space if for all vectors* $v_a \in \mathsf{A}$*, the set of vectors:* $\mathsf{A} - v_a \overset{def}{=} \{u_a - v_a | u_a \in \mathsf{A}\}$ *is a* linear *space.*

Hence, we can pick a vector $v_{a_1} \in \mathsf{A}$ and define the translation $T_a : \mathsf{A} \to V$, where V is a linear space, $V = \mathsf{A} - v_{a_1}$. This translation is trivially extended for any vector $v' \in \mathcal{R}^n$, by defining $T_a(v') = v' - v_{a_1}$. In order to project a vector $v \in R^n$ onto A, we apply the translation T_a to v and project $T_a(v)$ onto V, which results in a vector $\mathcal{P}(T_a(v))$ in V. By applying the inverse transform T_a^{-1} to it, we obtain the projection of v on A.

This process is demonstrated in Figure 10.1, in which the linear space is the two dimensional vector space $\{\langle x, y\rangle | \; y = -x\}$, and the affine space is $\{\langle x, y\rangle | \; y = -x + 4\}$. The transform T_a in this case consists of subtracting the vector $\langle 0, 4\rangle$. The solid arrow in the figure corresponds to the direct projection of the vector v onto the point $\mathcal{P}(v)$ of the affine space. The dashed arrows represent the projection via translation of v to $T_a(v)$, the projection of the latter onto the linear vector space, and the inverse translation of the result, $\mathcal{P}(T_a(v))$, onto the affine space.

By applying this projection process, we obtain the values that replace the entries in the heading relations vector that were not yet fixed by the previous stage of the algorithm, and did not satisfy additivity. We plug these values into their correct entries in the vector and obtain a heading vector that satisfies additivity. We note that it is sufficient to project only the entries corresponding to undefined values of $\mu_{0,1}, \ldots, \mu_{0,N-1}$; the rest of the values can be deduced through the expression: $\mu_{i,j} = \mu_{0,j} - \mu_{0,i}$. We conclude this section by completing the example provided earlier.

Example 10.1 (Cont.) *We needed to project the values* $\langle 90, 110, -90, -110\rangle$ *onto a space of the form* $\langle x, x+5, -x, -x-5\rangle$. *It is sufficient to project the entries* $\langle 90, 110\rangle$ *onto the space* $\langle x, x+5\rangle$, *and the rest of the values are obtained through anti-symmetry. The linear space* $\langle x, x\rangle$ *is obtained from the affine space* $\langle x, x+5\rangle$ *through the transform of subtracting the vector* $\langle 0, 5\rangle$. *Applying this transform to the vector* $\langle 90, 110\rangle$ *results in* $\langle 85, 105\rangle$. *The space* $\langle x, x\rangle$ *is spanned by the orthonormal basis* $b = \langle\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\rangle$. *The projection obtained by taking* $b^T b$ *applied to the vector* $\langle 85, 105\rangle$ *results in the vector* $\langle 95, 95\rangle$. *We apply the inverse translation, adding* $\langle 0, 5\rangle$ *and obtain the vector* $\langle 95, 100\rangle$.

The complete vector obtained by plugging $\langle 95, 100, -95, -100\rangle$ *into the yet undetermined entries of the heading relations vector gives us the following additive heading reestimate:*

$$\langle 95, 100, -95, 5, -100, -5\rangle \ .$$

Although the procedure for preserving additivity over headings is not proven to preserve monotone convergence of the likelihood function towards a local maximum, our experiments have shown that monotone convergence is preserved. However, under the current form of additivity enforcement, the quality of the results, as demonstrated in the next chapter, compared with those obtained when only symmetry is enforced, does not clearly justify the additional effort involved.

Chapter 11

Experiments Enforcing Additivity

Similar to the experiments presented in previous chapters, we test the algorithm both within a global framework under the perpendicularity assumption, and within a relative coordinate framework, where the perpendicularity assumption is relaxed. Again, we use both real robot data and data obtained from the same simulated model as before.

As before, we provide both qualitative results in terms of plots of the models, and statistical evaluation. Section 11.1 demonstrates the effects of additivity enforcement under the perpendicularity assumption, while Section 11.2 demonstrates its effects when the perpendicularity assumption is relaxed. Note that there is a significant difference between the two settings. The reason is that when the perpendicularity assumption is dropped, obtaining the correct headings is crucial, since the evaluation of both x and y measurements depend on the heading. Through the use of projection over the heading we compromise the quality of the heading, and possibly harm x and y estimates. Another problem with the additivity enforcement is its reliance on μ_{0i} as a basis for the constraints. A bad estimate for the μ_{0i}'s can result in a bad estimation of all the geometrical parameters.

Section 11.3 describes some experiments designed specifically for studying the effects of odometry and additivity on the quality of the topology and the geometry of learnt models.

11.1 Results within a Global Framework

We applied the algorithm described in Section 10.1 to the same robot-gathered and simulated sequences described and used in Chapter 7. The same evaluation methods are applied here as well.

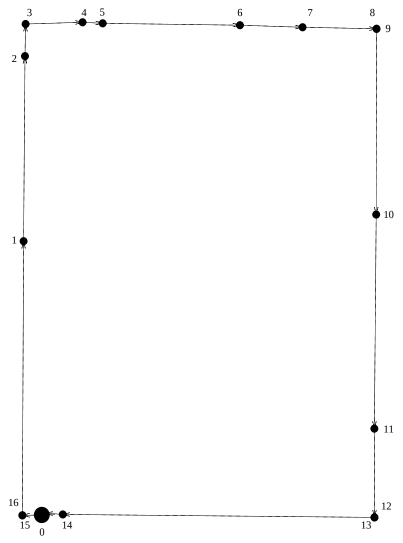

Figure 11.1:Learned model of the corridors Ramona traversed. Initialization is Tag-based.

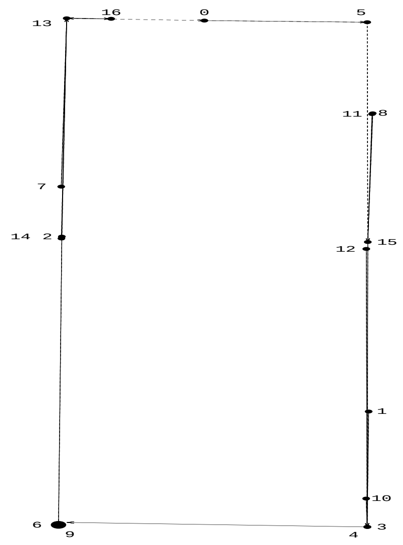

Figure 11.2:Learned model of the corridors Ramona traversed. Initialization is k-means based.

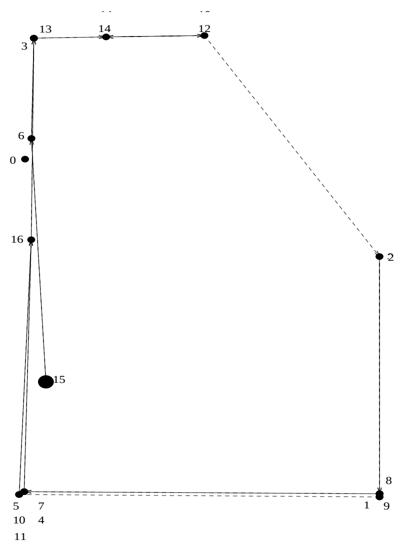

Figure 11.3:Learned model of the corridors Ramona traversed. Initialization is k-means based.

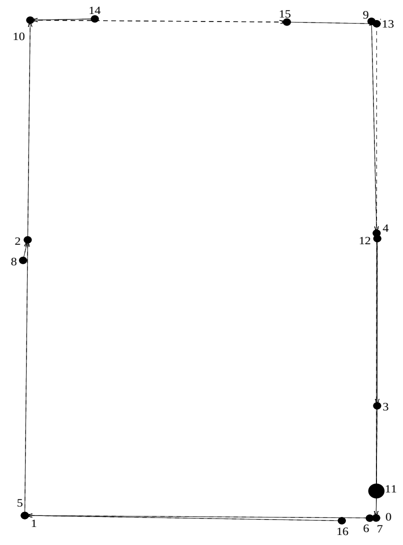

Figure 11.4:Learned model of the corridors Ramona traversed. Initialization is random.

Figures 11.1 – 11.4 show some of the models obtained by applying our algorithm enforcing the complete geometrical consistency while using the various initialization methods. Bold dashed arrows denote transitions whose probability is very close to the probability of the most likely transition from the state. The most accurate map is learned using the tag-based initialization technique. There is very little fluctuation in the quality of the models across multiple runs since the initialization is close to deterministic (up to the random noise superimposed on the data). However, this model is not very different from the one depicted in Figure 7.3 which was learned without enforcing additivity, using the same initialization algorithm.

The two examples of the learned geometry starting from k-means initialization demonstrate that geometrically consistent models can still vary in quality. Moreover, although the geometry looks almost accurate in Figure 11.2, note that the topology is still not as good, given that we don't have two states at each of the top two corners to denote change of headings during turns, and that there is a back and forth transition between states 13 and 16. The variability in the geometrical outline of the models across multiple runs, when starting with k-means initialization is smaller when additivity is enforced, than when only anti-symmetry is enforced. That is, we get more consistently good models and the dependence on the initial clustering seems to be reduced.

It is interesting to note that even when starting from a random initial model, the algorithm did converge several times to models with geometry that is close to the true one, as shown in Figure 11.4. Again, this behavior appears to occur more consistently than when only anti-symmetry is enforced.

Figures 11.5 – 11.8 show some of the models obtained by applying our algorithm to the simulated sequences, enforcing the complete geometrical consistency while using the various initialization methods. The most accurate map, shown in Figure 11.5 is learned using the k-means based initialization technique. Using a global framework allows this method to be effective. There is still, however, a lot of variability across multiple runs and training sequences, in the quality of the results under this initialization method, due to its random starting point. A much worse model, geometrically speaking, which does not represent the true environment well, despite its geometrical consistency, is shown in Figure 11.6. When using the tag-based initialization, we get more uniformity in the quality of the results across multiple runs, but there is still diversity when running over different data sequences. We show one of the better geometrical models in Figure 11.7, and not as good a model in Figure 11.8. A possible explanation to the diversity in quality of the learned simulated model, under tag-based initialization, as opposed to the models learned from robot data, is that the simulated environment is much larger. This causes the global relations between remote states, which are reflected in the geometrical consistency constraints, to be harder to learn.

For the purpose of quantitatively evaluating the learning algorithm we provide in Table 11.1 a summary of the results of running the algorithm under each of the 3 initialization settings, 10 times for each sequence. The results of the runs without odometric information

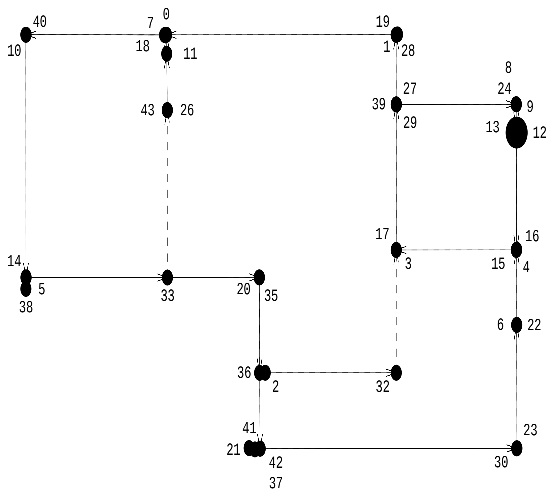

Figure 11.5:Learned model of the simulated environment. Initialization is k-means based.

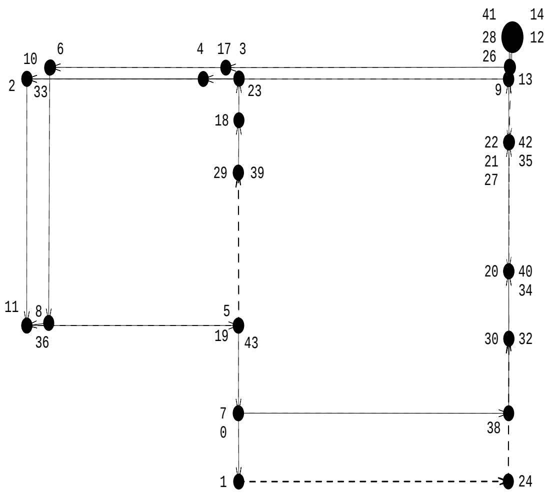

Figure 11.6:Learned model of the simulated environment. Initialization is k-means based.

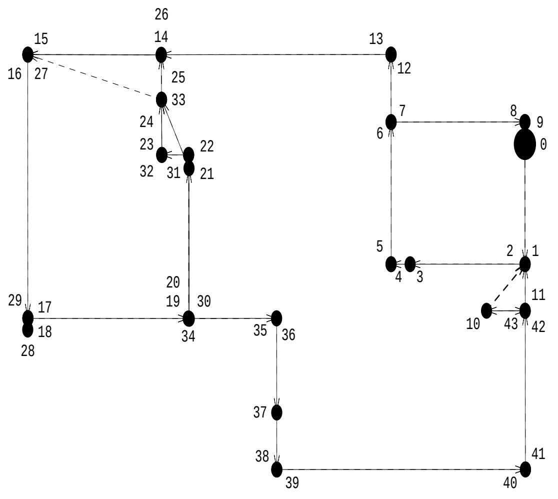

Figure 11.7:Learned model of the simulated environment. Initialization is tag-based.

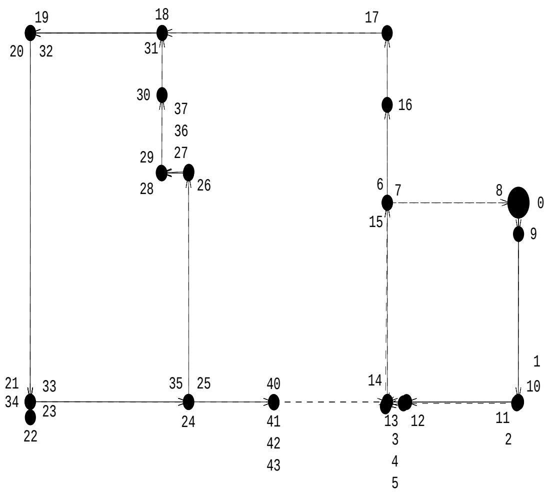

Figure 11.8:Learned model of the simulated environment. Initialization is tag-based.

are listed here again for comparison. The KL divergence with respect to the true model for models learned using odometry, starting from either a biased or a random initial model, is about *5-6 times smaller* than for models learned without odometric data. The standard deviation around the means is about 0.2 for KL distances for models learned with odometry using biased initialization and about 0.1 for models learned from a random starting point. The two-sample t-test still verifies that the odometric models are better than the non-odometric ones with a very high statistical significance.

In addition, the number of iterations required for convergence when learning using odometric information is roughly 4-5 times smaller than that required when ignoring such

Seq. #	**Tag-based**		**k-means**		**Random**		**No Odo**	
	KL	**Iter.#**	KL	**Iter. #**	KL	**Iter. #**	KL	**Iter. #**
1	0.981	16.70	1.149	29.90	1.061	28.40	6.351	124.1
2	1.290	20.90	1.037	27.80	1.037	27.70	4.863	126.0
3	1.115	22.30	1.085	20.40	1.110	21.50	5.926	113.0
4	1.241	12.70	1.144	26.60	1.055	19.90	6.261	107.4
5	1.241	27.50	1.442	20.40	1.028	29.20	4.802	122.9

Table 11.1: Average results of four learning settings with five training sequences.

information. Again, the t-test verifies the significance of this result. As before, the number of iterations required for convergence when starting from a tag-based initial model is statistically significantly lower than when starting with any other initialization method.

Under all three initialization settings, the models learned are topologically somewhat inferior (and this is with high statistical significance), in terms of the KL divergence, to those learned without enforcing additivity. This is likely to be a result of the very strong constraints enforced during the learning process, which prevent the algorithm from searching better areas of the learning-space, and restrict it to reach poor local maxima. The geometry looks superior in some cases, but it is not significantly better. However, there seems to be less variability in the quality of the geometrical models across multiple runs when additivity is enforced.

11.2 Results within a Relative Framework

We applied the algorithm described in Section 10.2 to the same robot-gathered and simulated sequences described and used in Chapter 9. The evaluation methods also stay the same. Figure 11.9 shows a typical model obtained by applying the algorithm enforcing the complete geometrical consistency, to the robot data shown in Figure 9.3, using tag-based initialization. The enforcement of consistency constraints resulted in a better preservation of the rectangular geometry of the environment. We notice that state 0 still does not participate in the loop. The main reason for this is that due to the relatively large number of states that are close together in the corresponding area of the true environment, it was not recognized that we ever returned to this particular state during the loop. Therefore, there was only one transition recorded from state 0 to state 1 according to the expected transition counts calculated by the algorithm. When projecting the angles to maintain additivity, the angle from state 0 to 1 was therefore compromised, allowing geometrical consistency to

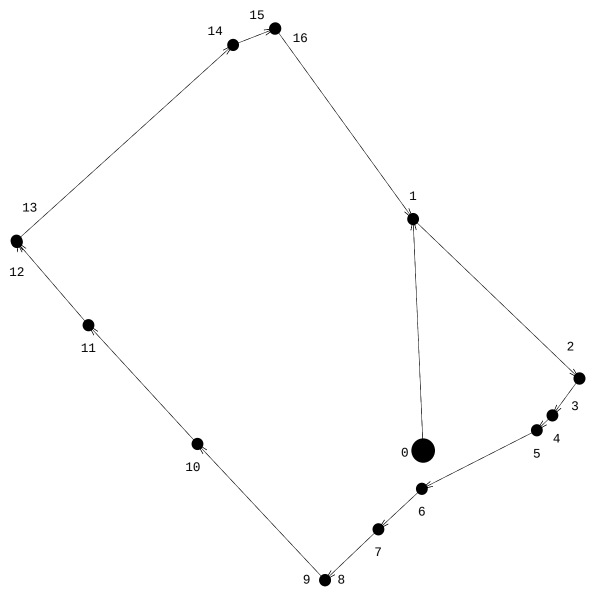

Figure 11.9: Learned model of the corridors Ramona traversed. Initialization is tag-based.

Seq.	Tag-based		k-means		Random		No Odo	
#	KL	Iter.#	KL	Iter. #	KL	Iter. #	KL	Iter. #
1	1.462	11.80	1.066	36.50	1.076	35.50	6.919	113.3
2	1.184	36.80	1.051	30.20	1.097	32.00	9.926	113.1
3	1.195	30.70	1.270	45.00	1.116	31.00	10.030	102.0
4	1.025	24.60	1.216	40.80	1.043	30.40	9.539	104.2
5	1.223	33.30	1.100	40.80	1.083	35.90	12.431	112.5

Table 11.2: Average results of four learning settings with five training sequences.

maintain the rectangular geometry among the more regularly visited states.

Figures 11.10 – 11.11 show two of the models obtained by applying our algorithm to the simulated sequences, enforcing the complete geometrical consistency while using the tag-based initialization methods. The geometry of rectangular combination is clear, but, obviously, these are highly inaccurate geometrical representations of the simulated environment.

For the purpose of quantitatively evaluating the learning algorithm we provide in Table 11.2 a summary of the results of running the algorithm under each of the 3 initialization settings, 10 times for each sequence. For comparison the results of the runs without odometric information are repeated here. As before, the KL divergence with respect to the true model is significantly smaller when odometric information is used. The standard deviation around the means is about 0.2 for KL values of models learned with odometry using

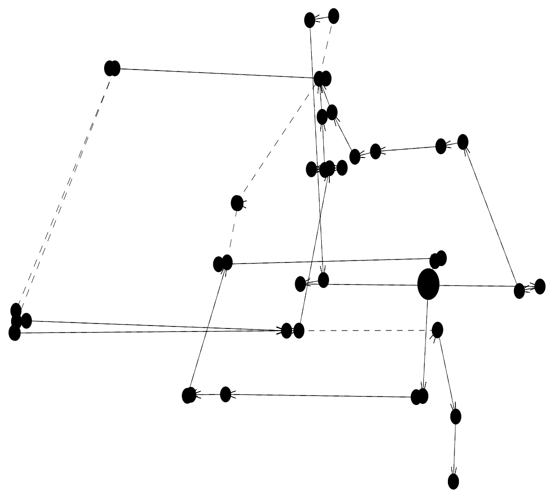
Figure 11.10:Learned model of the simulated environment.

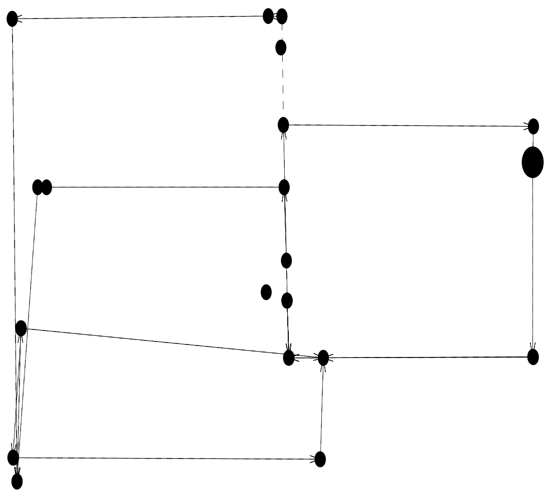
Figure 11.11:Learned model of the simulated environment.

tag-based initialization and about 0.1 for models learned from a random or k-means-based starting point.

The standard deviation around the number of iterations is about 10 for models started from either a random or a tag-based model and about 20 for models whose initialization was based on k-means. The number of iterations when additivity is enforced in the relative coordinate setting is consistently higher, with high statistical significance, than when only *anti-symmetry* is enforced, in the case of *k-means* based initialization. An explanation for this is that in the relative setting, the bad initialization starting from k-means leads to low likelihood values, which in turn causes numerical instability in the process of solving the set of equations required for enforcing additivity. This causes a slowdown in the convergence of the EM algorithm. An increase in the number of iterations, which is not as dramatic and not as statistically significant, also exists when using any of the other two initialization methods.

11.3 Studying the Effects of Odometry and Additivity

To better understand the impact of enforcing geometrical consistency in particular and that of odometric information in general, we study two small examples. We designed two models; One that intuitively does not require additivity in order to be learned, and the other which seemingly does require the enforcement of the complete geometrical consistency. We sampled data from both models, and applied our algorithms to it. The models, and the results analysis are described throughout the rest of this section.

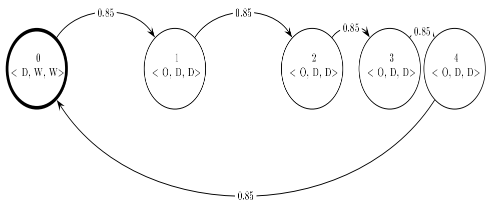

Figure 11.12:A model of a simulated environment. The distances between states are drawn to scale. Initial state is distinct.

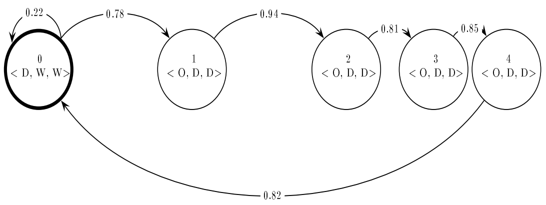

Figure 11.13:A learned model. Initialization is Tag-based. Distances are drawn to scale.

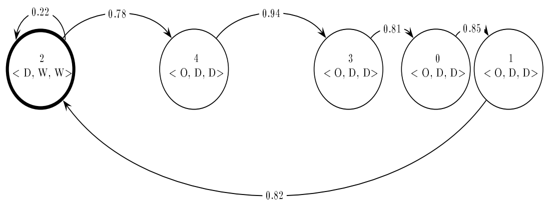

Figure 11.14:A learned model. Initialization is k-means based.

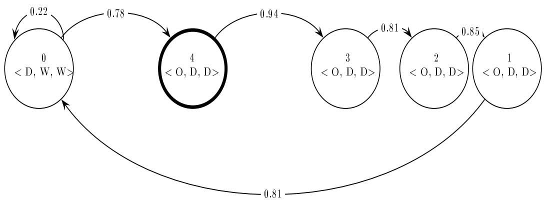

Figure 11.15:A learned model. Initialization is k-means based.

Experiment 11.1 To examine the effects of odometric information on learning a model, we used a small cyclic model, for which the initial state has a distinct tuple of observations, while all other states look alike observation-wise, but can be distinguished by their geometrical placement in the environment. Figure 11.12 shows a geometric layout of the model as a probabilistic state-transition diagram. Note that the states are placed in different distances from each other. This placement is significant, and represents the geometry of the state space; all the states lie on the same line, with diminishing distances from one state to the next. The distances between the states in the figure are drawn to scale. Edges denote transitions with probability greater than 0.2. The numbers on the edges correspond to the actual transition probabilities. At each state, the observations consist of the view on the front, left and right, which can be either a door, *D*, a wall, *W*, or and open area *O*. The most likely observations are shown in the diagram. These observations are typically seen 80-90 percent of the times when a state is visited.

Using Monte-Carlo sampling we generated two sequences of 300 observations and learned 5 models from each of them under each of the four learning settings, enforcing only anti-symmetry during the learning process.

Figure 11.13 shows a typical model learned using tag-based initialization. Again, the states are placed according to the relation matrix learned. It is clear that there is an almost perfect correspondence between both the geometry and the topology under this learning setting. All models learned using this algorithm look almost identical to this one.

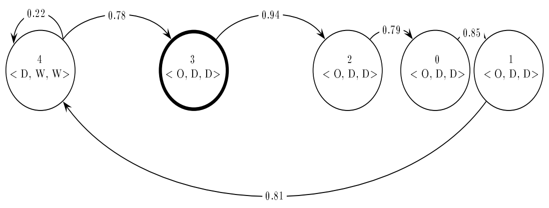

Figure 11.16:A learned model. Initialization is random.

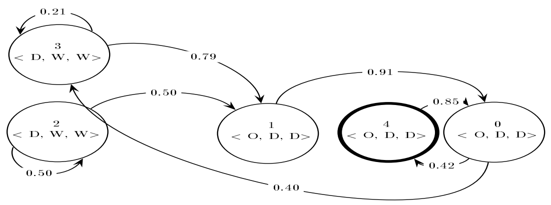

Figure 11.17:A learned model. Initialization is random.

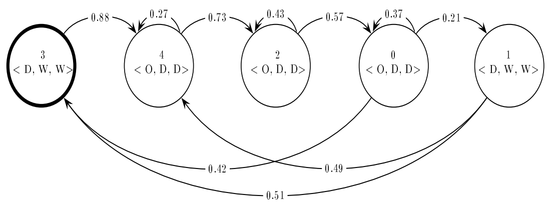

Figure 11.18:A learned model. No odometry used. State placing is arbitrary,

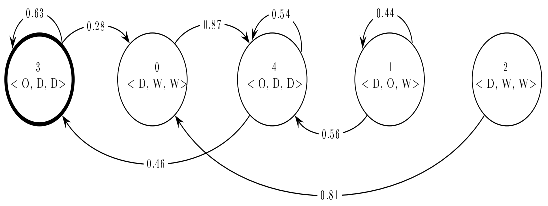

Figure 11.19:A learned model. No odometry used. State placing is arbitrary.

Figures 11.14 and 11.15 demonstrate the results of learning using k-means based initialization. The figure on the left shows a typical good result, where both topology and geometry almost exactly fit the true model. However, when starting from a poor clustering, the model is not as good; although it is still topologically correct, the initial state is not correctly identified. This is shown in Figure 11.15.

Figures 11.16 and 11.17 show the results of learning starting from a randomly initialized model. The figure on the left shows the less frequent case in which the algorithm managed to learn a good model despite the random initialization. The figure on the right demonstrates one of the less well-learned models, in which the geometry is clearly not as good as in all the other odometric model. The two leftmost states are actually placed in the same position according to the relation matrix, and are placed one above the other for the sake of readability only. These two states correspond the initial state in the original model. The cyclicity is still present and the observations are still the same as in the original model, but both the topological and the geometrical structure is different from the true one.

Figures 11.18 and 11.19 show the results of learning without the use of odometry. It is important to note that the placement of the states here has *no geometrical significance*, and the layout is imposed in order to clarify the plot. Figure 11.18 depicts one of the better models, in which the observation distribution corresponds well to that of the original model although the topology is different. In Figure 11.19, we see that in state 1, the triple $\langle D, O, W \rangle$ that is highly unlikely in the original model was learned as a likely observation triple, although, in an unlikely-to-be-reached state.

Seq. #	Tag-based Mean	Tag-based StD	k-means Mean	k-means StD	Random Mean	Random StD	No Odo Mean	No Odo StD
1	0.073	0.019	0.641	0.766	0.699	0.101	0.609	0.198
2	0.032	0.006	0.091	0.067	0.805	0.239	0.987	1.117

Table 11.3: Average results of four learning settings with two training sequences.

We can see that under this setting, odometry greatly helps to distinguish the states from each other, and the tag-based initialization ensures consistently good results.

To verify these results we also used the Kullback-Liebler divergence, evaluated based on generating 50 sequences of length 20 each from the true model and measuring the difference between their likelihood with respect to the true model and the likelihood with respect to the learned model (averaging over the total number of data points – 1000 in this case). Table 11.3 lists the means and standard deviations of the Kullback-Liebler measure for each of the sequences averaged over the 5 different learning experiences. The KL divergence for models learned using tag-based initialization are much smaller than for those learned using any other initialization method, or not using odometry at all. Using k-means based initialization typically gives very good results in this setting, but due to two severe outliers when starting from bad initial clustering, the mean is not much lower than when using random initialization or no odometry in the case of the first sequence. Note that the geometrical setting of the model is such that there are no turns, and therefore cumulative rotational error does not interfere with the effectiveness of the k-means based initialization. (Still, the algorithm used here does take into account cumulative rotational errors, as described in Chapter 9.)

Models learned using odometry, starting from a random initialization, are not much better than those learned without odometric information at all, although, at times the former still does very well, while the latter sometimes performs much worse. (Specifically, when using the second sequence the difference between the two settings is apparent). It is important to note that the example uses a small model which is not very peaked, and that there was an abundance of data provided from it. Hence, even the non-odometric learning performed quite well, if all we are concerned about is the probability distribution over sequences.

Experiment 11.2 To examine the effects of enforcing additivity on the learning process, it takes a model in which not only is odometry needed to *distinguish* between seemingly similar states, but also, odometry helps to tell that a state reached via two distinct routes

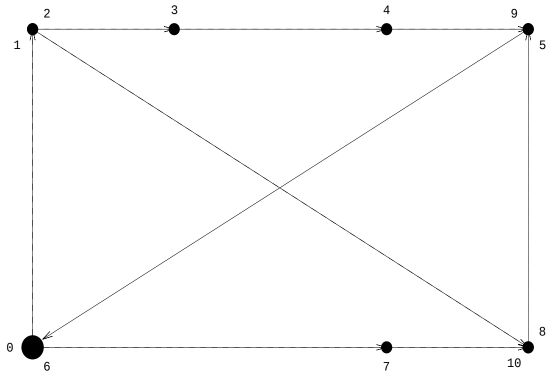

Figure 11.20:A model of a simulated environment.

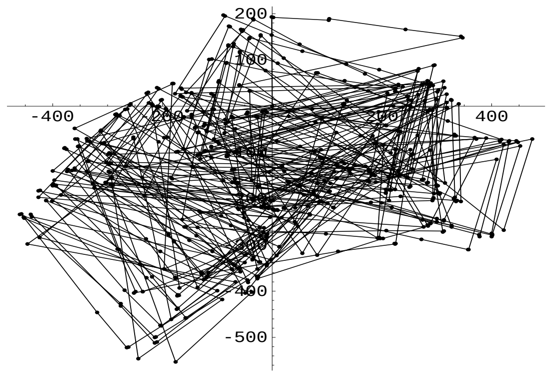

Figure 11.21:A geometrical projection of a 600 observations sequence sampled from the model

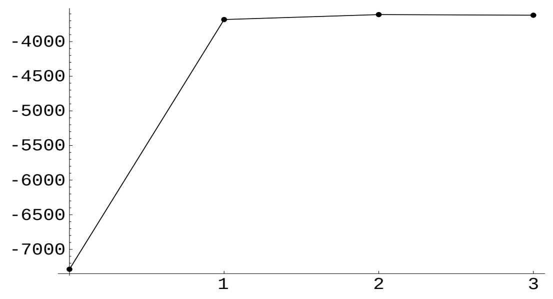

Figure 11.22:Log-likelihood as a function of the number of iterations. Learning from sequence 2.

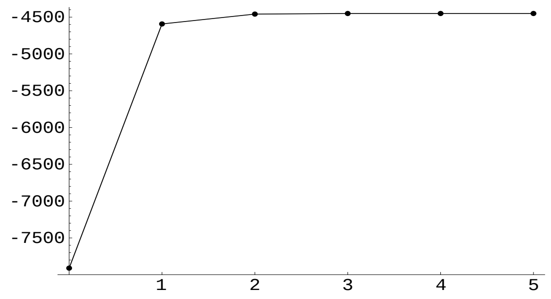

Figure 11.23:Log-likelihood as a function of the number of iterations. Learning from sequence 3.

is still the *same state*. This is where we expect geometrical consistency to play a major role. We use a small model as shown in Figure 11.20. State 5 is reachable from state 0 both by going north, turning east, and then turning north again, (the state sequence in this case is: $\langle 0, 1, 2, 3, 4, 9, 5\rangle$) and also by first going east then north (the state sequence is $\langle 0, 6, 7, 10, 8, 5\rangle$). From state 5 we can go directly to state 0, and also, from state 2 we can go with equal probability either to state 3 or to state 8.

Using Monte-Carlo sampling we generated three sequences of 600 observations and learned 5 models from each of them under all learning settings, with and without the enforcement of additivity. One of these sequences is plotted in Figure 11.21. The results presented here are only concerned with the value of enforcing additivity. Therefore, we limit our discussion to using the tag-based initialization which is the superior initialization method according to all of our experiments, in the presence of cumulative rotational error.

Typically when the model is small, tag-based initialization performs well and plays a significant role in capturing the geometry of the model. However, the iterative learning

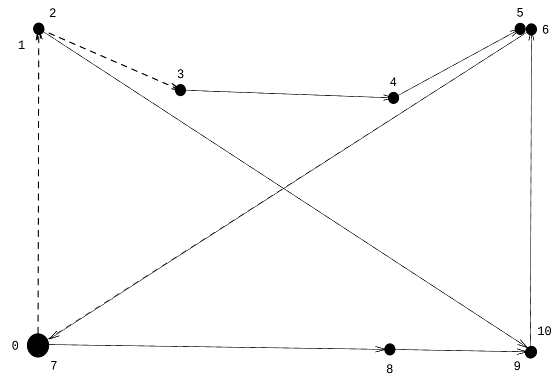

Figure 11.24:Model learned from sequence 1. Additivity is enforced.

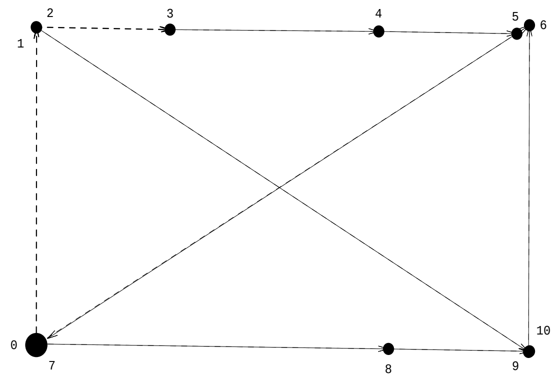

Figure 11.25:Model learned from sequence 1. Only anti-symmetry is enforced.

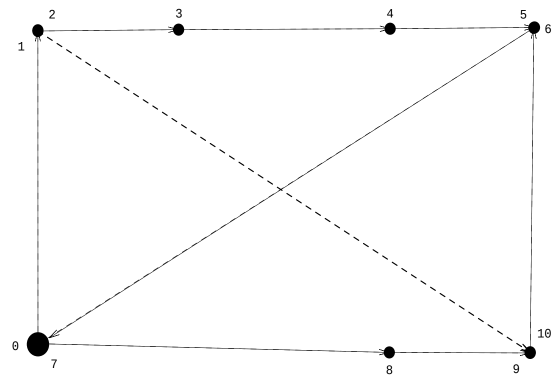

Figure 11.26:Model learned from sequence 2. Additivity is enforced.

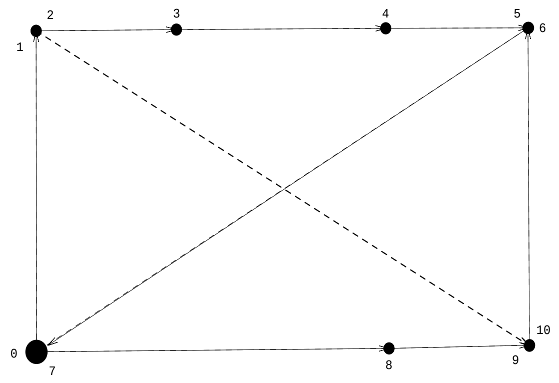

Figure 11.27:Model learned from sequence 2. Only anti-symmetry is enforced.

procedure still contributes a lot to learning the model parameters and improving the likelihood. This is demonstrated by the plots in Figures 11.22 and 11.23. The figures show the log-likelihood, as it increases during the learning process, as a function of the number of iterations, for two typical runs of the learning algorithm. The likelihood at iteration 0 is the likelihood of the data given the *initial* model. There is always a significant increase in the likelihood function following the first iteration of the algorithm, which shows that the initialization stage is not sufficient, in and of itself, to account for the quality of the learned models.

Figures 11.24 – 11.28 depict models that were learned from each of the three sequences using the enforcement of additivity, while Figures 11.25 – 11.29 depict models learned with only anti-symmetry enforced. The variability around these example models is very small, due to the almost-deterministic nature of the initialization method. The bold dashed arrows correspond to transitions that are very close in probability to the most likely one.

Figure 11.24 demonstrates that even though the model is learned very well from the

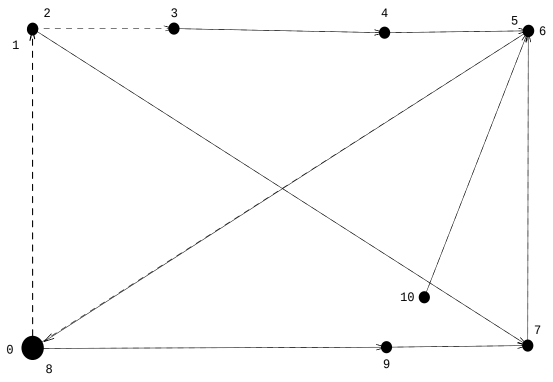

Figure 11.28:Model learned from sequence 3. Additivity is enforced.

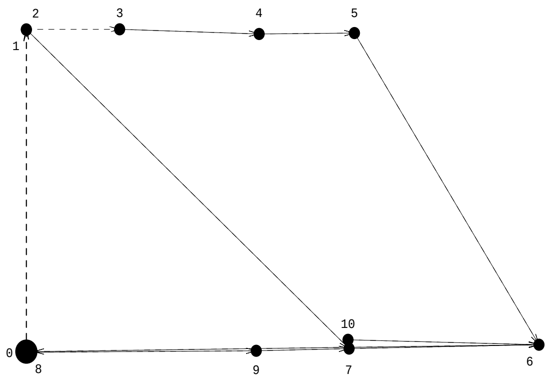

Figure 11.29:Model learned from sequence 3. Only anti-symmetry is enforced.

data sequence without the enforcement of additivity (Figure 11.25), our attempt to preserve geometrical consistency has compromised some of the angular relationships between states. States 3 and 4, which should have been aligned with states 1 and 5, are not placed exactly where they should be. It can be explained as a side-effect of the dependency on the relationship of each state to state 0 when enforcing global consistency between every pair of states (see Section 10.2). If the relations to state 0 are not substantiated by data, it is likely that the estimates obtained from the constrained equations are compromised. Future research will examine the possibility to change the reference system when enforcing geometrical consistency to be with respect to state transitions that are estimated to have been traversed the most.

Figures 11.26 and 11.27 demonstrate a case in which the results are very similar with and without enforcing additivity.

Figures 11.28 and 11.29 demonstrate a case in which the enforcement of additivity actually makes a significant difference in the quality of the learned model. Here we see that an almost correct model is obtained by the use of additivity (aside for state 10, which is unreachable and probably should have corresponded to state 10 in the true model). Without additivity, the model learned (Figure 11.29) is not geometrically consistent. State 6 is placed as shown in the figure when drawn with respect to state 7, while it should be close to state 5 if drawn with respect to state 5. Note that the returning edge to the initial state is from state 7 rather than from state 5.

There is no significant difference in the Kullback-Leibler divergence between the symmetric and the additive case.

In summary, to obtain models that are good both geometrically and topologically, it is crucial to start with a good initial model. The tag-based method we have developed proves useful in many cases. It is particularly effective for small models in which the accumulated variance calculated and used when filling the relations table is not too big. An important research direction is that of learning small pieces of a large model and then combining the small models into one large model of the complete environment.

The enforcement of geometrical consistency throughout the model can be helpful at times, but is not, in its present form, guaranteed to be significantly advantageous over the simpler anti-symmetry enforcement. In addition, it may increase the number of iterations required for convergence. By strongly constraining the learning process, it may even result in topologically inferior models, as demonstrated by our larger simulated experiments in Sections 11.1 and 11.2. It is possible that enforcing anti-symmetry at the early stages of the learning process, and the complete geometrical consistency only towards the end, would allow the algorithm to converge to better models. As for the consistency-enforcement process itself, an alternative choice of the basis for the consistency constraints may rely on the estimated state transitions counts, choosing as a basis the transitions that have the most estimated counts.

Chapter 12

Conclusions and Future Work

In this work we introduced a way in which readily available information is used to improve the quality of acquired models for robot navigation, as well as to reduce the resources required for obtaining such models.

We have shown that the separation commonly made between geometrical and topological models as mutually exclusive entities, (see an extensive discussion by Thrun [Thr99]), is not necessary. Not only can it be bridged, as done by Thrun in a two-tiered fashion by learning first a geometrical model and then a topological one from it, but rather, geometrical-in-nature odometric information can be directly incorporated into the topological realm, and used to improve the acquisition process of a topological model. This section summarizes the contributions of the thesis, and surveys several directions for future work.

12.1 Contributions

The theoretical contributions of the thesis are as follows:

- Extension of the formal hidden Markov models framework to accommodate odometric information.

- Extension of the Baum-Welch algorithm to use odometric information for learning hidden Markov models, providing proof of convergence for most of the algorithmic extensions.

- Pointing out several special issues in handling directional data in the context of robot navigation. In particular, the need for directional distributions such as the von-Mises distribution, and the associated estimation procedures.

- Providing a basis for maintaining geometrical consistency throughout the system, both in terms of projection and in terms of direct optimization under geometrical constraints. Such statistical estimation under constraints is hardly treated in the main-stream statistical literature [Bar84].

The practical contributions are:

- Implementation of the learning algorithm for both POMDPs and HMMs, as well as the supporting packages for parsing, testing, comparing, sampling sequences from these models, and for plotting HMMs.
- Empirical tests showing that through the use of odometric information better models can be learned, while requiring fewer iterations and shorter data sequences.
- Application of the algorithm to real robot data gathered from a globally ambiguous environment which contains loops. Learning models for environments with loops is considered one of the hardest problems in model acquisition for robot navigation.
- A new heuristic for finding an initial model, based on odometric information. The algorithm is robust in the presence of cumulative rotational error, and may also serve as a possible basis for estimating the number of states in the model.

12.2 Future Work

Learning topological maps through the use of odometric information is by no means a solved problem. First, we stress that the reduction in the number of iterations does not currently translate to a reduction in the expected run time, since the computation of the normal and von-Mises distribution for each data point and each pair of states in each iteration is an expensive operator. Through the use of lookup-tables, caching, and exploitation of the symmetries in the relations table, this cost should be reduced, allowing us to take advantage of the fewer iterations. Our current naïve implementation does not benefit much from the fewer iterations, aside from the fact that the run time would have been much greater if to achieve the same quality of models as our algorithm achieves, we would have needed as many iterations as in the non-odometric case.

As demonstrated in Chapter 11, the geometrical-consistency maintenance suffers from two main drawbacks. It is more complicated due to the need to solve a potentially large set of linear equations, and it depends on obtaining good estimates for the values of μ_{0i} along all dimensions. Despite the relatively complicated reestimation procedure, having to solve a set of equations is not a computational bottleneck of the algorithm. However, it may still prove beneficial to use advanced techniques for treating sparse matrices for solving the equations. Such techniques may also prove effective for addressing the numerical instabilities occurring while solving the equations. An important direction to explore, addressing the dependency on the means with respect to state 0, is that of choosing the expressions for geometrical constraints to be based on the transitions which have the most support. This may be done in a way similar to the one taken for treating the projection of heading estimates. Such an approach might complicate the algorithm on one hand, but may make it much more accurate on the other.

The value of enforcing complete geometrical consistency is not fully determined. It may be an effective learning procedure to enforce anti-symmetry rather than complete geometrical consistency for several learning iterations, enforcing complete geometrical consistency only in the last stages of the learning process. An additional alternative is to enforce additivity over the x and y dimensions, while enforcing only anti-symmetry along the heading dimension.

Another important issue is the understanding of the effects of the parameters that are currently provided to the algorithm. These include the number of states, the initial default variance, and the weight assigned to sines and cosines when calculating the distance between odometric measures. Our experience has shown that all of the latter parameters effect the results of the initialization (both the k-means and the tag-based), and although it is feasible to adjust these few parameters manually according to the problem at hand, it is desirable to have definite guidelines in choosing them, or a fully automated procedure that does it based on the magnitude of the input data. We stress that all the experiments reported in this work were conducted under the same fixed set of constants. The only varying parameter was the number of states in the model, as explicitly stated.

An additional direction we have started to explore is that of combining learning the model and planning within it into a unified framework, based on reinforcement learning. The basic idea behind this possible extension, is to take the current learned model of a POMDP and augment each state with a reward that is *invertly proportional* to our confidence in the accuracy of the probability distributions currently associated with the state. That is, states whose distributions are believed to be well supported by the data sequence from which the model was learned, are assigned *low* rewards while states whose distributions are unsupported by enough data are assigned *high* rewards. The robot, moving in the environment according to its current model, trying to maximize its expected reward, would concentrate on arriving at those states about which it knows the least, thus obtaining more data about them. It can then improve the current model based on the newly obtained data. As a consequence – both the model and the reward assignment change. The process then continues under the updated model and reward function. Such "ignorance rewards" can potentially be combined with the standard POMDP rewards, thus enabling the tasks of *planning* within a POMDP environment [CKL94, Cas98] and *learning* it to be combined.

12.3 Beyond Robotics

Hidden Markov models serve as useful modeling tools in a variety of domains other than robot navigation, from natural language understanding [Cha93] to computational molecular biology [BCH+93, KBM+94].

Our work demonstrates that through the use of domain-specific information and constraints, automatic model acquisition is made more effective while requiring fewer iterations and less gathered data. We strongly believe that this idea can be applied for learning HMMs and POMDP models in areas other than robotics.

One appealing application domain is medical decision support. Probabilistic models such as Bayesian networks and POMDP models have been recently introduced as aids for diagnostics and decision making in medicine [SDL+93, SOA97, HF98]. The patient's state and symptoms that evolve through time as a result of treatment, can be naturally modeled as a POMDP. Various conditions that the patient may be in are mutually exclusive and time dependent. Thus there are many potential constraints on the change in the patient's condition. These constraints can be exploited in order to learn models both for the development of the disease and for the expected change in the patient's state as treatment is applied. Such models can be of great value for predicting the results of possible treatments, and for assisting physicians in deciding on such treatments.

Another area that is rapidly developing is computational biology [KBM+94, GM96, FMG+97, KSB+97]. Hidden Markov models are already successfully used for modeling proteins and DNA sequences. Such large molecules have an intricate 3-dimensional geometrical structure. It is likely that by enforcing geometrical constraints, similar to the ones discussed throughout this work, acquiring models for proteins and DNA sequences can be made better and faster.

In both the medical and the biological domain, the ability to learn from relatively small quantities of data is particularly important. There are medical treatments that are rarely applied due to their high cost or high risk, as well as medical conditions that are rarely encountered. Similarly, some families of proteins, for which models need to be learned, have only a few instances that are fully analyzed. In order to obtain models from the existing data in these cases, it is important to be able to take advantage of the available data to the fullest. As demonstrated by the experiments described in previous chapters, our algorithm retains its good performance even when the amount of data available to it is significantly reduced. This capability is expected to be of great value if our approach is applied in the bio-medical domain.

Appendix A

An Overview of the Odometric Learning Algorithm for HMMs

The algorithm takes as input an experience sequence $\mathsf{E} = \langle r, V \rangle$, consisting of the odometric sequence r and the observation sequence V.

```
LEARN ODOMETRIC HMM($\mathsf{E}$)
 1  Initialize matrices $A, B, R$        (See Chapter 6)
 2  $max\_change \leftarrow \infty$
 3  while ( $max\_change > \epsilon$)
 4     do Calculate Forward probabilities, $\alpha$            (Formula 4.1)
 5        Calculate Backward probabilities, $\beta$           (Formula 4.2)
 6        Calculate state-occupation probabilities, $\gamma$    (Formula 4.3)
 7        Calculate State-transition probabilities, $\xi$,     (Formula 4.4)
 8        $Old\_A \leftarrow A$,   $Old\_B \leftarrow B$
 9        $A \leftarrow$ Reestimate ($A$)             (Formula 4.5)
10        $B \leftarrow$ Reestimate ($B$)             (Formula 4.6)
11        $R^{\theta} \leftarrow$ Reestimate ($R^{\theta}$)           (Formulae 5.4  and 5.6)
12        $\langle R^x, R^y \rangle \leftarrow$ Reestimate($R^x$, $R^y$), using either
             • Formulae 4.13 and 4.12 (within a global framework), or
             • Formulae 8.3,  8.4 and 4.12 (within a relative framework)
13        $max\_change \leftarrow$ MAX(GET_MAX_CHANGE($A$, $Old\_A$ ),
                                   GET_MAX_CHANGE($B$, $Old\_B$ ))
```

If additivity is enforced, step 13 is followed by a projection of the reestimated R^{θ} onto an additive affine space, as described in Section 10.3. In addition, step 12 is substituted by the procedure described in either Section 10.1 or 10.2. That is, if we are operating within a global framework, the equations denoted by Formula 10.3 are solved and the means are calculated from the solution according to equation 10.4. If we are operating within a state-relative framework, the system of equations B.1, B.2 is solved, and the means are calculated according to equations 10.6 and 10.7.

GET_MAX_CHAGE is a function that takes two matrices and returns the maximal element-wise absolute difference between them.

Appendix B

Differentiation Details

We provide here two differentiations whose details were omitted earlier.

B.1 Unconstrained Odometric Reestimation Formulae

In Section 4.3.2, Formula 4.19, we rewrote Baum's auxiliary function, Q, restricted to a pair of states i, j, and for a single odometric dimension, m, as:

$$Q_{ij}^{m}(R, \overline{R}) = \sum_{t=0}^{T-2} \xi_t(i,j)(\log(\overline{f}_{i,j}^{m}(r_{t+1}^{m})) - \log(\overline{\sigma}_{ij}^{m})) \ ,$$

and claimed that by setting its partial derivatives, $\frac{\partial Q_{ij}^m}{\partial \overline{\mu}_{ij}^m}$ and $\frac{\partial Q_{ij}^m}{\partial \overline{\sigma}_{ij}^m}$ to 0, we obtain the unconstrained reestimation formulae 4.7 and 4.8:

$$\overline{\mu}_{i,j}^{m} = \frac{\sum_{t=0}^{T-2} r_t[m]\xi_t(i,j)}{\sum_{t=0}^{T-2}\xi_t(i,j)} \ , \qquad \overline{\sigma}^{2}{}_{i,j}^{m}) = \frac{\sum_{t=0}^{T-2}(r_t[m] - \overline{\mu}_{i,j}^{m})^2\xi_t(i,j)}{\sum_{t=0}^{T-2}\xi_t(i,j)} \ .$$

The derivative of Q_{ij}^m with respect to $\overline{\mu}_{ij}^m$ is:

$$\frac{\partial Q_{ij}^m}{\partial \overline{\mu}_{ij}^m} = \frac{\partial \sum_{t=0}^{T-2} \xi_t(i,j)\frac{-(r_t^m - \overline{\mu}_{ij}^m)^2}{2(\overline{\sigma}_{ij}^m)^2}}{\partial \overline{\mu}_{ij}^m} = \frac{\sum_{t=0}^{T-2} \xi_t(i,j)(r_t^m - \overline{\mu}_{ij}^m)}{(\overline{\sigma}_{ij}^m)^2} \ .$$

By setting this derivative to 0, we obtain the equation:

$$\frac{\sum_{t=0}^{T-2} \xi_t(i,j)(r_t^m - \overline{\mu}_{ij}^m)}{(\overline{\sigma}_{ij}^m)^2} = 0 \ ,$$

whose solution is indeed: $$\overline{\mu}_{ij}^m = \frac{\sum_{t=0}^{T-2} \xi_t(i,j) r_t^m}{\sum_{t=0}^{T-2} \xi_t(i,j)} \ .$$

The derivative of Q_{ij}^m with respect to $\overline{\sigma}_{ij}^m$ is:

$$\frac{\partial Q_{ij}^m}{\partial \overline{\sigma}_{ij}^m} = \frac{\partial \sum_{t=0}^{T-2} \xi_t(i,j)\left(\frac{-(r_t^m - \overline{\mu}_{ij}^m)^2}{2(\overline{\sigma}_{ij}^m)^2} - \log(\overline{\sigma}_{ij}^m)\right)}{\partial \overline{\sigma}_{ij}^m} = \sum_{t=0}^{T-2} \xi_t(i,j)\left(\frac{(r_t^m - \overline{\mu}_{ij}^m)^2}{(\overline{\sigma}_{ij}^m)^3} - \frac{1}{\overline{\sigma}_{ij}^m}\right) \ .$$

Setting it to 0 results in the equation:

$$\sum_{t=0}^{T-2} \xi_t(i,j)\left(\frac{(r_t^m - \overline{\mu}_{ij}^m)^2}{(\overline{\sigma}_{ij}^m)^3} - \frac{1}{\overline{\sigma}_{ij}^m}\right) = 0 \ ,$$

whose solution is indeed: $$(\overline{\sigma}_{ij}^m)^2 = \frac{\sum_{t=0}^{T-2} \xi_t(i,j)(r_t^m - \overline{\mu}_{ij}^m)^2}{\sum_{t=0}^{T-2} \xi_t(i,j)} \ . \qquad \square$$

B.2 Enforcing Additivity within a Relative Framework

In Section 10.2, Formula 10.5, we rewrote Baum's auxiliary function, Q, restricted to the odometric dimensions x, y, as a function of the locations, $\langle x_0^0, y_0^0\rangle, \ldots \langle x_{N-1}^0, y_{N-1}^0\rangle$, of states $s_0, \ldots, s_{N-1}$, along the global x and y coordinate system, as follows:

$$Q^{x,y}(R,\overline{R}) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\sum_{t=0}^{T-2} \xi_t(i,j)\left(\frac{-(r_{t+1}^x - (\cos(\mu_{0,i}^\theta)(x_j^0 - x_i^0) - \sin(\mu_{0,i}^\theta)(y_j^0 - y_i^0)))^2}{2(\sigma_{ij}^x)^2}\right.$$
$$\left. - \log(\sigma_{ij}^x) - \frac{(r_{t+1}^y - (\sin(\mu_{0,i}^\theta)(x_j^0 - x_i^0) + \cos(\mu_{0,i}^\theta)(y_j^0 - y_i^0)))^2}{2(\sigma_{ij}^y)^2} - \log(\sigma_{ij}^y)\right) \ .$$

The location $\langle x_0^0, y_0^0\rangle$ is assumed to be the origin, $\langle 0, 0\rangle$. Rotations according to the heading changes with respect to the origin, $\mu_{0,0}^\theta, \ldots, \mu_{0,N-1}^\theta$, are applied in order to account for the representation of the odometric relation within a state-relative framework.

We stated that by differentiating this expression according to each x_j^0 and each y_j^0 where $j \neq 0$, and equating each derivative to 0 we obtain a set of $(2N-2)$ linear equations in $(2N-2)$ unknowns. These equations are solved at each iteration of the EM algorithm when additivity in a relative framework is enforced.

We give here the explicit expressions for the derivatives and the resulting equations that are solved at each iteration.

For each x_k^0 and y_k^0, where $0 < k \leq N-1$:

$$\frac{\partial Q^{x,y}(R,\overline{R})}{\partial x_k^0} =$$

$$\sum_{\substack{i=0\\ i\neq k}}^{N-1} \left(\sum_{t=0}^{T-2} \xi_t(i,k) \left[\frac{-(r_t^x - \cos(\mu_{0,i}^\theta)(x_k^0 - x_i^0) + \sin(\mu_{0,i}^\theta)(y_k^0 - y_i^0))(-\cos(\mu_{0,i}^\theta))}{(\sigma_{ik}^x)^2} - \right.\right.$$
$$\left. \frac{(r_t^y - \sin(\mu_{0,i}^\theta)(x_k^0 - x_i^0) - \cos(\mu_{0,i}^\theta)(y_k^0 - y_i^0))(-\sin(\mu_{0,i}^\theta))}{(\sigma_{ik}^y)^2} \right] +$$
$$\sum_{t=0}^{T-2} \xi_t(k,i) \left[\frac{-(r_t^x - \cos(\mu_{0,k}^\theta)(x_i^0 - x_k^0) + \sin(\mu_{0,k}^\theta)(y_i^0 - y_k^0))\cos(\mu_{0,k}^\theta)}{(\sigma_{ki}^x)^2} - \right.$$
$$\left.\left. \frac{(r_t^y - \sin(\mu_{0,k}^\theta)(x_i^0 - x_k^0) - \cos(\mu_{0,k}^\theta)(y_i^0 - y_k^0))\sin(\mu_{0,k}^\theta)}{(\sigma_{ki}^y)^2} \right] \right) \quad \text{and}$$

$$\frac{\partial Q^{x,y}(R,\overline{R})}{\partial y_k^0} =$$

$$\sum_{\substack{i=0\\ i\neq k}}^{N-1} \left(\sum_{t=0}^{T-2} \xi_t(i,k) \left[\frac{-(r_t^x - \cos(\mu_{0,i}^\theta)(x_k^0 - x_i^0) + \sin(\mu_{0,i}^\theta)(y_k^0 - y_i^0))\sin(\mu_{0,i}^\theta)}{(\sigma_{ik}^x)^2} - \right.\right.$$
$$\left. \frac{(r_t^y - \sin(\mu_{0,i}^\theta)(x_k^0 - x_i^0) - \cos(\mu_{0,i}^\theta)(y_k^0 - y_i^0))(-\cos(\mu_{0,i}^\theta))}{(\sigma_{ik}^y)^2} \right] +$$
$$\sum_{t=0}^{T-2} \xi_t(k,i) \left[\frac{-(r_t^x - \cos(\mu_{0,k}^\theta)(x_i^0 - x_k^0) + \sin(\mu_{0,k}^\theta)(y_i^0 - y_k^0))(-\sin(\mu_{0,k}^\theta))}{(\sigma_{ki}^x)^2} - \right.$$
$$\left.\left. \frac{(r_t^y - \sin(\mu_{0,k}^\theta)(x_i^0 - x_k^0) - \cos(\mu_{0,k}^\theta)(y_i^0 - y_k^0))\cos(\mu_{0,k}^\theta)}{(\sigma_{ki}^y)^2} \right] \right) .$$

By equating all these partial derivatives to 0, we get for each k, where $0 < k \leq N-1$, two equations as follows:

$$\begin{aligned}
&\sum_{\substack{i=0\\ i\neq k}}^{N-1}\left[\sum_{t=0}^{T-2}\xi_t(i,k)\left(\frac{r_t^x\cos(\mu_{0,i}^\theta)}{(\sigma_{ik}^x)^2}+\frac{r_t^y\sin(\mu_{0,i}^\theta)}{(\sigma_{ik}^y)^2}\right)-\sum_{t=0}^{T-2}\xi_t(k,i)\left(\frac{r_t^x\cos(\mu_{0,k}^\theta)}{(\sigma_{ki}^x)^2}+\frac{r_t^y\sin(\mu_{0,k}^\theta)}{(\sigma_{ki}^y)^2}\right)\right]=\\
&\quad\sum_{\substack{i=0\\ i\neq k}}^{N-1}(x_k^0-x_i^0)\left[\sum_{t=0}^{T-2}\xi_t(i,k)\left(\frac{\cos^2(\mu_{0,i}^\theta)}{(\sigma_{ik}^x)^2}+\frac{\sin^2(\mu_{0,i}^\theta)}{(\sigma_{ik}^y)^2}\right)+\right.\\
&\qquad\qquad\left.\sum_{t=0}^{T-2}\xi_t(k,i)\left(\frac{\cos^2(\mu_{0,k}^\theta)}{(\sigma_{ki}^x)^2}+\frac{\sin^2(\mu_{0,k}^\theta)}{(\sigma_{ki}^y)^2}\right)\right]-\\
&\quad\sum_{\substack{i=0\\ i\neq k}}^{N-1}(y_k^0-y_i^0)\left[\sum_{t=0}^{T-2}\xi_t(i,k)\sin(\mu_{0,i}^\theta)\cos(\mu_{0,i}^\theta)\left(\frac{1}{(\sigma_{ik}^x)^2}-\frac{1}{(\sigma_{ik}^y)^2}\right)+\right.\\
&\qquad\qquad\left.\sum_{t=0}^{T-2}\xi_t(k,i)\sin(\mu_{0,k}^\theta)\cos(\mu_{0,k}^\theta)\left(\frac{1}{(\sigma_{ki}^x)^2}-\frac{1}{(\sigma_{ki}^y)^2}\right)\right]\quad,
\end{aligned}\tag{B.1}$$

$$\begin{aligned}
&\sum_{\substack{i=0\\ i\neq k}}^{N-1}\left[\sum_{t=0}^{T-2}\xi_t(i,k)\left(\frac{r_t^x\sin(\mu_{0,i}^\theta)}{(\sigma_{ik}^x)^2}-\frac{r_t^y\cos(\mu_{0,i}^\theta)}{(\sigma_{ik}^y)^2}\right)+\sum_{t=0}^{T-2}\xi_t(k,i)\left(\frac{r_t^y\cos(\mu_{0,k}^\theta)}{(\sigma_{ki}^x)^2}-\frac{r_t^x\sin(\mu_{0,k}^\theta)}{(\sigma_{ki}^x)^2}\right)\right]=\\
&\quad\sum_{\substack{i=0\\ i\neq k}}^{N-1}(x_k^0-x_i^0)\left[\sum_{t=0}^{T-2}\xi_t(i,k)\sin(\mu_{0,i}^\theta)\cos(\mu_{0,i}^\theta)\left(\frac{1}{(\sigma_{ik}^x)^2}-\frac{1}{(\sigma_{ik}^y)^2}\right)+\right.\\
&\qquad\qquad\left.\sum_{t=0}^{T-2}\xi_t(k,i)\sin(\mu_{0,k}^\theta)\cos(\mu_{0,k}^\theta)\left(\frac{1}{(\sigma_{ki}^x)^2}-\frac{1}{(\sigma_{ki}^y)^2}\right)\right]-\\
&\quad\sum_{\substack{i=0\\ i\neq k}}^{N-1}(y_k^0-y_i^0)\left[\sum_{t=0}^{T-2}\xi_t(i,k)\left(\frac{\sin^2(\mu_{0,i}^\theta)}{(\sigma_{ik}^x)^2}+\frac{\cos^2(\mu_{0,i}^\theta)}{(\sigma_{ik}^y)^2}\right)+\right.\\
&\qquad\qquad\left.\sum_{t=0}^{T-2}\xi_t(k,i)\left(\frac{\sin^2(\mu_{0,k}^\theta)}{(\sigma_{ki}^x)^2}+\frac{\cos^2(\mu_{0,k}^\theta)}{(\sigma_{ki}^y)^2}\right)\right]\quad.
\end{aligned}\tag{B.2}$$

Bibliography

[AC82] S. Abeyasekera and D. Collett, On the Estimation of the Parameters of the von Mises Distribution, *Communications in statistics, Theory and methods*, 11 (18), pp. 2083–2090, 1982.

[Ang87] D. Angluin, Learning Regular Sets from Queries and Counterexamples, *Information and Computation*, 75, pp. 87–106, 1987.

[Apo69] T. M. Apostol, *Calculus*, vol. 2, pp. 303–320, Blaisdell Publishing Company, 1969.

[Asa91] M. Asada, Map Building for a Mobile Robot from Sensory Data, in *Autonomous Mobile Robots*, S. S. Iyengar and A. Elfes, eds., pp. 312–322, IEEE Computer Society Press, 1991.

[AW92] N. Abe and M. K. Warmuth, On the Computational Complexity of Approximating Distributions by Probabilistic Automata, *Machine Learning*, 9 (2), pp. 205–260, 1992.

[Bar84] R. Bartels, Estimation in a Bidirectional Mixture of von Mises Distributions, *Biometrics*, 40, pp. 777–784, 1984.

[BBS95] A. G. Barto, S. J. Bradtke and S. P. Singh, Learning to Act Using Real-time Dynamic Programming, *Artificial Intelligence*, 72, pp. 81–138, 1995.

[BCH$^+$93] P. Baldi, Y. Chauvin, T. Hunkapiller and M. A. McClure, Hidden Markov Models in Molecular Biology: New Algorithms and Applications, *Advances in Neural Information Processing*, 5, pp. 747–754, 1993.

[BDK95] K. Basye, T. Dean and L. P. Kaelbling, Learning Dynamics: System Identification for Perceptually Challenged Agents, *Artificial Intelligence*, 72 (1), 1995.

[BE67] L. E. Baum and J. A. Eagon, An Inequality with Applications to Statistical Estimation for Probabilistic Functions of Markov Processes and to a Model for Ecology, *American Mathematical Society Bulletin*, 73 (3), pp. 360–363, 1967.

[BG95] J.-M. Bertille and M. Gilloux, A Probabilistic Approach to Automatic Handwritten Address Reading, in *Proceedings of the Third International Conference on Document Analysis and Recognition*, pp. 368–371, 1995.

[Bil59] P. Billingsley, Statistical Methods in Markov Chains, *Annals of Mathematical Statistics*, 32, pp. 12–40, 1959.

[BPS$^+$70] L. E. Baum, T. Petrie, G. Soules and N. Weiss, A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains, *The Annals of Mathematical Statistics*, 41 (1), pp. 164–171, 1970.

[BS68] L. E. Baum and G. R. Sell, Growth Transformations for Functions on Manifolds, *Pacific Journal of Mathematics*, 27 (2), pp. 211–227, 1968.

[Cas98] A. R. Cassandra, *Exact and Approximate Algorithms for Partially Observable Markov Decision Processes*, Ph.D. thesis, Department of Computer Science, Brown University, Providence, RI, 1998.

[Cha93] E. Charniak, *Statistical Language Learning*, MIT Press, 1993.

[Chu89] G. A. Churchill, Stochastic Models for Hetertogeneous DNA Sequences, *Bulletin of Mathematical Biology*, 51 (1), pp. 79–94, 1989.

[CKK96] A. R. Cassandra, L. P. Kaelbling and J. A. Kurien, Acting Under Uncertainty: Discrete Bayesian Models for Mobile-Robot Navigation, in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1996.

[CKL94] A. R. Cassandra, L. P. Kaelbling and M. L. Littman, Acting optimally in Partially Observable Stochastic Domains, in *Proceedings of the Twelfth National Conference on AI*, pp. 1023–1028, Seattle, Washington, 1994.

[CKS$^+$90] P. Cheeseman, J. Kelly, M. Self, J. Stutz, W. Taylor and D. Freeman, AutoClass: A Bayesian Classification System, in *Readings in Machine Learning*, J. W. Shavlik and T. G. Dietterich, eds., pp. 296–306, Morgan-Kaufmann, 1990.

[CKZ94] M.-Y. Chen, A. Kundu and J. Zhou, Off-Line Handwritten Word Recognition Using a Hidden Markov Model Type Stochastic Network, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16 (5), pp. 481–496, 1994.

[DeG86] M. H. DeGroot, *Probability and Statistics*, Addison-Wesley, 2nd edn., 1986.

[DH73] R. O. Duda and P. E. Hart, *Unsupervised Learning and Clustering*, chap. 6, John Wiley and Sons, 1973.

[DLR77] A. P. Dempster, N. M. Laird and D. B. Rubin, Maximum Likelihood from Incomplete Data via the EM Algorithm, *Journal of the Royal Statistical Society*, 39 (1), pp. 1–38, 1977.

[Elf89] A. Elfes, Using Occupancy Grids for Mobile Robot Perception and Navigation, *Computer, Special Issue on Autonomous Intelligent Machines*, pp. 46–57, 1989.

[EM92] S. P. Engelson and D. V. McDermott, Error Correction in Mobile Robot Map Learning, in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 2555–2560, Nice, France, 1992.

[ET93] B. Efron and R. J. Tibshirani, *An Introduction to the Bootstrap*, chap. 17, pp. 237–257, Chapman-Hall, 1993.

[Fie86] M. Fiedler, *Special Matrices and their Applications in Numerical Mathematics*, chap. 5, pp. 112–135, Martinus Nijhoff Publishers, 1986.

[FMG+97] V. D. Francesco, P. McQueen, J. Garnier and P. J. Munson, Incorporating Global Information into Secondary Structure Prediction with Hidden Markov Models of Protein Folds, in *Proceedings of the International Conference on Intelligent Systems for Molecular Biology*, pp. 100–103, 1997.

[GGD53] E. G. Gumbel, J. A. Greenwood and D. Durand, The Circular Normal Distribution: Theory and Tables, *American Statistical Society Journal*, 48, pp. 131–152, 1953.

[GHW79] G. H. Golub, M. Heath and G. Wahba, Generalized Cross-Validation as a Method for Choosing a Good Ridge Parameter, *Technometrics*, 21 (2), pp. 215–223, 1979.

[GM96] V. Geetha and P. J. Munson, Simplified Representation of Proteins, *Journal of Biomolecular Structure and Dynamics*, 13 (5), pp. 781–793, 1996.

[GMR98] M. Golfarelli, D. Maio and S. Rizzi, Elastic Correction of Dead-Reckoning Errors in Map Building, in *Proceedings of the International Conference on Intelligent Robotic Systems*, B.C., Canada, 1998.

[Gol78] E. M. Gold, Complexity of Automaton Identification from Given Data, *Information and Control*, 37, pp. 302–320, 1978.

[GSD98] D. Gottlieb, M. Sever and A. Ditkowski, Private Discussion,Department of Applied Mathematics, Brown University, 1998.

[HF98] M. Hauskrecht and H. Fraser, Planning Medical Therapy Using Partially Observable Markov Decision Processes, in *Proceedings of the Ninth International Workshop on Principles fo Diagnostics*, pp. 182–189, Cape Cod, Massachusetts, 1998.

[HU79] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison & Wesley, 1979.

[JLS86] B. H. Juang, S. E. Levinson and M. M. Sondhi, Maximum Likelihood Estimation for Multivariate Mixture Observations of Markov Chains, *IEEE Transactions on Information Theory*, 32 (2), 1986.

[JR85] B. H. Juang and L. R. Rabiner, A Probabilistic Distance Measure for Hidden Markov Models, *AT&T Technical Journal*, 64 (2), pp. 391–408, 1985.

[Jua85] B. H. Juang, Maximum Likelihood Estimation for Mixture Multivariate Stochastic Observations of Markov Chains, *AT&T Technical Journal*, 64 (6), 1985.

[Kae93] L. P. Kaelbling, *Learning in Embedded Systems*, MIT Press, 1993.

[KB91] B. Kuipers and Y.-T. Byun, A Robot Exploration and Mapping Strategy Based on a Semantic Hierarchy of Spatial Representations, *Journal of Robotics and Autonomous Systems*, 8, pp. 47–63, 1991.

[KBM$^+$94] A. Krogh, M. Brown, I. S. Mian, K. Sjolander and D. Haussler, Hidden Markov Models in Computational Biology, Applications to Protein Modeling, *Journal of Molecular Biology*, 235, pp. 1501–1531, 1994.

[KJ82a] S. Kotz and N. L. Johnson, eds., *Encyclopedia of Statistical Sciences*, vol. 2, pp. 381–386, John Wiley and Sons, 1982.

[KJ82b] S. Kotz and N. L. Johnson, eds., *Encyclopedia of Statistical Sciences*, vol. 1, p. 479, John Wiley and Sons, 1982.

[KL51] S. Kullback and R. A. Leibler, On Information and Sufficiency, *Annals of Mathematical Statistics*, 22 (1), pp. 79–86, 1951.

[KS96a] S. Koenig and R. G. Simmons, Passive Distance Learning for Robot Navigation, in *Proceedings of the Thirteenth International Conference on Machine Learning*, pp. 266–274, 1996.

[KS96b] S. Koenig and R. G. Simmons, Unsupervised Learning of Probabilistic Models for Robot Navigation, in *Proceedings of the IEEE International Conference on Robotics and Automation*, 1996.

[KSB$^+$97] K. Karplus, K. Sjölander, C. Barrett, M. Cline, D. Haussler, R. Hughey, L. Holm and C. Sander, Predicting Protein Structure using Hidden Markov Models, *Proteins: Structure, Function, and Genetics, Supplement*, (1), pp. 134–139, 1997.

[LDWC91] J. Leonard, H. F. Durrant-Whyte and I. J. Cox, Dynamic Map Building for an Autonomous Mobile Robot, in *Autonomous Mobile Robots*, S. S. Iyengar and A. Elfes, eds., pp. 331–338, IEEE Computer Society Press, 1991.

[Lip82] L. A. Liporace, Maximum Likelihood Estimation for Multivariate Observations of Markov Sources, *IEEE Transactions on Information Theory*, 28 (5), 1982.

[Lit96] M. L. Littman, *Algorithms for Sequencial Decision Making*, Ph.D. thesis, Department of Computer Science, Brown University, Providence, RI, 1996.

[LM97] F. Lu and E. E. Millios, Globally Consistent Range Scan Alignment for Environment Mapping, *Autonomous Robots*, 4, pp. 333–349, 1997.

[LRS83] S. E. Levinson, L. R. Rabiner and M. M. Sondhi, An Introduction to the Application of the Theory of Probabilistic Functions of a Markov Process to Automatic Speech Recognition, *The Bell System Technical Journal*, 62 (4), pp. 1035–1074, 1983.

[Mar67] J. J. Martin, *Bayesian Decision Problems and Markov Chains*, no. 13 in Publications in Operations Research, John Wiley & Sons, 1967.

[Mar72] K. V. Mardia, *Statistics of Directional Data*, Academic Press, 1972.

[MB98] N. Meuleau and P. Bourgine, Exploration of Multi-States Environments: Local Measurements and Back-Propagation of Uncertainty, *Machine Learning*, 1998, (To appear).

[ME85] H. P. Moravec and A. Elfes, High Resolution Maps from Wide Angle Sonar, in *Proceedings of the International Conference on Robotics and Automation*, pp. 116–121, 1985.

[MK97] G. J. McLachlan and T. Krishnan, *The EM Algorithm and Extensions*, John Wiley & Sons, 1997.

[Mor88] H. P. Moravec, Sensor Fusion in Certainty Grids for Mobile Robots, *AI Magazine*, 9 (2), pp. 61–74, 1988.

[NPB95] I. Nourbakhsh, R. Powers and S. Birchfield, DERVISH: An Office-Navigating Robot, *AI Magazine*, 16 (1), pp. 53–60, 1995.

[PK97] D. Pierce and B. Kuipers, Map Learning with Uninterpreted Sensors and Effectors, *Artificial Intelligence*, 92 (1-2), pp. 169–227, 1997.

[Put94] M. L. Puterman, *Markov Decision Processes*, John Wiley & Sons, 1994.

[PW89] L. Pitt and M. K. Warmuth, The Minimum Consistent DFA Problem Cannot be Approximated within any Polynomial, in *Proceedings of the Twenty First Annual Symposium on Theory of Computing*, pp. 421–432, Seattle, Washington, 1989.

[Rab89] L. R. Rabiner, A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, *Proceedings of the IEEE*, 77 (2), pp. 257–285, 1989.

[RJ93] L. Rabiner and B.-H. Juang, *Fundamentals of Speech Recognition*, Prentice-Hall, 1993.

[RJL$^+$85] L. R. Rabiner, B. H. Juang, S. E. Levinson and M. M. Sondhi, Some Properties of Continuous Hidden Markov Model Representation, *AT&T Technical Journal*, 64 (6), pp. 1251–1269, 1985.

[RS87a] R. L. Rivest and R. E. Schapire, Diversity Based Inference of Finite Automata, in *Proceedings of the IEEE Twenty Eighth Annual Symposium on Foundations of Computer Science*, pp. 78–87, Los Angeles, California, 1987.

[RS87b] R. L. Rivest and R. E. Schapire, A New Approach to Unsupervised Learning in Deterministic Environments, in *Proceedings of the Fourth International Workshop on Machine Learning*, pp. 364–375, University of California, Irvine, 1987.

[RS89] R. L. Rivest and R. E. Schapire, Inference of Finite Automata Using Homing Sequences, in *Proceedings of the Twenty First Annual Symposium on Theory of Computing*, pp. 411–420, Seattle, Washington, 1989.

[RST94] D. Ron, Y. Singer and N. Tishbi, Learning Probabilistic Automata with Variable Memory Length, in *Proceedings of the Seventh Annual Workshop on Computational Learning Theory*, pp. 35–46, 1994.

[RST95] D. Ron, Y. Singer and N. Tishbi, On the Learnability and Usage of Acyclic Probabilistic Finite Automata, in *Proceedings of the Eighth Annual Workshop on Computational Learning Theory*, pp. 31–40, 1995.

[RST98] D. Ron, Y. Singer and N. Tishby, On the Learnability and Usage of Acyclic Probabilistic Finite Automata, *Journal of Computer and Systems Science*, 56 (2), 1998.

[Saa95] Y. Saad, *Iterative Methods for Sparse Linear Systems*, pp. 33–37, John Wiley and Sons, 1995.

[SDL$^+$93] D. J. Spiegelhalter, A. P. Dawid, S. L. Lauritzen and R. G. Cowell, Bayesian Analysis in Expert Systems, *Statistical Science*, 8 (3), pp. 219–283, 1993.

[Sha93] J. Shao, Linear Model Selection by Cross-Validation, *Journal of the American Statistical Association*, 88 (422), pp. 486–494, 1993.

[SIK86] Y. Sakamoto, M. Ishiguro and G. Kitagawa, *Akaike Information Criterion Statistics*, D. Reidel Publishing Company, 1986.

[SK95] R. G. Simmons and S. Koenig, Probabilistic Navigation in Partially Observable Environments, in *Proceedings of the International Joint Conference on Artificial Intelligence*, 1995.

[SK97a] H. Shatkay and L. P. Kaelbling, *Learning Hidden Markov Models with Geometric Information*, Tech. Rep. CS-97-04, Department of Computer Science, Brown University, 1997.

[SK97b] H. Shatkay and L. P. Kaelbling, Learning Topological Maps with Weak Local Odometric Information, in *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, Nagoya, Japan, 1997.

[SK98] H. Shatkay and L. P. Kaelbling, Heading in the Right Direction, in *Proceedings of the Fifteenth International Conference on Machine Learning*, Madison, Wisconsin, 1998.

[SOA97] M. Suojanen, K. G. Olesen and S. Andreassen, A Method for Diagnosing in Large Medical Expert Systems Based on Causal Probabilistic Networks, in *Proceedings of the Sixth Conference on Artificial Intelligence in Medicine Europe*, pp. 285–295, Grenoble,France, 1997.

[SSC91] R. Smith, M. Self and P. Cheeseman, A Stochstic Map for Uncertain Spatial Relationships, in *Autonomous Mobile Robots*, S. S. Iyengar and A. Elfes, eds., pp. 323–330, IEEE Computer Society Press, 1991.

[Sto74] M. Stone, Cross-validatory Choice and Assessment of Statistical Predictions, *Journal of the Royal Statistical Society, B*, 36 (2), pp. 111–147, 1974.

[Sut90] R. S. Sutton, Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming, in *Proceedings of the Seventh International Conference on Machine Learning*, pp. 216–224, 1990.

[TB96a] S. Thrun and A. Bücken, Integrating Grid-Based and Topological Maps for Mobile Robot Navigation, in *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pp. 944–950, 1996.

[TB96b] S. Thrun and A. Bücken, *Learning Maps for Indoor Mobile Robot Navigation*, Tech. Rep. CMU-CS-96-121, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1996.

[TBF98] S. Thrun, W. Burgard and D. Fox, A Probabilistic Approach to Concurrent Map Acquisition and Localization for Mobile Robots, *Machine Learning*, 31, pp. 29–53, 1998.

[TGF$^+$98] S. Thrun, J.-S. Gutmann, D. Fox, W. Bugard and B. J. Kuipers, Integrating Topological and Metric Maps for Mobile Robot Navigation: A Statistical Approach, in *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pp. 989–995, 1998.

[Thr92] S. Thrun, The Role of Exploration in Learning Control, in *Handbook of Intelligent Control: Neural, Fuzzy and Adptive Approaches*, D. A. White and D. A. Sofge, eds., Van Nostrand Reinhold, Florence, Kentucky, 1992.

[Thr99] S. Thrun, Learning Metric-Topological Maps for Indoor Mobile Robot Navigation, *AI Journal*, 1, pp. 21–71, 1999.

[Upt73] G. J. G. Upton, Single-Sample Tests for the von Mises Distribution, *Biometrika*, 60 (1), pp. 87–99, 1973.

[Vap95] V. N. Vapnik, *The Nature of Statistical Learning Theory*, Springer, 1995.

[Wu83] C. F. J. Wu, On the Convergence Properties of the EM Algorithm, *The Annals of Statistics*, 11 (1), pp. 95–103, 1983.