



“SAPIENZA” UNIVERSITÀ DI ROMA

DOTTORATO DI RICERCA IN INGEGNERIA INFORMATICA

XXI CICLO – 2008

Looking Inside for Mapping the Outside:  
Introspective Simultaneous Localization and  
Mapping

Gian Diego Tipaldi





“SAPIENZA” UNIVERSITÀ DI ROMA

DOTTORATO DI RICERCA IN INGEGNERIA INFORMATICA

XXI CICLO - 2008

Looking Inside for Mapping the Outside:  
Introspective Simultaneous Localization and  
Mapping

Gian Diego Tipaldi

Thesis Committee

Prof. Daniele Nardi (Advisor)  
Prof. Camil Demetrescu

Reviewers

Prof. Wolfram Burgard  
Prof. Paul Newman

Copyright © 2008  
by Gian Diego Tipaldi

ISBN: 1234567890

AUTHOR'S ADDRESS:

Gian Diego Tipaldi

Dipartimento di Informatica e Sistemistica "Antonio Ruberti"

Università di Roma "Sapienza"

Via Ariosto 25, I-00185 Roma, Italy.

E-MAIL: [tipaldi@dis.uniroma1.it](mailto:tipaldi@dis.uniroma1.it)

WWW: <http://www.dis.uniroma1.it/~tipaldi/>

*A nonno Luigi e nonna Adelina*



# Abstract

This thesis focuses on the development and improvement of Simultaneous Localization and Mapping algorithms. In particular, we are interested in reliable and effective techniques for unstructured large scale environments. While previous researcher focused on the analysis of class of algorithms or general effects of different algorithms (e.g. complexity, convergence or consistency), in this thesis we focus on analyzing the mapping process itself.

At first, we present an introspection analysis that allows efficient optimizations for Rao-Blackwellized SLAM on grid maps. The key idea is based on an analysis of the mapping process which allows us to perform filter updates *conditioned* to the state of the mapping system: *localization, mapping* or *loop closing*. We are able to update the complex posterior with substantially less resources by performing the computations only for a set of representatives instead of for all potential hypotheses.

Extending this introspective analysis from a filter perspective to a general one, we find out that the SLAM problem can be decomposed in three main subproblems: a) incremental mapping, that is the process of providing local constraints between consecutive poses, in order to maintain the map locally consistent; b) loop closure: that is the process of finding global constraints among different parts of the map; c) map optimization: that is the process of combining local constraints (provided by the incremental mapper) and global constraints (provided by the loop-closure algorithm) to obtain an overall consistent map. We show solution to this three problems when using a laser range finder in both static and dynamic environments.





# Acknowledgments

It was a pleasure for me to work with all the people here in the lab in Rome. First of all I would like to thank my advisor, Daniele Nardi, for giving me the opportunity to develop my thesis and the idea of introspection as well. I want to thank him also for teaching me how to write scientific papers and for the opportunity to visit several labs during the years of my PhD.

In particular I want to thank my friends and colleagues in the lab and in the department for the fruitful discussions and happy time we had together: Luca Iocchi, Vittorio Amos Ziparo, Alessandro Farinelli, Giorgio Grisetti, Pier Francesco Palamara, Stefano Pellegrini, Daniele Calisi, Giuseppe Settembre, Luca Marchetti, Matteo Leonetti, Francesca Giannone, Andrea Cherubini, Simone Elviretti, Andrea Censi, Alessio Pascucci, Anna Belardinelli, Andrea Carbone, Matia Pizzoli, Gabriele Randelli, Vincenzo Bonifaci, Fabio Patrizi. Thanks for the nice moments we shared together in the department and the countless coffe breaks.

I would also thank Wolfram Burgard for giving me the possibility to work in his lab in Freiburg for six months. In particular I want to thank my colleagues Patrick Pfaff, Christian Plagemann, Giorgio Grisetti, Kai Oliver Arrass, Cyrill “Cirillo” Stachniss, Maren Bennewitz, Janna, Barbara Frank, Slawomir Grzonka, Bastian Staeder, Rainer Kümmerle, Daniel Meyer-Delius, Axel Rottmann, Boris Lau, Matthias Lubert, Hauke Strasdat, Oscar Martinez Mozos, Jurgen Sturm. I also want to thank my friends in Freiburg for the nice time they spent with me: Alex Kleiner, Gabi Röger, Malte Helmert, Sebastian “Waschdl” Kupferschmid, Fang Wei, Sarah Finkel, Daniel Bär.

I want to thank Dieter Fox for letting me be a visiting student at the University of Washington. He gave me fruitful ideas for dealing with moving objects. I also want to thank the people in Seattle that supported me: Jonathan Ko, Brian Ferris, Subhanshu Gupta, Mrinal Murari, Nadja Douglas, Cecil Bibi Kabagema, Kanti Mani, Sudip Shenkar, Aldo Compagnoni, Maurizio Di Pierro, Silje Soerland, Martha Alem, Meghan Vita, Cimer Ergen, Urooj Khan, Katia Kutavina, Angeli Rawat.

I wish to thank the external reviewers, Paul Newman and Wolfram Burgard for spending a lot of time in patiently reviewing this document. I really believe my work has been consistently strengthen thanks to their comments.

The experiments in this thesis have been made by using a wide range of third party logs. I would acknowledge all the people who made available their data: Dirk Haehnel, Cyrill Stachniss, Hendrik Andreasson, Tom Duckett, Mike Bosse, Dieter Fox, Mike Montemerlo, Nik Roy, Ben Kuipers, Patrick Beeson, Andrew Howard and many others.

Last but not least, I want to thank my family, my friends in Rome and my friends in San Nazzaro. Thanks to you all for allowing me to relax a bit between an equation and the other...



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Contributions	3
1.2	Publications	4
1.3	Collaborations	4
<b>2</b>	<b>Preliminaries</b>	<b>5</b>
2.1	Bayesian State Estimation	5
2.1.1	Optimal Bayes Filter	6
2.1.2	Kalman Filter	7
2.1.3	Information Filter	8
2.1.4	Particle Filters	11
2.2	Grid Mapping	17
2.2.1	Occupancy Probability Mapping	18
2.2.2	Reflection Probability Mapping	19
<b>3</b>	<b>A Taxonomy of Simultaneous Localization and Mapping</b>	<b>21</b>
3.1	Introduction	21
3.2	Map Representations	22
3.2.1	Landmark Maps	22
3.2.2	Grid Maps	24
3.2.3	Topological Maps	25
3.2.4	Hybrid Maps	26
3.3	Techniques	27
3.3.1	Incremental Maximum Likelihood	28
3.3.2	Optimization Based Techniques	31
3.3.3	Expectation Maximization	33
3.3.4	Kalman Filter	34
3.3.5	Information Filter	35
3.3.6	Rao-Blackwellized Particle Filters	36
3.3.7	Topological Approaches	36
3.3.8	Hybrid Approaches	37
3.4	Discussion	39
3.4.1	Recovery From Failure	39
3.4.2	Dynamic Environments	40

## **I Introspective Analysis of Filtering Solutions to Simultaneous Localization and Mapping** **43**

<b>4 Rao-Blackwellized Mapping</b>	<b>45</b>
4.1 Rao-Blackwellized Particle Filter for Simultaneous Localization and Mapping	45
4.2 Specification of Filter Components	47
4.2.1 Proposal Distribution	47
4.2.2 Resampling Strategies	49
4.2.3 Representation	52
4.3 Discussion	52
<b>5 Introspective Filter</b>	<b>55</b>
5.1 Introduction	55
5.2 Introspective Analysis of Rao-Blackwellized Mapping	56
5.3 Map Representation	59
5.3.1 Basic Operations	61
5.4 Situation Based Rao-Blackwellized Particle Filters	62
5.4.1 Exploration	63
5.4.2 Localization	66
5.4.3 Loop Closing	68
5.5 Situation Assessment	71
5.6 Overall Algorithm	73
5.7 Experiments	79
5.7.1 Number of Particles	80
5.7.2 Mapping Experiments	81
5.7.3 Performances	81
5.8 Connections with Previous Works	84
5.9 Conclusions	85

## **II Introspective Decomposition of Simultaneous Localization and Mapping** **87**

<b>6 Motion Clustering and Incremental Estimation</b>	<b>89</b>
6.1 Introduction	89
6.2 Conditional Random Fields	90
6.2.1 Inference	91
6.2.2 Pseudo-Likelihood Parameter Learning	91
6.3 Motion Estimation in Static Environment	92
6.3.1 Covariance Estimation	94
6.4 Motion Clustering and Estimation	95
6.4.1 Model Definition	95
6.4.2 Local Feature	96
6.4.3 Pairwise Features	97
6.4.4 Inference Procedure	98
6.4.5 Computing the Number of Clusters	98
6.5 Experimental Evaluation	100

6.5.1	Convergence Properties	100
6.5.2	Cluster Evaluation Measure	100
6.5.3	Comparison with Consistency-based Detection	102
6.5.4	Comparison with Modified K-Means	103
6.6	Connection With Previous Works	105
6.7	Conclusions	107
<b>7</b>	<b>Loop Closing as Lazy Localization</b>	<b>109</b>
7.1	Introduction	109
7.2	Global and Lazy Localization	110
7.3	Particle Filter Approaches	112
7.3.1	Approximate Particle Smoothing	112
7.3.2	Improving Approximation by using a Scan Matcher	114
7.4	The Frozen-Time Smoother	115
7.4.1	Translated Likelihood Computation by GHT	116
7.5	Discussion	118
7.5.1	On the Use of a Grid	118
7.5.2	On the Assumption of a Precise Incremental Estimate	119
7.5.3	On the Laziness of FTS	119
7.5.4	On the Probability Distribution	120
7.6	Experiments	120
7.6.1	Connection with Previous Works	124
7.7	Conclusions	125
<b>8</b>	<b>Probabilistic Map Optimization</b>	<b>127</b>
8.1	Introduction	127
8.2	Gaussian Markov Random Field	128
8.2.1	Inference	129
8.3	SLAM as a Gaussian Markov Random Field	130
8.3.1	FullSLAM Graphical Model	130
8.3.2	Delayed State SLAM Graphical Model	131
8.4	Approximate Covariance Computation	132
8.4.1	Covariance Intersection	133
8.4.2	Loopy Intersection Propagation	134
8.4.3	Algorithm	135
8.5	Experiments	136
8.5.1	Statistical Experiments	138
8.5.2	Real World Data	139
8.6	Connections with Previous Work	141
8.7	Conclusions	142
<b>9</b>	<b>Discussion</b>	<b>145</b>
9.1	Conclusions	145
9.2	Future Works	147
9.2.1	3D Environments	147
9.2.2	Multi Hypothesis Tracker for Data Association	148
9.2.3	Simultaneous Localization, Mapping and Tracking	148

<b>III</b>	<b>Appendix</b>	<b>149</b>
<b>A</b>	<b>Conditional Random Field for Semi-Supervised Clustering</b>	<b>151</b>
A.1	Semi-Supervised Clustering . . . . .	151
A.2	CRF-Clustering . . . . .	152
A.2.1	The Model . . . . .	152
A.2.2	Generalized Constraints . . . . .	154
A.2.3	Inference . . . . .	155
<b>B</b>	<b>Multi Hypothesis Data Association</b>	<b>157</b>
B.1	Problem formulation . . . . .	157
B.2	Rao-Blackwellized Particle Filters . . . . .	158
B.3	State Estimation with RBPF . . . . .	159
B.3.1	Fixed and known number of features . . . . .	160
B.3.2	Variable and unknown number of features . . . . .	160
B.4	Experiments . . . . .	161
	<b>List of Algorithms</b>	<b>165</b>
	<b>List of Figures</b>	<b>171</b>
	<b>List of Tables</b>	<b>173</b>
	<b>Bibliography</b>	<b>187</b>

# Chapter 1

## Introduction

Autonomous mobile robots are a fascinating research challenge in current Robotics and AI research fields. The knowledge of the robot position in the environment plays an important role, when the robot has to accomplish rather complex tasks taking long time. Moreover, a representation of the environment has to be provided, in order to allow the robot to reason about its surroundings.

Usually, robots are equipped with proprioceptive sensors for estimating their relative movements. However, those information are affected by systematic errors and random noise. The integration of those effects over time yields to a growing drift between the estimated and the real pose. Exteroceptive sensors (such as stereo cameras, laser range finders, inertial measurement units, and so on...), in conjunction with a prior knowledge of the environment (a map), are used to filter out that noise and to provide better estimates.

There are situations when a prior knowledge of the map is not feasible. In these cases, exteroceptive sensors (usually providing range and/or bearing information) are used to build an environment representation. Those sensors present limited sight range, thus the robot needs to move while mapping. Unfortunately, when the robot moves, information obtained from proprioceptive sensors are used. As stated above, the error on robot location increases over time, and the resulting map accuracy decreases with the length of the traveled path. In order to avoid that, the position of the robot has to be corrected with respect to the partial map built so far. The combination of both localizing the robot and building the map is known as the Simultaneous Localization And Mapping (SLAM) problem, and represents one of the most challenged problem in autonomous mobile robotics.

Since Smith, Self and Cheeseman seminal paper [[Smith, Self, & Cheeseman, 1990](#)], researchers have been converging toward a common understanding: probabilistic estimation. They estimate the distribution over the possible maps and robot poses, given the sensor readings. Many approaches have been proposed. They differ by the underlying representation used (grids, features, etc.), and the estimation algorithm employed (Particle Filters, Extended Kalman Filters, Information Filters, etc.).

With features, one can have a nice analysis of the problem, as the world is modelled in a finite-dimensional way. Robot poses and features are random variables, and observations are constraints among those. Once modelled in this way, several interesting results can be provided. Some authors focused the attention on stability and consistency properties, mainly

showing the limitation of the Extended Kalman Filter when applied to this problem. The major problem consist in the linear assumption, which is quite valid locally but not globally. Example of this works are [Frese & Hirzinger, 2001; Newman & Leonard, 2003; Julier & Uhlmann, 2001]. Frese *et. al* [Frese & Hirzinger, 2001] showed that the linear assumption in the Kalman Filter can cause systematic errors.

Other researches analyzed the complexity of The EKF and showed some results on the sparsity of its dual: the Information Filter. An analysis of the sparsity of the covariance matrix has been also done by Frese [Frese & Hirzinger, 2001]. Moreover, one of the major problems of this kind of algorithm seems to be the computational complexity, which depends on the size of the system state space to track. Such a size is proportional to the number of landmarks  $n$ , and the filter requires  $O(n^3)$  for each update. Therefore, the use of the Information Filter for solving the SLAM problem is becoming more and more popular. The key concept of this approach is that if the covariance matrix of a system can be dense, the information matrix may be sparse, or its entries can be small. Example of this works are [Thrun *et al.*, 2004; Eustice, Walter, & Leonard, 2005; Eustice, Singh, & Leonard, 2005]. The same insight has been proposed by Thrun *et al.* [Thrun *et al.*, 2004]. In that work, the Information matrix is made sparse, by zeroing some elements. Further improvements have been made by Eustice. In a first work [Eustice, Walter, & Leonard, 2005] he pointed out some insight into matrix sparsification and convergence. In a second work he proved that a delayed state formulation of the problem yields to a sparse information matrix [Eustice, Singh, & Leonard, 2005].

A deeper analysis is carried by Dellaert [Dellaert, 2005a] showed some connections among Linear Algebra, Graph Theory and the full SLAM, when it is modelled as an undirected graphical model. Those undirected models describe the problem in terms of inter-variable relationships. These relationship express spatial constraints among the poses of the delayed state, thus resulting in good formalization for optimization techniques like [Duckett, Marsland, & Shapiro, 2002; Frese, Larsson, & Duckett, 2005; Howard, Matarić, & Sukhatme, 2001b; Lu & Milios, 1997a; Olson, Leonard, & Teller, 2006; Grisetti *et al.*, 2007a].

The SLAM problem has also been modelled with a Dynamic Bayesian Network. DBNs are powerful in expressing infinite time series, by means of an evolution and an observation model. Providing this models is straightforward in the case of slam, as they correspond to the odometry and the observation likelihood. Moreover, the use of DBN shows an interesting property of the SLAM problem:

*Given the histories of poses, the landmarks are independent from each others.*

This intuition is at the basis of a Rao-Blackwellized particle filter solution [Murphy, 1999] and the FastSLAM algorithm [Montemerlo, Koller, & Wegbreit, 2003]. The state space is factored into the robot trajectory, which is represented by particles, and the map, which can be estimated analytically. While this algorithm has been introduced for maps with features, it is later extended to grid maps by Hähnel *et al.* [Hähnel *et al.*, 2003a].

Some authors studied the complexity of this family of algorithms, both from a time and a memory point of view. As every particles has to store a complete map of the environment, the memory requirements can be far beyond actual capabilities. A memory efficient map structures have been introduced in [Eliazar & Parr, 2003].

Particle depletion [van der Merwe *et al.*, 2000] has been studied. This problem often occurs when the likelihood is highly peaked compared to the motion model, resulting in particles having low coverage of the likelihood function support. A solution to this problem is to use an improved proposal distribution, which take the actual measurement into



account [Grisetti, Stachniss, & Burgard, 2006].

However, SLAM is not yet an effective and reliable technology. The problem is very complex, with scalability representing an important issue. State estimation algorithms present poor performances in huge environments, especially in presence of significant angular errors. Optimization techniques alleviate non-linear problems, but they need a good initial guess and unique landmarks.

## 1.1 Contributions

The contributions of this thesis are based on a novel analysis of the SLAM problem. While previous researcher focused on the analysis of class of algorithms or general effects of different algorithms (e.g. complexity, convergence or consistency), in this thesis we focus on analyzing the mapping process itself. At first, we made a consistent analysis of current and past research, providing an interesting taxonomy of the SLAM approaches. A second contribution is an introspective analysis of Rao-Blackwellized particle filter solutions for the SLAM problem. Thanks to this analysis, we were able to derive an approximate filter which is one order of magnitude faster than current solutions, and requires several orders of magnitude less memory, making it possible to compute in real time even larger maps. Extending this introspective analysis from a filter perspective to a general one, we find out that the SLAM problem can be decomposed in three main subproblems

**Incremental Mapping:** The process of building an incremental map. This process provides local constraints between consecutive poses, in order to maintain the map always locally consistent.

**Loop Closure:** The process of finding global constraints among different parts of the map. When a mapping robot returns in a previously mapped area, the error accumulated is typically such that the head and the tail of the map estimate are inconsistent. Global constraint between the robot pose at closure time and a previous pose in the already mapped area are needed to obtain global consistent maps.

**Map Optimization:** The process of combining local constraints (provided by the incremental mapper) and global constraints (provided by the loop-closure algorithm) to obtain an overall consistent map.

In this thesis, we provide a solution for each of this subproblems, both in a static and dynamic environments.

This thesis is organized as follows. Bayes filtering and grid mapping are first introduced, as they will be used in the rest of the thesis. An exhaustive taxonomy of the SLAM research is provided in [Chapter 3](#).

The first part of the thesis concentrates on the introspective analysis of filtering solutions to SLAM. In [Chapter 4](#) we describe the Rao-Blackwellized framework for SLAM, describing how several aspect have been addressed by the researchers. An introspective analysis of this framework is made in [Chapter 5](#), and an approximate solution is derived.

The second part of the thesis focused on the introspective decomposition described before. In [Chapter 6](#) we describe how to compute local estimate of the robot motion in both a static and dynamic environments. A novel formulation of loop closing and a related algorithm are described in [Chapter 7](#). Finally, in [Chapter 8](#) a probabilistic framework for map optimization

is presented. This framework is able to combine local and global constraints to provide a mean and covariance estimation of the map.

## 1.2 Publications

Part of this thesis have been published in the following journal articles, conference and workshop proceedings:

- Grisetti, G.; Tipaldi, G.; Stachniss, C.; Burgard, W.; and Nardi, D. Fast and accurate slam with rao-blackwellized particle filters. *Robotics and Autonomous Systems, Special issue on Simultaneous Localization and Map Building*.
- Tipaldi, G. D.; Grisetti, G.; and Burgard, W. Approximated covariance estimation in graphical approaches to slam. In *IEEE/RSJ Intern. Conf. on Intelligent Robots and Systems (IROS)*.
- Censi, A., and Tipaldi, G. D. Lazy localization using the frozen time smoother. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.
- Grisetti, G.; Tipaldi, G. D.; Stachniss, C.; Burgard, W.; and Nardi, D. Speeding up rao blackwellized slam. In *IEEE International Conference on Robotics and Automation (ICRA)*.
- Iocchi, L.; Pellegrini, S.; and Tipaldi, G. D. Building multi-level planar maps integrating LRF, stereo vision and IMU sensors. In *Proc. of IEEE International Workshop on Safety, Security and Rescue Robotics (SSRR)*.
- Calisi, D.; Farinelli, A.; Grisetti, G.; Iocchi, L.; Nardi, D.; Pellegrini, S.; Tipaldi, G. D.; and Ziparo, V. A. Contextualization in mobile robots. *ICRA'07 Workshop on Semantic Information in Robotics*.
- Calisi, D.; Farinelli, A.; Grisetti, G.; Iocchi, L.; Nardi, D.; Pellegrini, S.; Tipaldi, D.; and Ziparo, V. A. Uses of contextual knowledge in mobile robots. In *Proc. of the 10th Congress of the Italian Association for Artificial Intelligence*.
- Tipaldi, G. D.; Farinelli, A.; Iocchi, L.; and Nardi, D. Heterogeneous feature state estimation with rao-blackwellized particle filters. In *Proc. of IEEE International Conference on Robotics and Automation (ICRA)*.

## 1.3 Collaborations

Part of this thesis have been done in collaboration with other people. The introspective filter and the analysis of the state of the art have been done together with Giorgio Grisetti. Some ideas have been presented in his PhD thesis [Grisetti, 2006]. The work on motion clustering and estimation was done in tight collaboration with Fabio Ramos and Dieter Fox, during my stay in Seattle.

## Chapter 2

# Preliminaries

This chapter introduces some basic techniques which will be used through the remaining of this thesis. First we will introduce the Bayesian framework for estimating the state of a stochastic dynamic system. We then introduce two techniques for building grid maps from range data, namely occupancy grid and reflection grid.

### 2.1 Bayesian State Estimation

In this section, we revise the Bayesian state estimation framework, focusing in particular on the *filtering problem*.

A probabilistic filter for a dynamic system is a mathematical tool, which goal is to estimate a distribution of the possible system state  $x_t$  given the input  $u_{1:t}$  and the observation  $z_{1:t}$  history

$$p(x_t | z_{1:t}, u_{1:t}) \quad (2.1)$$

Several on line and off line techniques for solving the filtering problem have been proposed [Kalman, 1960; Gordon, Salmond, & Ewing, 1993; Arumampalam *et al.*, 2001; Pitt & Shephard, 1999]. Most of them rely on the assumption that the process being observed is Markovian. A process is Markovian if the current measurement is independent from the past ones, given the current state

$$p(z_t | z_{1:t-1}, x_t) = p(z_t | x_t) \quad (2.2)$$

In the context of the SLAM problem, for the Markov assumption to hold, no moving objects unknown to the robot can populate the environment. This imposes obvious restrictions to the application domains. However, in moderately dynamic environments most of the techniques proposed in this section have shown to work. In case of big violations of the Markov assumption, a typical approach consists in pre-processing the filter input in order to skip the sensor readings generated by dynamic objects.

In the rest of this chapter we describe a wide range of useful tools for Bayes filtering. In particular we will describe:

- The optimal Bayes filter (BF), which represent the exact Bayesian inference but is usually not applicable in practice due to the intractability of the required integrals;
- The Kalman filter (KF), which is an exact, closed form filter working with linear systems, affected by zero mean Gaussian noise;
- The Information filter (IF), which is the counterpart of the Kalman Filter exploiting a different parametrization of the Gaussian distribution;
- The Particle filter (PF), which is a Monte Carlo method suitable for the state estimation of non linear non Gaussian dynamic systems.

### 2.1.1 Optimal Bayes Filter

Let  $p(z|x)$  be the *observation model*, that is the density of the measurement  $z$ , given that the system state is  $x$ , and let  $p(x_t|x_{t-1})$  be the *evolution model*<sup>1</sup>. If the Markov assumption holds, the posterior of the state chain up to time  $t$  is

$$\begin{aligned}
p(x_{1:t}|z_{1:t}) &= \frac{p(z_t|x_{1:t}, z_{1:t-1})p(x_{1:t}|z_{1:t-1})}{p(z_t|z_{1:t-1})} \quad [\text{by Markov assumption}] \\
&= \frac{p(z_t|x_t)p(x_{1:t}|z_{1:t-1})}{p(z_t|z_{1:t-1})} \\
&= \frac{p(z_t|x_t)p(x_t|x_{t-1})}{p(z_t|z_{1:t-1})} p(x_{1:t-1}|z_{1:t-1}) \tag{2.3}
\end{aligned}$$

If one is interested in estimating the current state distribution, the filtering equation becomes the following:

$$\begin{aligned}
p(x_t|z_{1:t}) &= \frac{p(z_t|x_t)p(x_t|z_{1:t-1})}{p(z_t|z_{1:t-1})} \\
&= \frac{p(z_t|x_t) \int p(x_t|x_{t-1})p(x_{t-1}|z_{1:t-1})dx_{t-1}}{p(z_t|z_{1:t-1})} \\
&= \frac{p(z_t|x_t) \int p(x_t|x_{t-1})p(x_{t-1}|z_{1:t-1})dx_{t-1}}{\int p(z_t|z_{1:t-1}, x_t)p(x_t|z_{1:t-1})dx_t} \\
&= \eta p(z_t|x_t) \int p(x_t|x_{t-1})p(x_{t-1}|z_{1:t-1})dx_{t-1} \tag{2.4}
\end{aligned}$$

here  $\eta$  is a normalization factor ensuring that [Equation 2.4](#) correctly represents a probability distribution.

Usually, the evaluation of [Equation 2.4](#) is done in two steps: **prediction** and **update**. In the prediction step, the result of the state transition from  $x_{t-1}$  to  $x_t$  is computed. In the update step, the last observation  $z_t$  is incorporated in the previously computed probability

<sup>1</sup> In the following sections, the evolution model is also referred to as *motion model*, since it is used for describing the change of the robot state after motion. In the SLAM context the state transitions are governed by the proprioceptive sensings  $u_{0:t-1}$ . For clarity of notation, but with loss of generality, in this chapter we omit the terms  $u_{0:t-1}$  which affect the transition model. This is equivalent to consider a time dependent transition model. In practice  $p(x_t|x_{t-1}, u_{t-1}) = p_t(x_t|x_{t-1})$ .

density. Referring to [Equation 2.4](#), one can argue that the predict step consists in computing the integral term. The update step is performed by weighting the predicted belief  $\int p(x_t|x_{t-1})p(x_{t-1}|z_{1:t-1})dx_{t-1}$  with the last observation likelihood  $p(z_t|x_t)$ . These two steps can be found in all of the filters described in the remainder of this chapter.

Bayes filtering in this form is exact and can be used on any kind of system for which the Markov assumption holds. Unfortunately, in the above equations there are some integrations over the state space. In many cases the state space is high dimensional, and the Bayes filtering cannot be directly implemented. For instance, in the SLAM problem the dimension of the system state is the sum of the robot location dimension and the map space dimension, which can easily be in the order of the hundreds or thousands. A straightforward evaluation of [Equation 2.4](#) would require an integration over the entire state space. For this reason approximated techniques are needed.

### 2.1.2 Kalman Filter

The Kalman filter (KF) [[Kalman, 1960](#); [Welch & Bishop, 2001](#)] is an exact filter that can be derived directly by [Equation 2.4](#), under the assumptions that the system is linear and the noise is Gaussian. Under these linearity hypotheses the system can be described by

$$\begin{aligned}x_t &= F_t x_{t-1} + w_t \\z_t &= H_t x_t + v_t\end{aligned}$$

The system noise  $w_t \sim \mathcal{N}(0, \Sigma_{w_t})$  and the observation noise  $v_t \sim \mathcal{N}(0, \Sigma_{v_t})$  are zero mean normally distributed. The key advantage of the Kalman Filter is that it represents the distributions in closed form, in terms of means and covariance matrix. The update of the Kalman filter can be carried out in the time of a matrix multiplication ( $O(n^3)$ , where  $n$  is the state dimension).

The iterative algorithm of the filter is the following:

- predict:

$$x'_t = F_t x_{t-1} \quad \Sigma'_t = F_t \Sigma_{t-1} F_t^T + \Sigma_{w_t}$$

- update:

$$\begin{aligned}K_t &= \Sigma'_t H_t (H_t \Sigma'_t H_t^T + \Sigma_{v_t})^{-1} \\x_t &= x'_t + K_t (z_t - H_t x'_t) \quad \Sigma_t = (I - K_t H_t) \Sigma'_t\end{aligned}$$

Unfortunately, in the mobile robot domain, the evolution model, as well as the observation model are non linear, thus the noise cannot be considered Gaussian. However, for mild evolution laws, a non linear extension can be used: the Extended Kalman Filter (EKF) [[Welch & Bishop, 2001](#)], in which local linearization of the state transition function  $f$  and the observation model  $h$  are performed. The extended Kalman filter algorithm can be expressed as

- predict:

$$x'_t = f_t(x_t) \quad \Sigma'_t = F_t \Sigma_{t-1} F_t^T + \Sigma_{w_t}$$

- update:

$$\begin{aligned}K_t &= \Sigma'_t H_t (H_t \Sigma'_t H_t^T + \Sigma_{v_t})^{-1} \\x_t &= x'_t + K_t (z_t - h_t(x'_t)) \quad \Sigma_t = (I - K_t H_t) \Sigma'_t\end{aligned}$$

here  $F_t = \nabla_x f_t|_{x_t}$  and  $H_t = \nabla_x h_t|_{x_t}$ .

The key limitations in the use of extended Kalman filter lies in the strong assumptions that have to be done on the estimated system, namely: Gaussian noise, and linearizability. In practical situations, mild violations to the above assumptions lead only to a loss of optimality. However, in most of the robotic systems used for localization and SLAM, the uncertainty is not expressible nor approximable as a Gaussian distribution, being multi modal and irregularly shaped. When more modes are present in a distribution, dealing with multiple hypotheses is needed, while the Kalman Filter works on their mean. In such a situations, its use is prone to failure. Moreover, the linearization of the system can introduce some systematic error in the estimate.

Finally, some systems cannot be linearized (being the 1st order derivatives of the state transition function null), thus the extended Kalman Filter cannot be applied. In these contexts the Unscented Kalman Filter (UKF) [Julier, Uhlmann, & Durrant-Whyte, 1995; van der Merwe *et al.*, 2000] can be used. The key difference among EKF and UKF lies in the filter update step. The first computes a local linearization of the transition function around the current mean estimate, and uses this local linearization for computing the predicted distribution. The second systematically selects a set of sampling points ( $\sigma$ -points) in the state space, around the mean. The  $\sigma$ -points are selected along the eigenvectors of the covariance matrix. Subsequently, the  $\sigma$ -points are translated according to the transition function, and their translation is used for computing the predicted mean and covariance. While the UKF in general behaves better than the Kalman filter the hypothesis of Gaussian noise is still required to hold.

Despite the above outlined limitations, the Kalman Filter is one of the most used tools in localization and SLAM, due to its simplicity. Moreover, when the underlying hypotheses hold, it exhibits a strong convergence rate if compared with other filtering techniques.

### 2.1.3 Information Filter

The information filter, or inverse covariance filter is the dual of the Kalman filter for linear Gaussian systems. In the information filter, the covariance matrix and state vector are replaced by the information or precision matrix and information vector, as the canonical representation of the Gaussian is used.

The multivariate Gaussian density function can be expressed in two complementary parametrization. The first and more used one is the *moment parametrization*

$$p(x) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} \exp \left\{ -\frac{1}{2} (\mu - x)^T \Sigma^{-1} (\mu - x) \right\} \quad (2.5)$$

where  $\mu$  and  $\Sigma$  are the parameters and  $n$  is the dimensionality. The name comes from the fact that the parameters reflects the first and second moment of the distribution

$$\mathbb{E}(x) = \mu \quad (2.6)$$

$$Cov(x) = \Sigma \quad (2.7)$$

Another way to represent the Gaussian distribution is through the *canonical parametriza-*

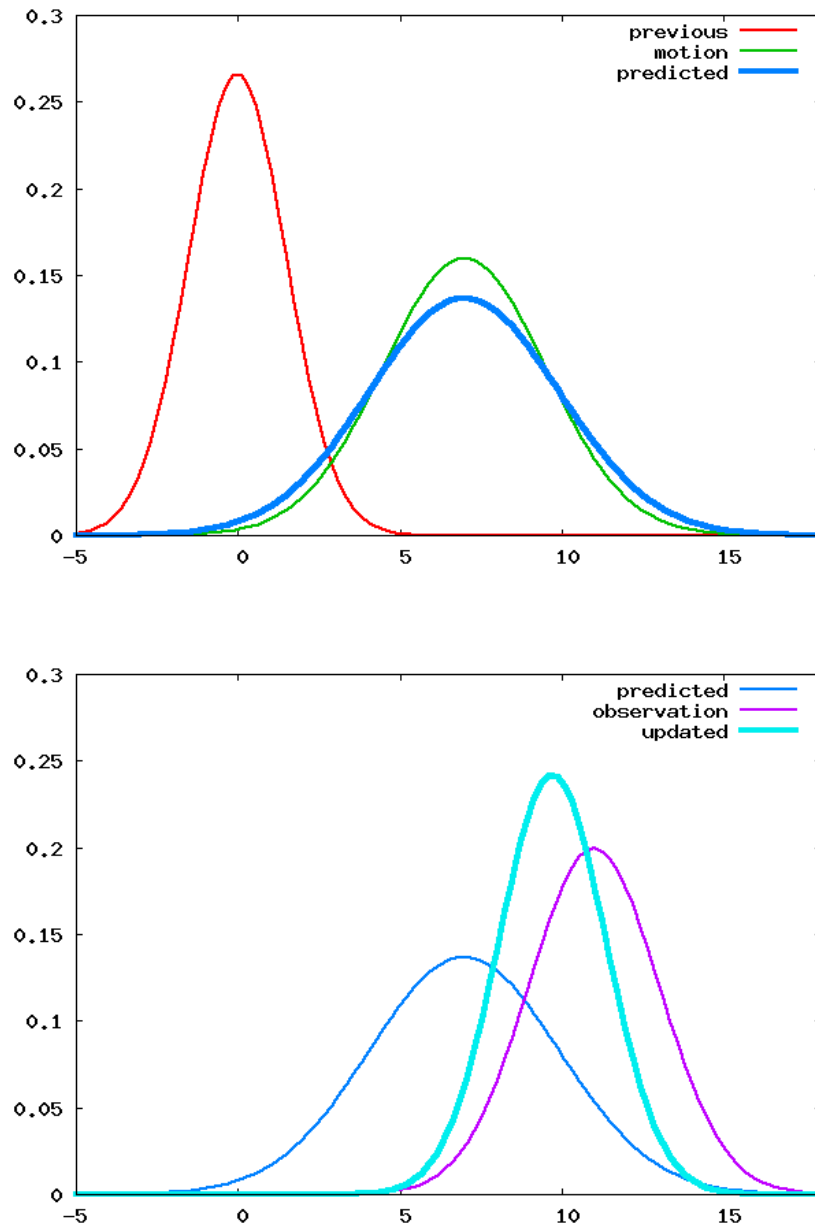


Figure 2.1: Kalman filter. The predict (top) and update (bottom) phases of the Kalman filter in the onedimensional case. the thick line in both graphs represent the estimation on the state and covariance for the prediction and update equation.

tion

$$p(x) = \exp \left\{ a + \eta^T x - \frac{1}{2} x^T \Lambda x \right\} \quad (2.8)$$

$$a = -\frac{1}{2} (n \log(2\pi) - \log |\Lambda| + \eta^T \Lambda^{-1} \eta) \quad (2.9)$$

where  $\Lambda = \Sigma^{-1}$  is the *information matrix* and  $\eta = \Sigma^{-1} \mu$  is the *information vector*.

The duality of the parametrization arise when performing standard operations like conditioning and marginalizing. Let  $x$  be a multivariate Gaussian random variable. Consider the following partition of  $x$

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (2.10)$$

where  $x_1$  and  $x_2$  are still multivariate Gaussian random variables. If  $x \sim \mathcal{N}(\mu, \Sigma)$  where

$$\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \quad (2.11)$$

$$\Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix} \quad (2.12)$$

$$(2.13)$$

then  $x_1 \sim \mathcal{N}(\mu_1^m, \Sigma_1^m)$  with

$$\mu_1^m = \mu_1 \quad (2.14)$$

$$\Sigma_1^m = \Sigma_{11} \quad (2.15)$$

If instead  $x \sim \mathcal{N}^{-1}(\eta, \Lambda)$  where

$$\eta = \begin{bmatrix} \eta_1 \\ \eta_2 \end{bmatrix} \quad (2.16)$$

$$\Lambda = \begin{bmatrix} \Lambda_{11} & \Lambda_{12} \\ \Lambda_{21} & \Lambda_{22} \end{bmatrix} \quad (2.17)$$

$$(2.18)$$

then  $x_1 \sim \mathcal{N}^{-1}(\eta_1^m, \Lambda_1^m)$  with

$$\eta_1^m = \eta_1 - \Lambda_{12} \Lambda_{22}^{-1} \eta_2 \quad (2.19)$$

$$\Lambda_1^m = \Lambda_{11} - \Lambda_{12} \Lambda_{22}^{-1} \Lambda_{21} \quad (2.20)$$

Thus, marginalizing is easier in the moment parametrization, as the operation is simply performed selecting some sub blocks of the parameters. The situation is reversed when we want to condition the distribution on  $x_2$  instead of marginalize. If  $x \sim \mathcal{N}(\mu, \Sigma)$  we have that  $x_{1|2} \sim \mathcal{N}(\mu_{1|2}^c, \Sigma_{1|2}^c)$ , with

$$\mu_{1|2}^c = \mu_1 - \Sigma_{12} \Sigma_{22}^{-1} (x_2 - \mu_2) \quad (2.21)$$

$$\Sigma_{1|2}^c = \Sigma_{11} - \Sigma_{12} \Sigma_{22}^{-1} \Sigma_{21} \quad (2.22)$$



if instead  $x \sim \mathcal{N}^{-1}(\eta, \Lambda)$  we have that  $x_{1|2} \sim \mathcal{N}^{-1}(\eta_{1|2}^c, \Lambda_{1|2}^c)$  with

$$\eta_{1|2}^c = \eta_1 - \Lambda_{12}x_2 \quad (2.23)$$

$$\Lambda_{1|2}^c = \Lambda_{11} \quad (2.24)$$

When the canonical parametrization is used, the Kalman filter becomes the information filter and has the following iterative structure and equations

- predict:

$$\eta'_t = L_t[F_t^{-1}]^T \eta_{t-1} \quad \Lambda'_t = L_t M_t L_t^T + C_t \Sigma_{\omega_t}^{-1} C_t^T$$

$$M_t = [F_t^{-1}]^T \Lambda_{t-1} F_t^{-1} \quad (2.25)$$

$$C_t = M_t [M_t + \Sigma_{\omega_t}^{-1}]^{-1} \quad (2.26)$$

$$L_t = I - C_t \quad (2.27)$$

- update:

$$\eta_t = \eta'_t + H_t \Sigma_{v_t}^{-1} z_t \quad \Lambda_t = \Lambda'_t + H_t \Sigma_{v_t}^{-1} H_t^T$$

The Extended Information Filter uses the same ideas behind the Extended Kalman Filter. The matrices  $F$  and  $H$  are replaced by first order Jacobian,  $F_t = \nabla_x f_t|_{x_t}$  and  $H_t = \nabla_x h_t|_{x_t}$ . Another approach, maybe the most used one, is to use the equation of the Kalman Filter update and then use some inversion theorem to derive a special case for updating the Information Filter.

The main advantage of the information filter is that  $N$  measurements can be filtered at each time step simply by summing their information matrices and vectors. Moreover, while the Kalman filter stores into the Covariance matrix all the information relatives, causing it to become dense, the Information Filter just stores within the Information Matrix only the direct information, keeping a sparse structure. This sparsity is the key of some efficient implementation of the information filter for SLAM.

### 2.1.4 Particle Filters

A particle filter is a non parametric implementation of a Bayes filter used to estimate the state of a non-linear non-Gaussian dynamic system. The state distribution is represented by a set  $S$  of  $N$  weighted samples, called particles

$$S = \{\langle x^{(i)}, w^{(i)} \rangle | i = 1, \dots, N\} \quad (2.28)$$

The distribution is then approximated by a weighted sum

$$p(x) \simeq \sum_i w^{(i)} \delta_{x^{(i)}}(x). \quad (2.29)$$

where  $\delta_{x^{(i)}}(x)$  is the impulse function centered in  $x^{(i)}$ . The denser are the samples  $x^{(i)}$  in a region, the higher is the probability that the current state falls within that region.

Ideally, if we want to estimate the state of a dynamic system given its observation, we would like to draw this samples from the posterior distribution  $p(x_{1:t}|z_{1:t})$ . Such a distribution is in general not available in a form suitable for sampling. However, the **Importance Sampling** (IS) principle ensures that if one can

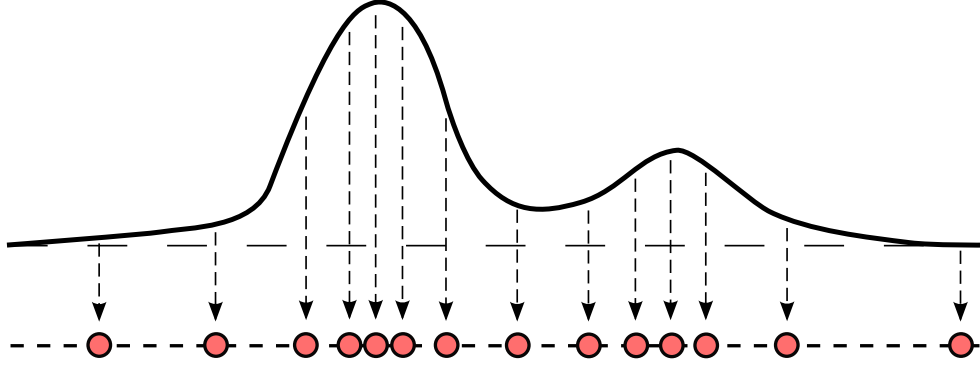


Figure 2.2: Sampling from a proposal distribution. The picture shows a proposal distribution and the samples drawn from it. The more dense are the samples in a region, the higher is the probability density in that region. We compute the probability mass falling in an interval  $\Delta p$  by summing the weights of the samples falling in the interval.

- evaluate point wise and draw samples from an arbitrarily chosen *importance density* function  $\pi(x_{1:t} | z_{1:t})$ , such that  $p(x_{1:t} | z_{1:t}) > 0 \Rightarrow \pi(x_{1:t} | z_{1:t}) > 0$ , and
- evaluate point wise  $p(x_{1:t} | z_{1:t})$ ,

then it is possible to recover the sampled approximation of  $p(x_{1:t} | z_{1:t})$  by computing the *importance weights*

$$\hat{p}(x_{1:t} | z_{1:t}) \propto \sum_i w^{(i)} \delta_{x_{1:t}^{(i)}}(x_{1:t}). \quad (2.30)$$

Here  $\{x_{1:t}^{(i)}\}$  are samples drawn from  $\pi(x_{1:t} | z_{1:t})$  and  $w_t^{(i)} = \frac{p(x_{1:t}^{(i)} | z_{1:t})}{\pi(x_{1:t}^{(i)} | z_{1:t})}$  is the importance weight related to the  $i$ th sample that takes into account the mismatch among the target distribution  $p(x_t | z_{1:t})$  and the importance function. Observe that, in case we are able to draw samples from the target distribution, such that  $p(x_{1:t}^{(i)} | z_{1:t}) \propto \pi(x_{1:t}^{(i)} | z_{1:t})$  then all of the weights are the same, and the variance of  $w^{(i)}$  is 0. An intuitive explanation of how the importance sampling principle works is given in [Figure 2.2](#) and [Figure 2.3](#).

### Sequential Importance Sampling

The Sequential Importance Sampling (SIS) algorithm is a Monte Carlo method that forms the basis for most of the particle filter algorithms. By restricting to the set of Markovian systems, and in particular focusing the choice on a particular class of importance functions, such that

$$\pi(x_{1:t} | z_{1:t}) = \pi(x_t | x_{1:t-1}, z_{1:t}) \pi(x_{1:t-1} | z_{1:t-1}) \quad (2.31)$$

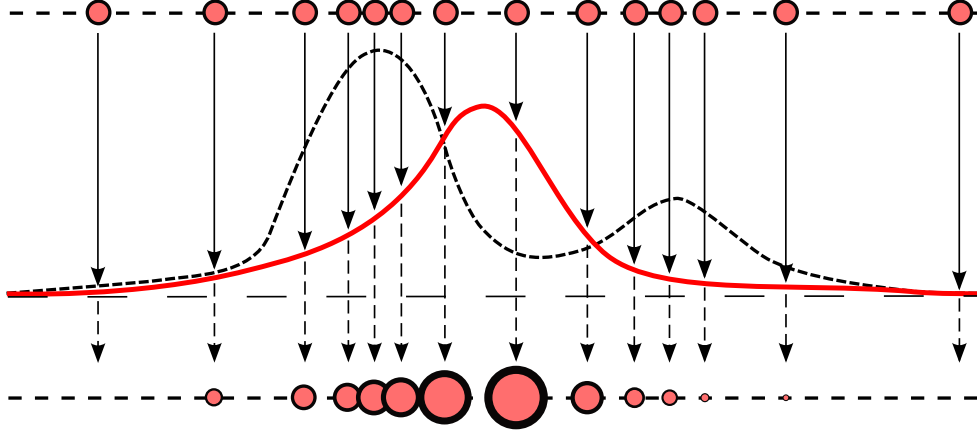


Figure 2.3: The Importance Sampling principle. The picture shows the samples, weighted according to the Importance Sampling principle. The ratio between the proposal and the target distribution is sketched in red. The particles are weighted according to this mismatch and their size reflects the weight value.

it is possible to compute recursively the importance weights, without revising the past generated trajectories, since

$$\begin{aligned}
 w_t^{(i)} &= \frac{p(x_{1:t}^{(i)} | z_{1:t})}{\pi(x_{1:t}^{(i)} | z_{1:t})} \\
 &= \frac{p(z_t | x_t^{(i)})p(x_t^{(i)} | x_{t-1}^{(i)})}{p(z_t | z_{1:t-1})\pi(x_t^{(i)} | x_{1:t-1}^{(i)}, z_{1:t})} \frac{p(x_{1:t-1}^{(i)} | z_{1:t-1})}{\pi(x_{1:t-1}^{(i)} | z_{1:t-1})} \\
 &= \eta \frac{p(z_t | x_t^{(i)})p(x_t^{(i)} | x_{t-1}^{(i)})}{\pi(x_t^{(i)} | x_{1:t-1}^{(i)}, z_{1:t})} w_{t-1}^{(i)} \\
 &\propto \frac{p(z_t | x_t^{(i)})p(x_t^{(i)} | x_{t-1}^{(i)})}{\pi(x_t^{(i)} | x_{1:t-1}^{(i)}, z_{1:t})} w_{t-1}^{(i)} \tag{2.32}
 \end{aligned}$$

Where  $\eta = 1/p(z_t | x_{1:t-1}^{(i)}, z_{1:t-1})$  is a normalization factor. Several approaches select the importance function to be the transition model  $p(x_t | x_{t-1})$ . According to the importance sampling principle the weights  $w_t^{(i)}$  can be computed as follows:

$$\begin{aligned}
 w_t^{(i)} &= \frac{p(x_{1:t}^{(i)} | z_{1:t})}{p(x_t^{(i)} | x_{t-1}^{(i)})\pi(x_{1:t-1}^{(i)})} = [\text{using Equation 2.32}] \\
 &\propto \frac{p(z_t | x_t^{(i)})p(x_t^{(i)} | x_{t-1}^{(i)})}{p(x_t^{(i)} | x_{t-1}^{(i)})} w_{t-1}^{(i)} \\
 &\propto w_{t-1}^{(i)} p(z_t | x_t^{(i)}). \tag{2.33}
 \end{aligned}$$

The appropriateness of such a choice depends on the the observation model  $p(z_t|x_t)$ . If the transition model variance is significantly greater than the observation model one, it can result that a small fraction of the generated samples falls into a high likelihood area, and the filter requires a huge number of particles for representing the state PDF. The SIS algorithm thus consists of a recursive propagation of the weights and support points as each measurement is received sequentially. A pseudo-code of the algorithm is given by [Algorithm 1](#)

---

**Algorithm 1:** Sequential Importance Sampling (SIS)
 

---

**Input:**  $\langle x_{t-1}^{(i)}, w_{t-1}^{(i)} \rangle_{i=1}^N, z_t$

**Output:**  $\langle x_t^{(i)}, w_t^{(i)} \rangle_{i=1}^N$

- 1 **for**  $i = 1$  **to**  $N$  **do**
  - 2     Draw  $x_t^{(i)} \sim \pi(x_t^{(i)} | x_{1:t-1}^{(i)}, z_{1:t})$
  - 3     Compute the particle weight  $w_t^{(i)}$  according to [Equation 2.32](#)
  - 4 **end**
- 

### Sampling Importance Resampling

The direct use of a Sequential Importance Sampling filter requires a huge number of samples, since as the system evolves all of the particles but one will have a high weight. For this reason the Sampling Importance Resampling (SIR) filter [[Gordon, Salmond, & Ewing, 1993](#)] has been introduced.

A SIR filter, sequentially processes the observations  $z_t$  and the state transitions  $x_{t-1} \rightarrow x_t$  as they are perceived, by updating a set of samples representing the estimated distribution  $p(x_{1:t} | z_{1:t})$ .

This is done by performing the following three steps:

1. **Sampling:** The next generation of particles  $\{x_t^{(i)}\}$  is obtained by the previous generation  $\{x_{t-1}^{(i)}\}$ , by sampling from a proposal distribution  $\pi(x_t | x_{1:t-1}^{(i)}, z_{1:t})$ .
2. **Importance Weighting:** An individual importance weight  $w^{(i)}$  is assigned to each particle, according to the IS principle

$$w^{(i)} = w_{t-1}^{(i)} \frac{p(x_t^{(i)} | x_{1:t-1}^{(i)}, z_{1:t})}{\pi(x_t^{(i)} | x_{1:t-1}^{(i)}, z_{1:t})}. \quad (2.34)$$

The weights  $w^{(i)}$  account for the fact that the proposal distribution  $\pi(x_t^{(i)} | x_{1:t-1}^{(i)}, z_{1:t})$  in general is not equal to the true distribution of successor states.

3. **Resampling:** Particles with a low importance weight  $w$  are typically replaced by samples with a high weight. This step is necessary since only a finite number of particles are used to approximate a continuous distribution. Furthermore, resampling allows to apply a particle filter in situations in which the true distribution differs from the proposal one.

A pseudo-code for the SIR filter is given by [Algorithm 2](#). For completeness, we report a common resampling technique in [Algorithm 3](#). Please note that a SIR filter here described, assumes the proposal to be suitable for sequential estimation. This means that  $\pi(x_t^{(i)} | x_{1:t-1}^{(i)}, z_{1:t})$  satisfies [Equation 2.31](#).

---

**Algorithm 2:** Sampling Importance Resampling (SIR)
 

---

**Input:**  $\langle x_{t-1}^{(i)}, w_{t-1}^{(i)} \rangle_{i=1}^N, z_t$   
**Output:**  $\langle x_t^{(i)}, w_t^{(i)} \rangle_{i=1}^N$

- 1 **for**  $i = 1$  **to**  $N$  **do**
- 2     Draw  $x_t^{(i)} \sim \pi(x_t^{(i)} | x_{1:t-1}^{(i)}, z_{1:t})$
- 3     Compute the particle weight  $w_t^{(i)}$  according to [Equation 2.32](#)
- 4 **end**
- // Normalize the weights
- 5 Compute  $t = \sum_{i=1}^N w_t^{(i)}$
- 6 **for**  $i = 1$  **to**  $N$  **do**  $w_t^{(i)} = \frac{w_t^{(i)}}{t}$
- // Resampling using [Algorithm 3](#)
- 7  $\langle x_t^{(i)}, w_t^{(i)} \rangle_{i=1}^N = \text{resample}(\langle x_t^{(i)}, w_t^{(i)} \rangle_{i=1}^N)$

---

### Particle Depletion

While the resampling step is needed for concentrating the computational effort of the filter in state space regions having a high likelihood, it introduces additional problems. This problem becomes evident when the proposal distribution concentrates the sample in regions of low probability of the target distribution. Most of the samples generated by such a proposal have low weight and are likely to be suppressed by resampling. In some degenerated situation, after the resampling step only one particle is retained. This problem is known as particle depletion [[van der Merwe et al., 2000](#)].

Two are the techniques for lessening particle depletion: sampling from a proposal which is closer to the target distribution, and adding artificial noise to the observation model. The first solution is structural, since the choice of a better proposal distribution makes the importance weights value to be similar for all of the particles, in fact limiting the particle suppression in the resampling stage. The second solution can lead to a working filter, but the introduction of artificial noise decreases the accuracy of the estimate with respect to the one achievable by using the original observation model. Moreover, when using a better proposal distribution particle depletion can be lessened by reducing the number of resampling actions.

Some alternative filtering schemes have been proposed, in order to lessen particle depletion, like the auxiliary particle filter by Pitt and Shephard [[Pitt & Shephard, 1999](#)], however the improvements achievable using these techniques are orthogonal to the selection of an improved proposal.

**Algorithm 3:** Systematic Resampling**Input:**  $\mathcal{S}$ , input sample set**Output:**  $\mathcal{R}$ , the unweighted output distribution

---

```

1  $\mathcal{R} = \{\}$ 
2  $n = \text{sampleFromUniform}(0, |\mathcal{S}|^{-1})$ 
3  $t = n$ 
4  $c = 0$ 
5 forall  $s^{(i)} = \langle w^{(i)}, x^{(i)} \rangle \in \mathcal{S}'$  do
6    $c = c + w^{(i)}$ 
7   while  $c > t$  do
8      $r^{(i)} = \langle x^{(i)}, |\mathcal{S}|^{-1} \rangle$ 
9      $\mathcal{R} = \mathcal{R} \cup \{r^{(i)}\}$ 
10     $t = t + |\mathcal{S}|^{-1}$ 
11   end
12 end

```

---

**Number of Effective Samples**

Liu [Liu, 1996] introduced the so-called effective number of particles  $N_{eff}$  to estimate how well the current particle set represents the true posterior. This quantity is computed as

$$N_{eff} = \frac{1}{\sum_{i=1}^N (w^{(i)})^2}. \quad (2.35)$$

The intuition behind  $N_{eff}$  is as follows. If the samples were drawn from the true posterior, the importance weights of the samples would be equal to each other, due to the importance sampling principle. The worse is the approximation the higher is the variance of the importance weights. Since  $N_{eff}$  can be regarded as a measure of the dispersion of the importance weights, it is a useful measure to evaluate how well the particle set approximates the true posterior.

**Choice of the optimal proposal distribution**

The optimal sequential importance function has been introduced in [Liu, 1996]:

$$\pi(x_t | z_{1:t}) = p(x_t | x_{t-1}^{(i)}, z_t) \quad (2.36)$$

$$= \frac{p(z_t | x_{t-1}^{(i)}, x_t) p(x_t | x_{t-1}^{(i)})}{p(z_t | x_{t-1}^{(i)})} \quad (2.37)$$

The optimality has to be intended as minimizing the variance of the importance weights.

If drawing from the optimal proposal distribution, the importance weight  $w^{(i)}$  for each particle  $i$  is computed according to [item 2.34](#):

$$\begin{aligned}
 w_t^{(i)} &= \frac{p(x_{1:t}^{(i)} | z_{1:t})}{\pi(x_{1:t}^{(i)} | z_{1:t})} = & (2.38) \\
 &= \frac{p(x_{1:t}^{(i)} | z_{1:t})}{p(x_t^{(i)} | x_{t-1}^{(i)}, z_t) \pi(x_{1:t-1}^{(i)} | z_{1:t-1})} \\
 &\propto \frac{p(z_t | x_t^{(i)}) p(x_t^{(i)} | x_{t-1}^{(i)})}{p(x_t | x_{t-1}^{(i)}, z_t)} w_{t-1}^{(i)} \\
 &= w_{t-1}^{(i)} p(z_t | x_{t-1}^{(i)}). & (2.39)
 \end{aligned}$$

### Rao-Blackwellization

When the state space is very big, sampling can be computationally expensive. In this case, however, it is possible to address the problem by exploiting the Rao Blackwell theorem, obtaining a Rao-Blackwellized particle filter [[Doucet, 1998](#)]. This technique can be applied when the dynamic Bayesian network of the system has a particular structure [[Doucet et al., 2000a](#)]. It consists in partitioning the state space  $X$  in two subspaces  $X^a$  and  $X^b$ . The partition has to be made according to the structure of the system, ensuring that, given a state  $x_t \in X$ , its projection  $x_t^a \in X^a$  depends only on the previous state  $x_{t-1}$  and the current observation  $z_t$ . The  $x$  projection  $x_t^b \in X^b$  should be updated analytically and efficiently once  $x_t^a$  is known. This partitioning allows to sample only from  $X^a$ , in fact decreasing the effective sampling space dimension.

## 2.2 Grid Mapping

There exist different types of models for representing the environment which are frequently used in mobile robotics. The most common ones are feature maps, geometric maps, and grid maps. A feature map stores a set of features detected in the environment. Typical features are lines and corners when proximity sensors are used. Other possibilities are visual features based on the scale invariant feature transform (SIFT) [[Lowe, 2004](#)] whenever a camera is used to perceive the environment. For each feature, these maps store the feature information together with a coordinate and eventually an uncertainty measure. Geometric maps represent all obstacles detected by the robot as geometric objects, like circles or polygons. This kind of representation is comparably compact and needs only few memory resources. Throughout this thesis, we use grid maps to model the environment. Grid maps discretize the environment into so-called grid cells. Each cell stores information about the area it covers. Most frequently used are occupancy grid maps that store for each cell a single value representing the probability that this cell is occupied by an obstacle. The advantage of grids is that they do not rely on predefined features which need to be extracted from sensor data. However, they have the disadvantages of discretization errors and of requiring a lot of memory resources. In this section, we first introduce the occupancy mapping algorithm, developed by Moravec and Elfes [[Moravec, 1988](#)]. Afterwards, we briefly describe a variant called reflection probability maps.

### 2.2.1 Occupancy Probability Mapping

Occupancy grids store for each cell  $c$  a probability ( $c$ ) of being occupied by an obstacle. In the following, we will derive the map update algorithm introduced by Moravec and Elfes [Moravec, 1988] which computes the occupancy probability  $p(m)$  for the grid map  $m$ . Assuming that the different cells are independent we have that

$$p(m) = \prod_{c \in m} P(c) \quad (2.40)$$

Let  $z_{1:t}$  be a sequence of observations obtained by the robot at the positions  $x_{1:t}$ . By conditioning the map probability over those variables, we obtain

$$p(c|x_{1:t}, z_{1:t}) = \frac{p(z_t|c, x_{1:t}, z_{1:t-1})p(c|x_{1:t}, z_{1:t-1})}{p(z_t|x_{1:t}, z_{1:t-1})} \quad (2.41)$$

Assuming that the process is Markovian, we have

$$p(c|x_{1:t}, z_{1:t}) = \frac{p(z_t|c, x_t)p(c|x_{1:t}, z_{1:t-1})}{p(z_t|x_{1:t}, z_{1:t-1})} \quad (2.42)$$

Now consider the term  $p(z_t|c, x_t)$ . Applying Bayes' rule we obtain

$$p(z_t|c, x_t) = \frac{p(c|x_t, z_t)p(z_t|x_t)}{p(c|x_t)} \quad (2.43)$$

Let now combine together Equation 2.42 and Equation 2.43. Let us furthermore assume that  $x_t$  and  $c$  are independent if not conditioned on  $z_t$ . This leads to

$$p(c|x_{1:t}, z_{1:t}) = \frac{p(c|x_t, z_t)p(z_t|x_t)p(c|x_{1:t}, z_{1:t-1})}{p(c)p(z_t|x_{1:t}, z_{1:t-1})} \quad (2.44)$$

Exploiting the fact that each cell is a binary variable, we have that

$$p(\neg c|x_{1:t}, z_{1:t}) = \frac{p(\neg c|x_t, z_t)p(z_t|x_t)p(\neg c|x_{1:t}, z_{1:t-1})}{p(\neg c)p(z_t|x_{1:t}, z_{1:t-1})} \quad (2.45)$$

dividing Equation 2.44 by Equation 2.45 we obtain

$$\frac{p(c|x_{1:t}, z_{1:t})}{p(\neg c|x_{1:t}, z_{1:t})} = \frac{p(c|x_t, z_t)p(\neg c)p(c|x_{1:t}, z_{1:t-1})}{p(\neg c|x_t, z_t)p(c)p(\neg c|x_{1:t}, z_{1:t-1})} \quad (2.46)$$

Finally, using the fact that  $p(\neg c) = 1 - p(c)$  and defining

$$\text{Odds}(x) = \frac{p(x)}{1 - p(x)} \quad (2.47)$$

we obtain

$$\text{Odds}(c|x_{1:t}, z_{1:t}) = \text{Odds}(c|x_t, z_t)\text{Odds}(c)^{-1}\text{Odds}(c|x_{1:t}, z_{1:t-1}) \quad (2.48)$$

To recover the occupancy probability from the Odds representation given in Equation 2.48 we use the following formula which can easily be derived from Equation 2.47

$$p(x) = \frac{\text{Odds}(x)}{1 + \text{Odds}(x)} \quad (2.49)$$



This leads to the following occupancy update formula

$$p(c|x_{1:t}, z_{1:t}) = \left[ 1 + \frac{1p(c|x_t, z_t)}{p(c|x_t, z_t)} \cdot \frac{p(c)}{1-p(c)} \cdot \frac{1p(c|x_{1:t1}, z_{1:t1})}{p(c|x_{1:t1}, z_{1:t1})} \right]^{-1} \quad (2.50)$$

**Equation 2.50** tells us how to update our belief  $p(c|x_{1:t}, z_{1:t})$  about the occupancy probability of a grid cell given sensory input. In practice, one often assumes that the occupancy prior is 0.5 for all cells so that  $\frac{p(c)}{(1p(c))}$  can be removed from the equation.

### 2.2.2 Reflection Probability Mapping

Beside occupancy probability grids, there exist alternative realization of grid maps. A frequently used model is the so-called reflection probability map or counting model. In contrast to occupancy grid maps, they store for each cell a reflection probability value. This value provides the probability that a measurement covering the cell is reflected. Note that the occupancy model and the counting model are similar but not identical. In this model, we are interested in computing the most likely reflection probability map  $m^*$  given the observations and poses of the robot.

$$m = \underset{m}{\operatorname{argmax}} p(m|x_{1:t}, z_{1:t}) \quad (2.51)$$

Assuming a uniform prior probability for  $p(m)$  this is equivalent to maximize

$$m = \underset{m}{\operatorname{argmax}} p(z_{1:t}|x_{1:t}, m) \quad (2.52)$$

$$= \underset{m}{\operatorname{argmax}} \prod_{i=1}^t p(z_i|x_i, m) \quad (2.53)$$

$$= \underset{m}{\operatorname{argmax}} \sum_{i=1}^t \log p(z_i|x_i, m) \quad (2.54)$$

Now we have to define the likelihood function  $p(z_t|x_t, m)$ . This is done by defining the likelihood of each beam  $z_{t,n}$

$$p(z_{t,n}|m, x_t) = \begin{cases} \prod_{k=0}^{z_{t,n}-1} (1 - m_{f(x_t, n, k)}) & \text{if } \xi_{t,n} = 1 \\ m_{f(x_t, n, k)} \prod_{k=0}^{z_{t,n}-1} (1 - m_{f(x_t, n, k)}) & \text{if } \xi_{t,n} = 0 \end{cases} \quad (2.55)$$

where  $m_j$  is the  $j$ -th cell of the map and  $f(x_t, n, k)$  maps the reading beam to cells of the map.  $\xi_{t,n}$  is binary variable defining if the reading is reflected by an object or is a max-reading one. By placing **Equation 2.55** into **Equation 2.54** we have

$$m^* = \underset{m}{\operatorname{argmax}} \left[ \sum_{j=1}^J \alpha_j \log m_j + \beta_j \log(1 - m_j) \right] \quad (2.56)$$

$$\alpha_j = \sum_{t=1}^T \sum_{n=1}^N \mathcal{I}(f_{x_t, n, z_{t,n}} = j)(1 - x_{t,n}) \quad (2.57)$$

$$\beta_j = \sum_{t=1}^T \sum_{n=1}^N \sum_{k=0}^{z_{t,n}-1} \mathcal{I}(f_{x_t, n, k} = j) \quad (2.58)$$

Where  $\alpha_j$  corresponds to the number of times a beam that is not max-range ended in cell  $j$  ( $hits(j)$ ) and  $\beta_j$  corresponds to the number of times a beam intercepted a cell  $j$  without ending in it ( $misses(j)$ ).

If we compute the gradient of the map with respect to the single cells, we obtain that

$$\frac{\partial m}{\partial m_j} = \frac{\alpha_j}{m_j} - \frac{\beta_j}{1 - m_j} \quad (2.59)$$

Setting the gradient to 0 we have that

$$m_j = \frac{\alpha_j}{\alpha_j + \beta_j} \quad (2.60)$$

The reflection propability is thus obtained by counting the number of times the cell got hit, divided by the number of times the cell got observed (hit + miss). Since the value of each cell can be determined by counting, this technique is also called counting model. The differences between occupancy probability and reflection probability maps is that the occupancy probability typically converges to 0 or 1 for each cell which is frequently observed. In contrast to that, reflection probability values converge to values between 0 and 1. Values significantly different from 0 or 1 often occur when mapping objects much smaller than the grid discretization or, for example, for glass panes which are repeatedly observed with a laser range finder.

## Chapter 3

# A Taxonomy of Simultaneous Localization and Mapping

### 3.1 Introduction

In this section, an overview of the Simultaneous Localization And Mapping is given. Roughly speaking, the SLAM problem consists in jointly estimating the position and orientation of the robot, as well as a map of the environment, given the observation histories. The two terms in the estimation process (robot pose and map) are strictly connected with each other. This is often characterized as a chicken and egg problem, as a map is needed for a good pose estimation, while a good estimation of the pose is needed for building a consistent map [Thrun, 2002]

Since Smith, Self and Cheeseman seminal paper [Smith, Self, & Cheeseman, 1990], researchers have been converging toward a common understanding: probabilistic estimation. They estimate the distribution over the possible maps and robot poses, given the sensor readings. This is primarily due to the simplicity of modeling measurement and process noise. Even if some other approaches, mainly based on optimization techniques, seem to diverge from this common view, they can be seen as special cases. They solve the problem by searching for the map/trajectory which minimizes the error among the observations and the candidate solution. However, the two approaches are substantially equivalent: the world configuration with minimum error can be seen as the configuration for which the readings are more likely.

The key idea is to model the system in terms of probability distributions, thus reducing the SLAM problem in a density estimation problem. A probabilistic formulation of the SLAM problem consists in the estimation of the following distribution:

$$p(m_t, x_t | z_{1:t}, u_{0:t-1}) \quad (3.1)$$

here  $m$  is the unknown map,  $x_t$  is the robot pose in the environment at time  $t$ ,  $z_{1:t} = \{z_1, z_2, \dots, z_t\}$  and  $u_{0:t} = \{u_0, u_1, \dots, u_{t-1}\}$  are respectively the exteroceptive sensing history and the proprioceptive sensing history up to time  $t$ . The exteroceptive sensors are able to

acquire robot centered environment measures. For this reason the exteroceptive sensing history  $z_{1:t}$  can be also referred to as observation history, since it is relative to the observations made about the environment. Conversely, proprioceptive sensors are able to measure variations of the robot parameters, like changes in relative position, or the speed of the wheels.

In designing a SLAM algorithm, a crucial issue is the choice of the map representation. This choice is affected by both the assumptions made on the system and on the environment, like the possibility of extracting unique landmarks, or the presence of structure in the operating scenario. Some methods focus on the construction of a topological representation of the environment, by describing the relationships among the different places, abstracting on the concept of distance. Others proposed pure metric representations.

In this work, we focus on the construction of dense grid maps, for planar environments, although some of the described concepts can be generalized to different representations. In the rest of this chapter, we describe some relevant map representations. Subsequently, we present the techniques that can be used for updating the above representations. Finally, we present and discuss some open issues.

## 3.2 Map Representations

The representation of the map is one of the fundamental issues in the SLAM problem. Good representations should be rich enough to let the robot localize and autonomously navigate, while being compact and with low spatial complexity. Moreover, different representations yield to different approaches and techniques. In this section, we will describe some of the most used map representations. Grid map representations are first described. Those kind of maps are very close to the human idea of a spatial representation, as they describe the world with a bidimensional or tridimensional grid, whose cells keep the probability of being occupied. They are very useful for accurate localization and path planning in presence of obstacles. However, they also require high memory occupation. Landmark maps describe the world with a set of landmarks, thus being less informative and requiring less memory. On the other hand, topological representations are described. While very compact, those representation are not suitable for accurate low level tasks, as they often describe an environment in terms of places and connections. Their main purpose is to perform qualitative reasoning about the environment. Lately, several efforts have been profused toward the integration of the previous representations. This integration seems to inherit good qualities from both, thus being very promising.

### 3.2.1 Landmark Maps

Landmark maps describe the world through as a set of spatially located features [Castellanos *et al.*, 1999],[Montemerlo *et al.*, 2002]. One of the advantage of this kind of representation is its compactness, which makes them suitable for describing very large planar and 3D environments. More formally, a map instance is represented by a vector in the space  $\mathbb{R}^{l \cdot n}$ , where  $l$  is the dimension of the single landmark and  $n$  is the number of landmarks.

If we assume that the noise affecting the system is Gaussian, the posterior over the map can be easily modeled as multivariate Gaussian distribution. Keeping a fully correlated distribution among the landmark locations is possible simply through a covariance matrix. The



Figure 3.1: Example of a landmark map. The picture shows the map of the Victoria Park dataset. The cyan dots represented the features while the blue line the robot path. Courtesy of Juan Nieto.

blocks along the diagonal of the matrix express the absolute uncertainties for the single landmark, while the off-diagonal blocks express the relationships among different landmarks. Moreover, assuming the system to be linear, the straightforward Kalman Filter solution is also the optimal one.

Despite these clear advantages, the use of landmark maps presents some drawbacks. Summarizing all of the knowledge about the environment in a set of points can discard useful information that can be gathered by the today accurate sensors. In order to operate with a landmark map some feature extraction algorithm, is needed. Usually, feature extraction algorithms assume some structure in the environment. For this reason environments for which some a priori knowledge is missing, can not be represented by landmark maps. Moreover, this features have to be uniquely identified. This problem, known as perceptual aliasing or data association, has to be explicitly addressed, as wrong associations have been proved to yield estimation divergence.

The process of feature extraction differs significantly according to the sensor used. On one side, when using sonar and laser range-finders, there are mainly two classes of features: corners and lines. Extraction is carried on by using different algorithms: split and merge [Borges, 2000], Hough transform [Atiquzzaman & Akhtar, 1994], RANSAC [Fischler & Bolles, 1981]. On the other end, when using camera sensors, feature are extracted by exploiting appearance information (mainly color and luminance) Features are represented as

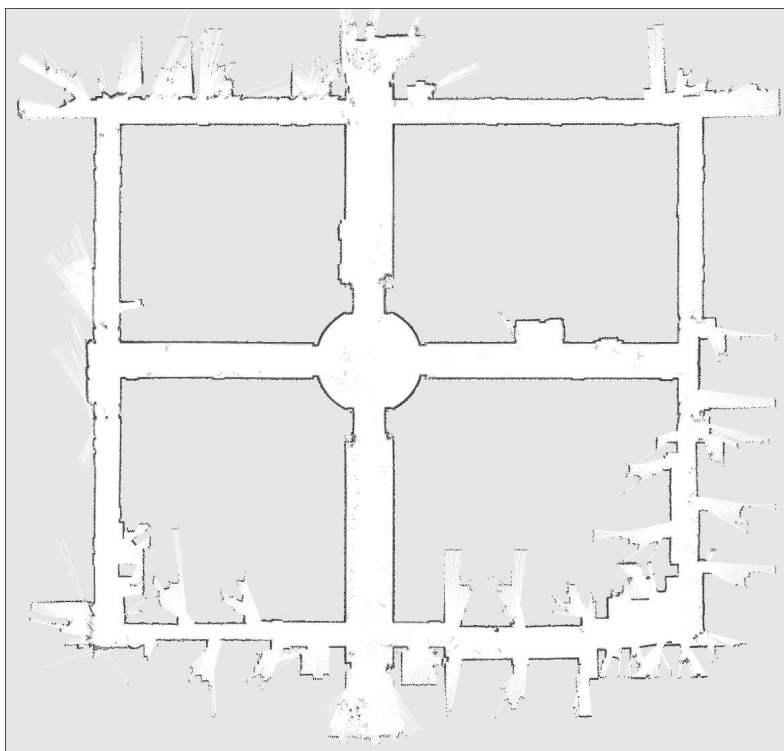


Figure 3.2: Example of grid map. The picture shows the map of the ACES data set. The gray value represent the probability of a cell of being occupied. The darker a cell is, the higher is the probability.

points and extracted using different algorithms: Harris [Harris & Stephens, 1988], STK [Shi & Tomasi, 1994], SIFT [Lowe, 2004]. Moreover, single cameras provide only bearing information, thus careful initialization procedures are needed [Montiel, Civera, & Davison, 2006]

An example of landmark map is shown in Figure 3.1.

### 3.2.2 Grid Maps

When a fine grained resolution is needed, typical for accurate path-planning, or environments are structure-free, landmarks map can not be informative enough. In these situations, the use of a dense map is handful. Dense maps can be thought as a two or three dimensional grid, whose cells keep the probability of being occupied or free. Algorithms for building this kind of maps relies on the Occupancy Grid framework of Moravec and Elfes [Elfes, 1989]. Another approach, mostly used with high accuracy sensors like laser scanner, is represented by the reflection grid. Both algorithm are explained in details in Section 2.2.

One of the main characteristic of this kind of data structure is the capability of representing unstructured environments, and that the accuracy can be tuned by setting different

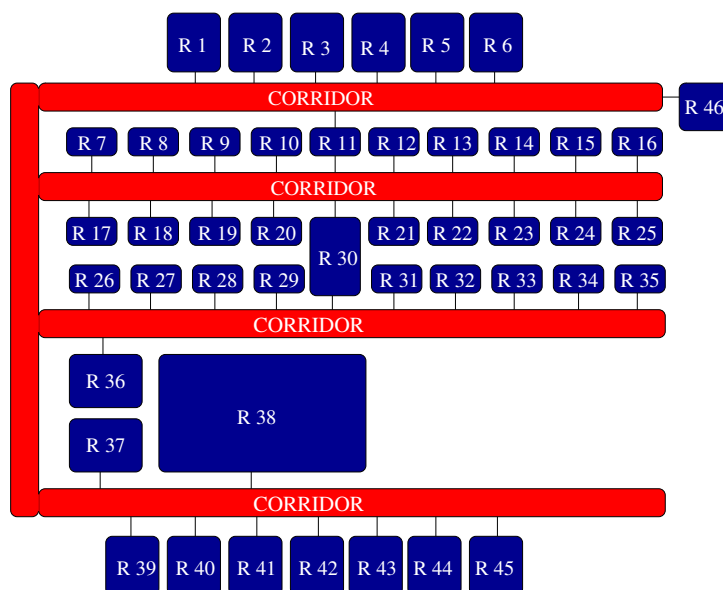


Figure 3.3: Example of a semantic/topological map. The picture shows a typical indoor environment. The blue nodes represent rooms in the environment, while the red ones represent the corridors. The edges represent the connections among the different rooms. Courtesy of Oscar Martinez Mozos.

grid resolution. Moreover, there is no need to use a feature extraction algorithm, making the perceptual aliasing problem easier and the overall algorithm more robust.

However, cells are considered independent, so inter-cell relations cannot be expressed. This poses limitations when dealing with dynamic environments, as groups of cells tend to move together. Another disadvantage of the grid maps are the memory requirements. Storing a grid map requires an amount of memory which grows with the area of the environment, and the desired resolution. This drawback becomes a problem when multiple grids are used for representing a sampled probability distribution over maps.

Dense maps are usually built using sonar or laser range finders. This is due to the need of using both range and bearing information. However, algorithm using stereo cameras have been developed lately. One example of this is the work of Sim and Little [Sim *et al.*, 2006].

An example of grid map is given in Figure 3.2.

### 3.2.3 Topological Maps

A topological map represents the environment as a graph. The nodes of the graph represent *places* in the environment, while the edges represent *relationships* among places [Kuipers & Byun, 1991; Mataric, 1992]. As opposed to the metric representation, within a pure topological map the spatial meaning of nodes and relationships is lost. This makes this kind of representation less sensitive to scale problems affecting metric approaches: as the size of



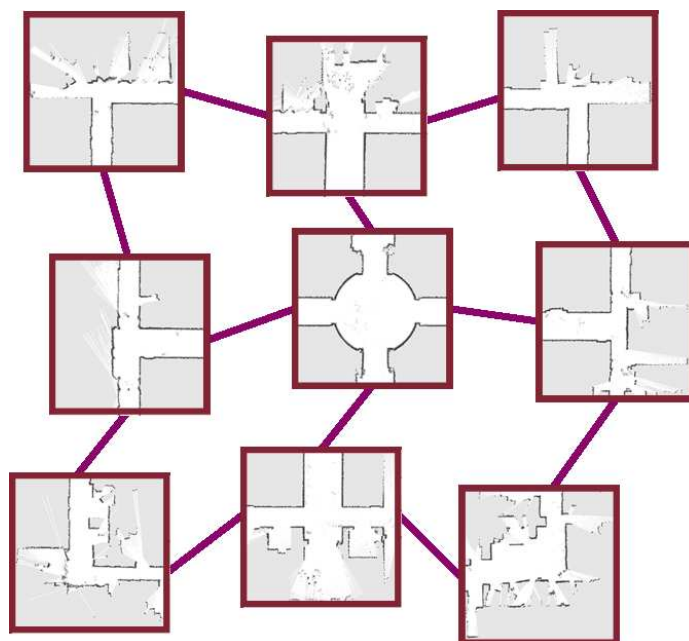


Figure 3.4: Example of a hybrid map. The picture shows the map of the ACES data set. The graph nodes represent the topological skeleton, while the small local maps represent its metrical part.

the environment grows the size of a metric maps describing it grows, while the size of the corresponding topological map remains the same.

The precise definition of a topological map is ontology dependent: the meanings of both “place” and “relationship” have to be specified. Different ontologies lead to different topological map definitions. The compactness of topological maps allows to perform high level reasoning about such a representation. Despite their attractiveness, it is difficult to use these representations without any kind of metric information. This kind of representation is often connected with some semantic interpretation of the world having the different places grouped into different classes, e.g. corridors, rooms and so on.

An example of semantic/topological map is given in [Figure 3.3](#)

### 3.2.4 Hybrid Maps

Hybrid maps try to get the best out of both metric and topological maps. The relationships among different parts of the environment are often *local*: elements of the environment that are close are highly correlated, while elements that are far away are weakly correlated. Hybrid maps try to capture this feature. The environment is described as a set of small local maps (patches), while the global constraints among them are retained in a global graph. This kind of representation is also called hierarchical map, as this decoupling can also be seen as a two-



layer map: the lower level stores the metrical and local information, the higher level stores the topological and global one. In principle, this hierarchicalization can be repeated for several levels, but most of the approaches in the literature use only two layers.

Within a two layer hybrid map the global map uses a graph for keeping the topological structure of the local maps locations, and for constructing a local view of the world in the neighborhood of the robot. Bosse *et al.* [Bosse *et al.*, 2003] have proposed a landmark based hierarchical representation in their Atlas framework. Howard *et al.* [Howard, 2004] proposed a hierarchical representation in which the landmarks are small grid maps, arranged in a graph having a precise spatial meaning. Observe that such a graph is substantially different from a topological representation, since it expresses in a compact way relationships among metric entities while, in a topological map, the metric meaning is lost.

An example of an hybrid map is given in [Figure 3.4](#)

### Delayed State Representation

Lately, the Delayed State or View Based Framework has become more and more popular. In contrast to conventional feature based approaches, the world is represented by a series of past vehicle poses with associated uncertainties. Previous observations, whether 2D or 3D laser based, or visual, are then associated to each pose. These can then be referred back, compared and perhaps registered, to other potential constraints on the global map of vehicle poses. This is most commonly done immediately after odometry-based state augmentation, but is also essential when revisiting previously traversed areas for loop closing. The use of Delayed State representation decouples two important aspects. Focusing on robot poses, it is possible to express some process properties abstracting on the real map representation, as soon as it can provide pose constraints. This offers natural ways of expressing a map hierarchy. The local part of the hierarchy focuses on map representations dealing with different sensors types. The global part is expressed as a network of poses, thus dealing with problems of global consistency.

## 3.3 Techniques

In this section, we discuss the diverse SLAM approaches developed in the past years. We start from a widely used class of metric mappers: the incremental Maximum Likelihood (e.g. scan matching, visual odometry) approaches. Subsequently, we introduce the optimization based approaches, developed for recovering a globally consistent map from a set of relations among parts of the environment. Furthermore, we illustrate an Expectation Maximization based algorithm which has been introduced in the attempt of lessening the local minima problems, which arise when using optimization techniques.

Solutions based on filtering techniques will be then presented. We start describing the Kalman filter, which has been widely used in landmark based SLAM, in [Subsection 3.3.4](#). We then describe approaches based on its dual, the information filter, in [Subsection 3.3.5](#). [Subsection 3.3.6](#) introduces the Rao-Blackwellized particle filter based approaches, which have been shown to be effective for dealing with complex multi-modal map/pose posteriors.

In [Subsection 3.3.7](#) we describe some approaches to the topological SLAM problem. In this setting, only a qualitative estimation of the environment is performed. Geometric an

metric information are discarded and the main focus is estimating the connectivity of diverse places in the world.

We conclude with the hybrid approaches which update a hierarchical representation combining both a metric and a topological aspect.

### 3.3.1 Incremental Maximum Likelihood

Popular approaches to on-line mapping are the Incremental Maximum Likelihood (ML) techniques [Lu & Milios, 1994; Gutmann, 1996; Biber & Strasser, 2003]. The general idea of these approaches can be summarized as follows. At any point  $t - 1$  in time, the robot is given an estimate of its pose  $\hat{x}_{t-1}$  and a map  $\hat{m}(\hat{x}_{1:t-1}, z_{1:t-1})$ , constructed using the incremental trajectory estimate  $\hat{x}_{1:t-1}$ . After the robot moves further on and after taking a new measurement  $z_t$ , the robot determines the most likely new pose  $\hat{x}_t$  as

$$\hat{x}_t = \operatorname{argmax}_{x_t} [p(z_t | x_t, \hat{m}(\hat{x}_{1:t-1}, z_{1:t-1})) \cdot p(x_t | u_{t-1}, \hat{x}_{t-1})]. \quad (3.2)$$

The idea is to trade off the consistency of the measurement with the map (first term on the right-hand side in Equation 3.2) and the consistency of the new pose with the control action and the previous pose (second term on the right-hand side in Equation 3.2). The map is then extended by the new measurement  $z_t$ , using the pose  $\hat{x}_t$  as the pose at which this measurement was taken. The key limitation of these approaches lies in the greedy maximization step. Once the location  $x_t$  at time  $t$  has been computed it is not revised afterward so that the robot cannot recover from errors affecting the past pose from which the map is computed (registration errors). Although they have been proved to be able to correct enormous errors in odometry, the resulting maps are often globally inconsistent, as local errors in the maximization are not considered, and can grow unbounded over time.

Observe that the definition of Maximum Likelihood approaches given in this section refers to *incremental* methods. Incremental ML methods substantially differ from Global ML approaches which attempt to find the robot trajectory which maximizes the likelihoods of *all* the observations and the odometry measurements. A global maximization is usually achieved using optimization based techniques (Subsection 3.3.2).

In general, the more accurate is the solution of Equation 3.2, the more accurate the map generated by the ML algorithm will be. The appreciable quality of generating locally consistent maps makes the Maximum Likelihood approaches suitable to be used as building blocks of more complex algorithms. These considerations motivate the research in developing increasingly efficient algorithms.

Techniques in this field mainly differ according to the sensor used. When a laser range finder is used, they are called scan matching techniques [Gutmann, Weigel, & Nebel, 2001; Hähnel, Schulz, & Burgard, 2002a; Röfer, 2002; Weiß, Wetzler, & von Puttkamer, 1994]. A common technique in the context of laser range scans is the iterative-closest-point (ICP) algorithm [Besl & MacKay, 1992; Zhang, 1994]. When, instead, the sensor used is a camera, they are called visual odometry techniques [Nister, Naroditsky, & Bergen, 2004; Ni & Dellaert, 2006; Agrawal & Konolige, 2006; Cheng, Maimone, & Matthies, 2005] and are mainly based on optical flow ideas [Horn & Schunck, 1992].

### Scan Matching

The scan matching problem can be expressed as: *given two sets of 2D data (i.e. a reference scan,  $s$  and a current scan,  $g$ ), determine a 2D rigid motion (a translation  $T$  and a rotation  $R_\phi$ ) that makes the scan data overlapping the reference data.*

In typical scenarios, however, the perfect overlap is not possible, thus the objective becomes to transform  $s$  in such a way that the distance between the points in  $g$  and the transformed points in  $s$  is minimized

$$R_\phi, T = \arg \min_{R_\phi, T} \sum_{s_i, g_i \in C} \|T + R_\phi s_i - g_i\|^2. \quad (3.3)$$

where  $s_i, g_i \in C$  is a pair of corresponding points in the two scans.

If the environment is rich in features that are invariant in rotation and translation (corners, straight walls) [Gutmann, 1996] and these are preserved notwithstanding the sensor noise, then it is possible to extract them and find a solution in linear time with respect to the number of points in the scan [Lingemann *et al.*, 2004]. This approach has been proved to be efficient and robust, when sufficient features are present in the environment.

In unstructured environments with relatively low noise, it is possible to employ algorithms of the ICP family [Zhang, 1994] that do not assume the existence of features. These are based on a two-step process: first, a series of heuristic correspondences between points in the two scans are established. Then a roto-translation that approximately satisfies the series of constraints is found. The solution is obtained by iteratively executing the two steps until the error drops below a given threshold. The first approach to be used in robotics is the work of Lu and Milios [Lu & Milios, 1994]. They combine the normal Nearest Neighbor search of ICP with angular constraints in the Iterative Dual Correspondence (IDC). The algorithm uses two types of correspondences (translation and rotation) and at each iteration performs two optimizations. They also propose the interpolation of lines between laser points to improve robustness for large transformations. After their work, Pfister *et al.* [Pfister *et al.*, 2002] propose a weighted algorithm, where the influence of each point in the error formulation is weighted according to uncertainty like measurement noise, sensor incidence angle and correspondence error. This error is then minimized using numerical optimization methods. An improved metric is introduced by Minguez *et al.* [Minguez, Montesano, & Lamiroux, 2006]. The correspondences between scans are established with this measure and the minimization of the error is also carried out in terms of this distance. As a result, the translation and rotation are compensated simultaneously. Censi [Censi, 2008] introduced a point to line metric and an exact closed-form for minimizing it. The resulting algorithm has some interesting properties: it converges quadratically, and in a finite number of steps.

Other methods do not require the establishment of any feature-to-feature or point-to-point correspondence but they search in the solution space. In [Hähnel, Schulz, & Burgard, 2002b] the solution is searched by performing gradient descent on a score function. Such a function is built by convolving the reference scan with a Gaussian kernel and then correlating it with the sensor scan. This approach has been subsequently refined in [Biber & Strasser, 2003] by defining a closed form for a potential function that can be minimized using the Newton algorithm. These approaches require a number of iterations that depends on the input configuration and the entity of the error. In [Weiß, Wetzler, & von Puttkamer, 1994] a correlation based approach that is well suited in polygonal environments is presented. The orientation of each scan point is found and the circular histogram of these is build. Then the rotation is

found by correlating the normal histograms. This can be done since the relative angles of a set of lines do not change when changing the reference system. Once the heading is known, the translational component of the displacement is recovered by determining pair of non parallel lines in the reference scan, and computing the translational error among the normal directions of those lines. Censi *et al* [Censi, Iocchi, & Grisetti, 2005] proposed a matching technique based on the Hough transform (HT). Within the Hough space the search of the rotation and the translation which maximize the overlapping can be decoupled. This way the complexity of the matching process which requires to search in a 3D space is decoupled in a sequence of mono-dimensional searches.

Finally, some researches focused on estimating the uncertainty of the scan matching algorithm. The most rigorous study of the covariance estimation problem has been developed by Bengtsson *et al* [Bengtsson & Baerfeldt, 2001]: nevertheless, the two methods proposed there (the Hessian method and the Offline method) have some drawbacks. The closed-form Hessian method over-estimates the covariance in some cases. The Offline method gives reasonable results but cannot be used online, as it is based on a computationally expensive procedure. Censi [Censi, 2007a] presented a method for estimating the covariance of the ICP algorithm, based on the analysis of the error function being minimized. He showed that underconstrained cases can be detected by examining the Fishers information matrix. In such cases the proposed method accounts for the errors on the observable manifold.

### Visual Odometry

Visual odometry estimates a continuous camera trajectory by examining the changes motion induces on the images. There are two main classes of techniques

- Dense algorithms, based on optical flow
- Sparse algorithms, based on structure from motion

Dense motion algorithms, exploiting the optical flow, track the image motion of brightness patterns over the full image [Horn & Schunck, 1992] Campbell *et al*. [Campbell, 2004] presents a visual odometry system that uses optical flow and a planar world assumption to obtain relative poses from a monocular camera. In the work of Morency and Gupta [Morency & Gupta, 2003] the relative transformation between two stereo image pairs is estimated using a hybrid registration algorithm which combines the robustness of multi-scale feature tracking for large movements and the accuracy of 3D normal flow constraints. Their hybrid technique takes advantage of depth information available from the stereo camera which makes the approach less sensitive to lighting variations. Similarly, Comport *et al*. [Comport, Malis, & Rives, 2007] use a dense minimization approach which directly exploits all gray-scale information available within the stereo pair (or stereo region) leading to very robust and precise results. Metric 3D structure constraints are imposed by consistently warping corresponding stereo images to generate novel viewpoints at each stereo acquisition. An iterative non-linear trajectory estimation approach is formulated based on a quadrifocal relationship between the image intensities within adjacent views of the stereo pair. A robust M-estimation technique is used to reject outliers corresponding to moving objects within the scene or other outliers such as occlusions and illumination changes.

Feature tracking methods track a small number of features from image to image. The use of features reduces the complexity of dealing with image data and makes realtime perfor-

mance more realistic. SfM systems are a type of feature tracking method that use structure-from-motion methods to estimate the relative position of two or more camera frames. These feature points are triangulated and then an absolute orientation step is used to estimate the 3D motion. The use of 3D point correspondences to obtain the motion suffers from a major drawback—triangulations are much more uncertain in the depth direction. Therefore, these 3D points have non isotropic noise, and a 3D alignment between small sets of such 3D points gives poor motion estimates. To take into account this anisotropic noise in the 3D coordinates, Matei and Meer [Matei & Meer, 1999] presented an approach based on a technique from statistics called bootstrap to estimate the covariance for the 3D points and solve a heteroscedastic, multivariate errors in variables regression problem. Nister *et al.* [Nister, Naroditsky, & Bergen, 2004] propose a method that proceeds by triangulating the feature points and then tracking them over time. The 3-point algorithm for single camera pose is then used to estimate the motion of the left camera. Each triplet of triangulated 3D points is used to generate a hypothesis in the RANSAC [Fischler & Bolles, 1981] framework. This hypothesis is then scored using pixel reprojections in both the left and the right cameras. Ni and Dellaert [Ni & Dellaert, 2006] present an approach of calculating visual odometry for outdoor robots equipped with a stereo rig. Instead of the typical feature matching or tracking, they use an improved stereo-tracking method that simultaneously decides the feature displacement in both cameras. Based on the matched features, a three-point algorithm for the resulting quadrifocal setting is carried out in a RANSAC framework to recover the unknown odometry. Agrawal and Konolige [Agrawal & Konolige, 2006] rely on stereo vision to robustly estimate frame-to-frame motion in real time. The motion estimation problem is formulated efficiently in the disparity space and results in accurate and robust estimates of the motion even for a small-baseline configuration. Inertial measurements are used to fill in motion estimates when visual odometry fails. This incremental motion is then fused with a low-cost GPS sensor using a Kalman Filter to prevent long-term drifts. Levin and Szeliski [Levin & Szeliski, 2004] study how estimates of ego-motion based on feature tracking can be improved using a rough (low accuracy) map of where the observer has been. Since absolute estimates of camera position are unreliable, they use stable local information such as change in orientation to perform the alignment. The final alignment is computed using a graphical model whose MAP estimate is inferred using loopy belief propagation.

### 3.3.2 Optimization Based Techniques

The main idea of this kind of approaches is to construct from the sensing history a graph of relations, in which each robot pose is a node. A relation among the pose  $x_i$  and the pose  $x_j$  holds if there is some landmark  $z$  that has been observed from both  $x_i$  and  $x_j$ . Let  $z = f(x, z')$  be a function that maps the local perception of a landmark to the global frame, according to the pose  $x$ . For instance, if  $z'$  are the coordinates of the landmark, expressed in the robot centered frame,  $f(x, z')$  can be the coordinate transformation which returns the global coordinate of  $z$ , given the robot pose.

Given a landmark  $z_l$  seen from both  $x_i$  and  $x_j$  we can define the following error term:

$$e_{ijl} = (f(x_i, z_{li}) - f(x_j, z_{lj}))^T \Sigma_{lij}^{-1} (f(x_i, z_{li}) - f(x_j, z_{lj})) \quad (3.4)$$

Here  $\Sigma_{lij}^{-1}$  is a positive semidefinite matrix that captures the strength of the constraint.

The total error of the system is the sum of the errors affecting each pair of poses  $\langle i, j \rangle$  for which a common landmark  $l$  has been seen. It can be computed as

$$E(x_0, \dots, x_t) = \sum_{i,j,l} e_{ijl}. \quad (3.5)$$

Now the problem is how to minimize the error expressed by Equation 3.5, being in general non linear and high dimensional. To this end several techniques have been proposed.

The original formulation of Lu and Milios [Lu & Milios, 1997a] minimizes Equation 3.5 using an iterative approach. At each step the original system is linearized around the previous iteration guess, and the corresponding system is solved. The termination occurs when the error is below a given threshold. Under the assumption that the correct solution is close to the starting point, a popular approach consists in computing the first order Taylor expansion of the error function around the starting position  $x_0$ :

$$f(x, z') \simeq f(x_0, z) + \nabla_x f|_{x_0} (x - x_0)$$

With this assumption Equation 3.5 translates in a quadratic form, whose maximization is straightforward. The linearized system size grows linearly with the number of poses, thus requiring a time quadratic in the number of landmarks for being solved. However, if the underlying problem is not linear, the procedure has to be iterated several times, as explained by Lu and Milios [Lu & Milios, 1997a].

Howard *et al.* [Howard, Mataric, & Sukhatme, 2001a] proposed to use relaxation for minimizing the error energy of the SLAM equivalent mechanical system. Equation 3.5 can be seen as the description of a set of rigid bodies interconnected with springs and dampers. At the initial instant the system is supposed not to be at equilibrium. However the equilibrium can be reached by moving in turn each of the rigid bodies (robot poses), according to the force that acts on it  $f_i = \nabla_{x_i} e_i$ . Repeating this procedure for all of the masses in the network, leads to a lower energy state, and therefore to a minimal error configuration. This procedure requires a quadratic number of iterations, for reducing the error term of a constant factor; therefore it poses some computational problem. Frese *et al.* [Frese & Duckett, 2003], proposed a modified version of the relaxation algorithm, that relies on a multi-resolution approach that is able to obtain an approximated solution in linear time (with respect to the number of nodes in the network).

Dellaert [Dellaert, 2005b] considered the optimization problem in terms of information smoothing. The minimization is viewed as the solution of a Least Square Problems, which is addressed using the QR factorization. As stated in his paper, exploiting the structure of SLAM problem the resulting matrices are exactly sparse, thus yielding to an efficient batch algorithm. Kaess *et al.* [Kaess, Ranganathan, & Dellaert, 2007] extended those ideas in an incremental information smoothing algorithm. In that work, they provided a full solution to SLAM, combining incremental smoothing with data association.

Olson *et al.* [Olson, Leonard, & Teller, 2006] addressed the problem by using gradient descent on a network described in a form which allows for efficient analytical updates. The method was further improved by Grisetti *et al.* [Grisetti *et al.*, 2008]. The authors use a different parametrization of the nodes in the network that takes into account the topology of the environment, resulting in a faster convergence. Graphical SLAM [Folkesson & Christensen, 2004] builds a graphical model of the smoothing problem. It optimizes the graph by defining an energy function for each node and then minimizing this energy.



Whereas these methods mainly focus on estimating the most likely configuration of the map, they leave open how to estimate the uncertainty of the solution. To the best of our knowledge, the only approach which computes both the ML configuration of the nodes and their marginal distribution has been proposed by Ranganathan *et al.* [Ranganathan, Kaess, & Dellaert, 2007]. They model the smoothing problem as a Gaussian Markov random field (GMRF) and use loopy belief propagation on this model. The covariances computed with their algorithm, however, are either overconfident, if using loopy belief propagation, or too conservative, if using loopy belief propagation over a spanning tree. An exact algorithm for those covariances can be found in the work of Kaess *et al.* [Kaess, Ranganathan, & Dellaert, 2007]. Their algorithm, however, is guaranteed to be efficient only in the case of band-diagonal matrices, and can be more expensive for general sparsity patterns.

In general, optimization based approaches represent an attractive choice for solving the problem, if associations among landmarks or among robot poses are known, since they are able to correct the pose in the past. However, the complexity which depends on the number of robot poses to optimize can make this methods not suitable for real time applications. Moreover, the problem of how to correctly determine the relations among poses is not straightforward.

### 3.3.3 Expectation Maximization

Expectation Maximization algorithm can also be considered as an optimization algorithm. Thrun *et al.* [Thrun, Fox, & Burgard, 1998] propose a probabilistic formalization of the SLAM problem, which does not address the data association. The mapping problem is formulated by exploiting the static world assumption, in terms of the joint belief over positions and maps:

$$p(x_t, m) = \eta_t p(z_t | x_t, m) \int p(x_t | u_{t-1}, x_{t-1}, m) p(x_{t-1}, m) dx_{t-1} \quad (3.6)$$

which involves the maximization of a function in a  $\dim(m) + \dim(x)$  space, which is a hard problem. By managing Equation 3.6 it is possible to obtain the posterior in a closed and factored form, suitable for the application of EM algorithm:

$$p(x_t, m) = \eta_t p(m) \int \int \cdots \int \prod_{\tau=0}^t p(z_\tau | x_\tau, m) \prod_{\tau=1}^t \int p(x_\tau | u_{\tau-1}, x_{\tau-1}, m) dx_1 dx_2 \cdots dx_t \quad (3.7)$$

By considering the robot path  $x_{0:t}$  as missing variables it is possible to maximize Equation 3.7, through the Expectation Maximization algorithm. The expectation step computes the likelihood of a robot path given the sensor readings and a known map, while the maximization step finds the optimal map given the likelihood of the path and the previous step map estimate. This is a very difficult problem, but in [Thrun, Fox, & Burgard, 1998] it has been straightforwardly solved by considering the cells of the map as independent. The occupancy probability of a cell is evaluated according to a frequency based update rule, and the pose likelihood. The EM approach in mapping and localization has shown to be very robust and able to map environments that were previously considered hard; its main problem is the computational complexity that grows with the length of the path, for the E steps, and with the size of the environment for convergence. Moreover this approach has the same weakness as the underlying estimation algorithm. Since EM performs the estimate through hill climbing

the posterior over maps and poses, if the initial estimate is not good enough, it can converge to a wrong local minima.

### 3.3.4 Kalman Filter

The first attempt to solve the SLAM problem was performed using the Extended Kalman Filter. The use of EKF imposes the following strong assumptions:

- the dynamic of the robot has to be linearizable
- the state distribution and the measurement error are assumed to be Gaussian.
- the map is represented by a set of distinguishable landmarks:  $m_1, \dots, m_k$ , whose uncertainty is described through a Gaussian.

The modeled system state can be partitioned in two components: the robot location  $x$  and the absolute landmarks position. The observed variables  $z = (m_1 \dots, m_k)^T$  are the locations of the landmarks with respect to the robot reference system. In general, the system state dimension grows as the system evolves, due to the incorporation of new landmarks. In order to work correctly, Kalman Filter based SLAM requires all those hypotheses to hold. Due to its simple formulation the structure of the solutions of the KF based approaches has been deeply investigated. Newmann *et al.* [Newmann, 1999; Gamini Dissanayake *et al.*, 2001] gave a proof of the linear SLAM problem convergence, under known data associations. In practice the error estimate is bounded by the initial position error and the first landmark sight.

Despite these ideal characteristics, using an EKF for solving the SLAM problem presents some shortcomings when the EKF hypotheses fail, or when the correspondences are not known. In this second case, it is possible to overcome the problem by using some heuristic such as the nearest neighbor principle. However, in environments crowded of landmarks such a heuristic, is prone to failure, and can cause the filter to converge to the wrong solution.

More robust data association strategies have been proposed for avoiding the overconfidence that shows up with the nearest neighbor, like the Joint Data Association by Neira *et al.* [Neira & Tardós, 2001]. Hähnel *et al.* [Haehnel *et al.*, 2003]. proposed a theoretical approach which is able to track several map hypotheses using an association tree. Each node in the tree represents an association hypotheses, while the leaves are the candidate maps. The tree grows as the robot moves, by considering only the most likely node. When a failure is detected, a backtracking occurs in order to consider a different sequence of associations that makes the observation consistent with respect to the map. However, the necessary expansions of this tree can make the approach unfeasible for real-time operation.

However, even if a robust data association is available, the problem of the landmark maps still exists. A landmark representation requires the ability to detect a series of spatially located features. This assumption holds if some knowledge about the environment is a priori available, and some feature extractor is used. Alternatively, the environment has to be structured by the insertion of artificial landmarks.

Frese *et al.* [Frese & Hirzinger, 2001] made an analysis of the SLAM process, and showed that the linear assumption in the Kalman filter introduces systematic errors. Julier and Uhlmann [Julier, Uhlmann, & Durrant-Whyte, 1995] showed how to lessen the effects of the nonlinearities in the motion model through the use of the UKF.

One of the major problems of this kind of algorithm seems to be the computational complexity, which depends on the dimension of the system state space to track. Such a dimension



is proportional to the number of landmarks  $n$ , and the filter requires  $O(n^3)$  for each update. This problem has been addressed by several approximated approaches, whose key concept is to exploit the weak dependence among landmarks far from each other. In the following of this section, some of the most common techniques are discussed.

### 3.3.5 Information Filter

In order to reduce the complexity of Kalman Filter based SLAM, some authors have investigated the use of its dual: the information filter. The key idea of this family of approaches can be stated as follows: if the covariance matrix  $\Sigma$  of a system can be dense, the information matrix  $\Lambda = \Sigma^{-1}$  may be sparse, or its entries can be small. When such a matrix is sparse, using an information filter the predictions and the updates can be done in linear time. In a work by Thrun [Thrun *et al.*, 2004], it is proposed to approximate the information filter solution, for speeding up the update. The key idea can be stated as follows: if the covariance matrix  $\Sigma$  of a system can be dense, the information matrix  $\Lambda = \Sigma^{-1}$  may be sparse, or its entries can be small. When such a matrix is sparse, using an information filter, the predictions and the updates can be done in linear time. At each time SEIF approximates the small entries in  $\Lambda$  with 0, and obtains a constant time approximated filter update. It is worth to observe that arbitrarily changing the entries of the precision matrix is not trivial because the resulting matrix must be still positive semidefinite. The solution adopted by SEIF is to approximate the map by taking a constant number of gradient descent steps in the probability density defined by  $\Lambda$ , and to approximate the marginal landmark covariances needed for data association using the conditional covariance. Subsequently, Eustice *et al.* [Eustice, Walter, & Leonard, 2005] computed the error boundaries introduced by SEIFs during the sparsification operation. In a more recent work, Eustice *et al.* [Walter, Eustice, & Leonard, 2007] showed that the information matrix is *exactly* sparse if a delayed state representation is used.

However, in spite of reduced time complexity, information filter approaches suffer from representational issues. Recovering a map requires inverting  $\Lambda$ , as the mean is encoded into the information vector  $\eta = \Lambda\mu$ . This represents a problem, not only because the map is inaccessible, but also because it is needed for system linearization. Thrun solution consists to approximate the map, by taking a constant number of gradient descent steps in the probability density, and to approximate the marginal landmark covariances needed for data association using the conditional covariance. Eustice, instead, solves a sparse linear systems by using the Conjugate Gradients method.

A work which is in between the Kalman filter and the information filter is presented by Paskin *et al.* [Paskin, 2003], where the use of graphical models for performing selective updates on landmark submaps is proposed. The graphical models can be seen as graphs in which each node represents a state variable (a landmark or a robot pose). Edges among nodes express conditional dependence among them. The observations affect the graphical model, resulting in the addition of nodes and edges. Marginalizing out a variable (for instance a past time position), results in the suppression of a node in the graph and in the introduction of additional edges among the neighbors of the suppressed node. Within the Thin Junction Tree framework the statistical parameters of the estimate are described through the use of a tree view of the graph. The complexity of the updates is bound through the "thinning" operation that prunes connections in the tree associated to weakly dependent variables. The behavior of this operation is similar to the sparsification introduced by the SEIF framework. Although the description of the involved variables through trees makes it difficult to represent loop

situations, experimental results have shown the effectiveness of the approach.

### 3.3.6 Rao-Blackwellized Particle Filters

In a work by Murphy [Murphy, 1999] Rao-Blackwellized particle filters (RBPF) have been introduced as an effective means to solve SLAM problem, due to their ability of tracking multiple map and robot pose hypotheses. Each particle in a RBPF represents a possible robot trajectory and a map. The framework has been subsequently extended by Montemerlo *et al.* [Montemerlo *et al.*, 2002; Montemerlo & Thrun, 2003] for approaching the SLAM problem with landmark maps.

For converging to the correct solution, a RBPF-based SLAM algorithm, requires that at least one particle carries a path close enough to the true one. Based on this consideration several authors conjuncture that the number of particles required for building a consistent map grows exponentially with the size of the largest unknown cyclic traveled path [Frese & Hirzinger, 2001; Paskin, 2003]. In the context of grid maps, RBPF have been also used by Eliazar and Parr [Eliazar & Parr, 2003], Hähnel *et al.* [Hähnel *et al.*, 2003a] and Grisetti *et al.* [Grisetti, Stachniss, & Burgard, 2005] for building grid maps using a mobile robot equipped with a laser range finder. Whereas the first work describes a memory efficient map representation that allows to represent a high number of map samples, the second presents an improved motion model that reduces the number of required particles.

In practice, a Rao-Blackwellized particle filter for SLAM is a particle filter in which each particle represents a possible robot trajectory, and the map obtained from the integration of the sensor readings along that trajectory. A deeper explanation of the Rao-Blackwellized framework for mapping is given in Chapter 4.

### 3.3.7 Topological Approaches

Topological approaches update a graph representation of the environment. Such a family of techniques is suitable for environments in which it is easy to define the concept of “place”. Estimating a graph rather than a complex metric structure has significant computational advantage. Moreover, it is possible to perform high level reasoning on estimate.

Choset *et al.* [Choset, , & J., 2000] proposed to use the Voronoi graph as a topological representation. A place is characterized by a node of such a graph, and the robot localizes itself in the Voronoi nodes. Kuipers *et al.* [Kuipers & Byun, 1991] proposed the Spatial Semantic Hierarchy, for representing several models of spatial knowledge within a hierarchy of ontologically different layers: from the robot control level (path planning), to the topological reasoning level. The possible topological maps are constructed by combining the expected outcome of the actions, and pruned by the observations of “places”. The possible topologies can be arranged into a tree. By enforcing additional constraints on the generated graphs the feasible solutions can be pruned. Savelli *et al.* [Savelli & Kuipers, 2004] proposed the use of planar constraints in the graph generation that substantially reduces the number of feasible hypotheses at each step.

Martinez Mozos and Burgard [Mozos & Burgard, 2006] present an approach to create topological maps from geometric maps. The approach utilizes AdaBoost, a supervised learning algorithm, to classify each point of the geometric map into semantic classes. They then apply a segmentation procedure based on probabilistic relaxation labeling on the resulting classifications to eliminate errors. The topological graph is then extracted from the individual

different regions and their connections. In this way, they obtain a topological map in the form of a graph, in which each node indicates a region in the environment with its corresponding semantic class (e.g., corridor, or room) and the edges indicate the connections between them.

In his PhD thesis, Ranganathan [Ranganathan, 2008] presents the concept of Probabilistic Topological Maps (PTMs), a sample based representation that approximates the posterior distribution over topologies given available sensor measurements. In [Ranganathan, Menegatti, & Dellaert, 2006] it has showed that the PTM can be obtained by performing Bayesian inference over the space of all possible topologies and provides a systematic solution to the problem of perceptual aliasing in the domain of topological mapping. In [Ranganathan & Dellaert, 2006] they propose a particle filtering algorithm for topological mapping based on the PTM framework using a laser scanner as sensor. The use of particle filters makes our algorithm incremental, as opposed to their previous work on PTMs that used Markov Chain Monte Carlo (MCMC).

Dudek *et al.* [Dudek *et al.*, 1991] address the problem of robotic exploration of a graph-like world, where no distance or orientation metric is assumed of the world. The robot cannot measure distances, and it does not have a compass. They demonstrate that this exploration problem is unsolvable in general without markers.

### 3.3.8 Hybrid Approaches

The techniques described so far rely on the use of a single representation and a single estimation algorithm. Some good results have been obtained by using hierarchical map representations and combining different estimation algorithms. For this reason in the rest of this document we refer to this family of techniques also as "integrated" approaches.

In the works by Gutmann *et al.* [Gutmann & Konolige, 1999] and Thrun *et al.* [Thrun, Burgard, & Fox, 2000], effective ways for integrating an optimization method with a maximum likelihood based approach are proposed. The core idea is to carry on the estimate using a scan matching procedure, and to use a Bayesian estimator to track the robot pose uncertainty.

In [Gutmann & Konolige, 1999] a Kalman filter is used for estimating the robot pose. As long as the robot moves through the environment, a topological map is built according to the scan matcher results. Possible loop closures are detected by checking for a possible matching of a local map within the previously built map. Such a local map is constructed by running a scan matching algorithm on the most recent scans. The candidate search area consists in the locations covered by the uncertainty ellipses resulting from the Kalman filter. When a matching that can close the loop is found, an optimization procedure (see [Subsection 3.3.2](#)) is run for all the poses in the loop, in this way a consistent estimate is obtained.

In [Thrun, Burgard, & Fox, 2000] the map is represented by storing the acquired laser scans, and the robot path. This representation can be seen as a patch representation in which each patch contains a single scan, while the robot trajectory is the graph of relations. A single map estimate is built through a scan matcher. The robot pose is estimated through a particle based localization algorithm. The effect of a loop closure is that the robot localizes itself in a previously seen part of the environment. This introduces a new constraint among robot locations in the robot path. At this point the trajectory along the loop is revised, enforcing the consistency with the newly introduced loop constraint. The backward revision is performed by uniformly distributing the error along the loop, and refined by a gradient descent based search.

These methods combine both the advantages of a maximum likelihood approach, and an optimization method. They are extremely fast in the average case, while being able to effectively close loops. The shortcoming of these algorithms is that they are able to track only one topological hypothesis, that if wrong, leads to an incorrect map estimate.

Howard *et al.* [Howard, 2004] proposed the use of a hierarchical metrical map, in which the loop closures are delayed. The idea is to represent a global map as a set of patches. These patches are arranged according to a graph, while the loop closures can be indefinitely delayed. At each point in time the robot operates in a part of the environment, where the map is locally consistent. This is achieved by considering only the patches which are reachable through a fixed-depth breadth-first visit of the global graph, starting from the patch in which the robot is located. The graph topology reflects the time order of the perceptions about different parts of the environment. When the robot moves, the graph node in which the robot is located in the graph is updated accordingly. If a loop closure is assessed, the conditions of the loop closure are propagated along the graph, and a global map is obtained by a non linear optimization.

Bosse *et al.* [Bosse *et al.*, 2003] presented the Atlas Framework for achieving a globally consistent map, combining different submaps. This is done by keeping a patch based representation, of the map. Each patch retains its robot pose estimate, while it is always possible to obtain an intra-patch pose posterior by propagating the estimates according to the graph structure. The framework explicitly handles the addition of a new patch, and the loop closing events through a state machine whose transitions are triggered by a map matching process. Although this approach does not directly provide a globally consistent map, it retains the topological consistence during the exploration, and it is suitable for real time operation. The final map can be obtained by an optimization procedure.

Lisien *et al.* [Lisien *et al.*, 2003] presents a hierarchical approach to the SLAM problem, by merging a topological and a metrical representation. A generalized Voronoi graph is used for partitioning the metric space in regions. Within each region a metric map can be represented. The choice of the Voronoi graph as a topological skeleton allows to unambiguously detect the nodes, and can be used for navigating among the different metric maps.

Modayil *et al.* [Modayil, Beeson, & Kuipers, 2004] propose the use of a mixed topological/metric representation for factoring out the data associations ambiguities, and reasoning about the possible topologies. Furthermore, they propose an approach for obtaining a global map from a topological description, and a scan matched map. This is done through an RBPF. The trajectories are sampled from an informed proposal distribution which considers also the topological skeleton. The final map is generated through an optimization technique.

Estrada *et al.* [Estrada, Neira, & Tardos, 2005] present a hierarchical mapping method that allows to obtain accurate metric maps of large environments in real time. The lower (or local) map level is composed of a set of local maps that are guaranteed to be statistically independent. The upper (or global) level is an adjacency graph whose arcs are labelled with the relative location between local maps. An estimation of these relative locations is maintained at this level in a relative stochastic map. The loop closing method, while maintaining independence at the local level, imposes consistency at the global level at a computational cost linear with the size of the loop.

Kouzobov and Austin [Kouzobov & Austin, 2004] present a topological/metric approach to solving the Simultaneous Localization and Mapping problem. The map is represented as a graph - nodes are local map frames, and edges are transformations between adjacent map frames. The underlying local mapping algorithm is FastSLAM. The local maps and transformations are modelled by sets of particles. There is no global map frame, each map's

uncertainties are restricted to its own map frame. The loop dosing is achieved via efficient map matching. We demonstrate our algorithm running in real-time in an indoor environment using a laser range sensor.

In the work of Tomatis *et al.* [Tomatis, Nourbakhsh, & Siegwart, 2001] the metric and topological paradigms are integrated in a hybrid system. A global topological map connects local metric maps, allowing a compact environment model, which does not require global metric consistency and permits both precision and robustness. Furthermore, the approach handles loops in the environment during automatic mapping by means of the information of the multimodal topological localization.

Blanco *et al.* [Blanco, Fernandez-Madrigal, & Gonzalez, 2008] introduces a new approach to Simultaneous Localization and Mapping (SLAM) which pursues robustness and accuracy in large-scale environments. Their approach is based on the reconstruction of the robot path in a hybrid discrete-continuous state space, which naturally combines metric and topological maps. The novelty of the approach lies in the use of a unified Bayesian inference approach both for the metrical and the topological parts of the problem and in the analytical formulation of belief distributions over hybrid maps.

A peculiar work is presented by Newmann *et al.* [Newman, Cole, & Ho, 2006a] They describe a 3D SLAM system using information from an actuated laser scanner and camera installed on a mobile robot. The laser samples the local geometry of the environment and is used to incrementally build a 3D point-cloud map of the workspace. Sequences of images from the camera are used to detect loop closure events using a novel appearance based retrieval system. The loop closure detection is robust to repetitive visual structure and provides a probabilistic measure of confidence. The images suggesting loop closure are then further processed with their corresponding local laser scans to yield putative Euclidean image-image transformations.

## 3.4 Discussion

In the previous Section, we presented some of the relevant approaches that have been proposed for SLAM. Although effective solutions for medium scale and static environments have been proposed, a general algorithm which is able to produce accurate maps of arbitrarily large environments is still missing.

Usually, in autonomous applications, a localization and mapping engine is used in conjunction with a path planning algorithm. In order to work in such scenario, a SLAM algorithm needs to fulfill some properties. It has to provide an up to date map as soon as the new readings are available, without the need of reprocessing the whole sensing history. The map has to be built *incrementally* and its estimation should be stable enough so that planning results on previous maps are still valid. Moreover, the algorithm should be able to correctly handle loop closing situations, and produce a *globally consistent map*, useful for navigation.

In the remaining of this Section, we will explain and describe some of the issues that have to be solved to have a reliable SLAM solution that can be used in real-world scenarios.

### 3.4.1 Recovery From Failure

Recently, many researchers are studying convergence and consistence conditions for filtering solutions to SLAM. One property seems to be emerging:

whichever estimator is used, there is always an environment it is unable to estimate.

Two main reasons justify this statement: density coverage and data association.

Density coverage stems from the fact that the real pose of the robot has to be within the probability density estimated in the filtering process. If this is not the case, no convergence criterion is met and every filter tends to diverge. In analyzing this aspect, a fine distinction has to be done between particle filters and Gaussian-based filters (Kalman and information). As for particle filters, the coverage problem arises when the number of particles is not “enough” to represent the probability density over the pose trajectories. In this case, it is possible that no particle is close to real pose and the convergence conditions on the distribution supports fail. Up to now, no methods have been proposed to estimate the good number of particles for a given environment, neither adaptive sampling schemes have been developed. As for Gaussian-based filters, the main problem consist in the inherent non linearity of the system. This leads to over confident estimates, resulting in the real pose being outside the  $3\sigma$  bounds of the estimated covariance. Some methods have been proposed to “inflate” the estimated covariance, but this leads to conservative and less accurate estimates. Another possibility is to use the unscented Kalman filter [Julier, Uhlmann, & Durrant-Whyte, 1995]. However, this mitigates, but not completely solves the problem [Julier & Uhlmann, 2001]

Data association is another source of algorithm failure. Wrong data association leads not only to a filter divergence, but it causes all optimization methods to fail. Wrong associations are usually made when considering only geometric relationships between landmarks. When only geometric properties are used, an overconfident estimate of the robot pose is not able to detect the correct association, while a conservative one makes the problem more difficult in cluttered environments due to the enlarged search space. To the best of our knowledge, only Hähnel *et al.* in [Haehnel *et al.*, 2003] explicitly addresses the problem of recovering from wrong associations. This is done by tracking the possible associations in a tree. A trivial construction of this tree requires to increase the depth of the tree each time a new observation is made, by considering all the possible ambiguous matches. However, this approach is clearly intractable and therefore does not represent a real solution to the problem. The original framework proposed in [Haehnel *et al.*, 2003] does not specify how to construct such a tree, while it proposes a theoretical framework. Recently, some works on appearance based data association have been proposed [Gil *et al.*, 2006; Cummins & Newman, 2007]. Moreover, a new branch of research is coming out: appearance-based SLAM [Cummins & Newman, 2008].

### 3.4.2 Dynamic Environments

The static world assumption under which most of the SLAM techniques have been developed is often not valid. Real world is dynamic: people and other robots are moving; furniture changes; doors can be open or close. Objects move according to different dynamics, and a lifelong SLAM algorithm should be able to handle that. The different classes of dynamics can be classified according to the “speed” of change they present

**Static:** objects which cannot be moved, like walls or corridors.

**Almost Static:** objects whose configuration can change over time, but it is in general fixed, as for example book shelves.



**Fixed Dynamic:** objects whose configuration in the environment can change between a limited set of configurations, like doors or elevators.

**Highly Dynamic:** objects whose position can change fast during data acquisition, like people, cars or other robots.

The first two categories can be merged together if the robot is used for short or medium time operations. In this conditions, almost static objects are really static. The problem of fixed dynamic object has been addressed by Stachniss *et al.* [Stachniss & Burgard, 2005]. They introduced a Rao-Blackwellized framework which is able to represent multiple configurations for a single map, and they proposed an approach for localizing in such a map structure. While this approach is attractive, it is unable to cope with highly dynamic objects, which can be present in the scene while acquiring the data. To this end Schultz *et al.* [Schulz *et al.*, 2003] proposed the use of sample-based joint probabilistic data association filters, for tracking the dynamic objects and neglecting the corresponding observations. This approach suffers from the problem that the shape of the dynamic objects should be known in advance, since the tracking algorithm relies on a feature extractor. Hähnel *et al.* [Hähnel *et al.*, 2003b] presented an offline, EM based approach for filtering moving points in range data. The algorithm maximizes the likelihood of the data using a hidden variable expressing the nature of the points (static or dynamic). However, this is an offline algorithm which is not suitable for real time applications.

Wang *et al.* [Wang *et al.*, 2007] defined an integrated solution for the mapping and tracking problem: static points are used for mapping while dynamic ones for tracking. The detection and segmentation of dynamic points is based on data differencing and consistency-based motion detection [Wang & Thorpe, 2002]. Points are classified in static and dynamic and clustered in segments. When a segment contains enough dynamic points is considered dynamic. Montesano *et al.* [Montesano, Minguez, & Montano, 2005] improved the classification procedure described in [Wang *et al.*, 2007] by jointly solve it in a Bayesian framework. Moreover, they integrated the mapping and tracking within a path planner for indoor navigation.

Although, most of these approaches focuses on how to track the different objects under different hypothesis, the detection part is mainly based on different heuristics. The main technique used is based on map differencing, where points are considered dynamic if there is some inconsistency between two consecutive scans (or a map and a scan). Moreover, the detection routine is only able to observe the actual position of the object (given a stable reference point) and the velocities are computed by the tracking algorithm.





## **Part I**

# **Introspective Analysis of Filtering Solutions to Simultaneous Localization and Mapping**



## Chapter 4

# Rao-Blackwellized Mapping

In this chapter we introduce the general Rao-Blackwellized particle filter framework for the Simultaneous Localization and Mapping problem. This framework is, however, only partly specified. The user needs to specify how to draw samples and perform the resampling algorithm. The former option is made by choosing the so called *proposal distribution*, the latter is mainly based on some heuristics. Moreover, a map representation needs to be chosen. This family of filters have been applied to both landmark and grid maps. However, their inner representation has to be chosen carefully, in order to reduce the memory requirements.

In the following, we first present the general RBPF framework for mapping. Subsequently, we discuss the different choices regarding the representation, the proposal and the resampling strategies. We conclude the chapter highlighting a novel analysis that will be addressed in the next chapter.

### 4.1 Rao-Blackwellized Particle Filter for Simultaneous Localization and Mapping

Classical particle filtering can be very inefficient in this scenario. This is due to the high dimensional space of the map. The key observation which makes it tractable in this case is that if the robot trajectory is known, then the posterior over the map can be factored analytically [Murphy, 1999]. This phenomena is known in the statistical literature as Rao-Blackwellization and the key idea of Rao-Blackwellized particle filters for SLAM is to estimate a posterior  $p(x_{1:t} \mid z_{1:t}, u_{0:t})$  about potential trajectories  $x_{1:t}$  of the robot given its observations  $z_{1:t}$  and its odometry measurements  $u_{0:t}$  and to use this posterior to compute a posterior over maps and trajectories:

$$p(x_{1:t}, m \mid z_{1:t}, u_{0:t}) = p(m \mid x_{1:t}, z_{1:t})p(x_{1:t} \mid z_{1:t}, u_{0:t-1}). \quad (4.1)$$

The joint space over robot trajectories  $\langle x, m \rangle$  and maps can be partitioned according to  $X^a \times X^b$ . The posterior over grid maps  $p(m \mid x_{1:t}, z_{1:t})$  can be computed analytically [Moravec, 1988], given the knowledge of  $x_{1:t}$  and  $z_{1:t}$ . To estimate the posterior

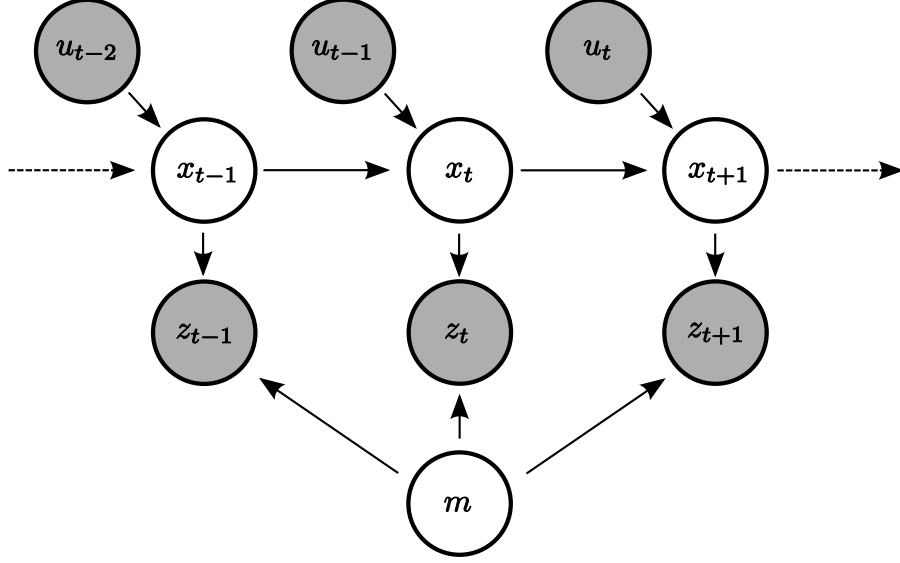


Figure 4.1: SLAM Problem seen as a DBN, the map is *static*. The observed variables are depicted in gray

$p(x_{1:t} \mid z_{1:t}, u_{0:t-1})$  over the potential trajectories, Rao-Blackwellized mapping uses a particle filter in which an individual map is associated to every sample. Each map is built given the observations  $z_{1:t}$  and the trajectory  $x_{1:t}$  represented by the corresponding particle. The trajectory of the robot evolves according to the robot motion and for this reason the proposal distribution is often chosen to be equivalent to the odometry motion model.

A Rao-Blackwellized SIR filter for mapping incrementally processes the observations and the odometry readings as they are available. This is done by updating a set of samples representing the posterior about the map and the trajectory of the vehicle. The filter performs the following four steps:

1. **Sampling:** The next generation of poses  $\{x_t^{(i)}\}$  is obtained from the current generation  $\{x_{t-1}^{(i)}\}$ , by sampling from a proposal distribution  $\pi(x_t \mid x_{1:t-1}^{(i)}, z_{1:t}, u_{t-1})$ .
2. **Importance Weighting:** An individual importance weight  $w^{(i)}$  is assigned to each particle, according to

$$w^{(i)} = \frac{p(x_{1:t}^{(i)} \mid z_{1:t}, u_{t-1})}{\pi(x_{1:t}^{(i)} \mid z_{1:t}, u_{t-1})} \quad (4.2)$$

$$\propto w_{t-1}^{(i)} \frac{p(z_t \mid x_t^{(i)})p(x_t^{(i)} \mid x_{t-1}^{(i)}, u_{t-1})}{\pi(x_t^{(i)} \mid x_{1:t-1}^{(i)}, z_{1:t}, u_{t-1})} \quad (4.3)$$

3. **Map Estimating:** for each pose sample  $x_t^{(i)}$ , the corresponding map estimate  $m_t^{(i)}$  is computed. The computation is based on the trajectory of the robot  $x_{0:t}^{(i)}$  and the history

of observations  $z_{1:t}$  according to  $p(m_t^{(i)} | x_{1:t}^{(i)}, z_{1:t})$ .

4. **Resampling:** Particles with a low importance weight  $w$  are replaced by samples with a high weight.

Note that in [Equation 4.2](#) we assumed the proposal to be suitable for sequential estimation. This enables to incrementally estimate the trajectories and the particle weights.

Moreover, selecting a sequential proposal allows to perform the map estimation in an incremental way. This is done by storing in each particle a map instance, which depends on the robot trajectory. When a new trajectory is computed from the old one, the particle map is updated with the new reading. This consideration allows to focus only on the last pose  $x_{t-1}^{(i)}$  and the last map  $m_{t-1}^{(i)}$  instead of the sequence of past trajectories and observations  $x_{1:t-1}^{(i)}, z_{1:t-1}$ . Representing a map and a pose for each particle, rather than a trajectory, has the significant advantage that the time required for a filter update does not grow with the length of the trajectory. On the other hand, this introduces more memory requirements, as the map dimension can grow over time.

## 4.2 Specification of Filter Components

Whereas the general framework of RBPF mapping can be summarized by the four steps previously described, it leaves open the choice of a proposal distribution, and the choice of whether or not to resample. In this chapter, we discuss how the choice of these two components can affect the performance of the algorithm. Moreover we discuss some of the approaches proposed for compactly representing a set of grid maps.

### 4.2.1 Proposal Distribution

The choice of a correct proposal distribution is a key issue of the RBPF. While the convergence in the theoretical framework is assured only for an infinite number of particles, in practical situation we see that the filter converges if the particle set is always exploring the meaningful part of the posterior density. The noise introduced in the sampling step has the main purpose of exploring the space of the possible robot trajectories generated by a robot motion/perception. In order to achieve good estimate, a proposal distribution should, on one side, cover the space of the robot feasible trajectory and, on the other side, be selective enough so that no computation is wasted on portion of space of low probability.

Moreover, since the maps depend on the robot trajectory, adding an excessive amount of noise to each time step can decrease the quality of the map estimate. Such a map estimate is then used in computing the next generation of samples/weights, and its quality loss results in a worsening of the subsequent filter evolution. This effect becomes evident when the robot navigates for a long time in a known environment: in the initial stage the robot acquires the map of the environment, and a correct map/trajectory distribution is assessed. From that point in time, the subsequent navigation/map-building has the effect of worsening the estimate, due to the errors introduced in the sample generation stage.

The original proposal distribution introduced by Murphy [[Murphy, 1999](#)] consisted by the raw odometry motion model  $p(x_t | x_{t-1}, u_t)$ . While this can be seen as a natural choice, as it is also the proposal for a localization problem, it also requires a high number of particle,

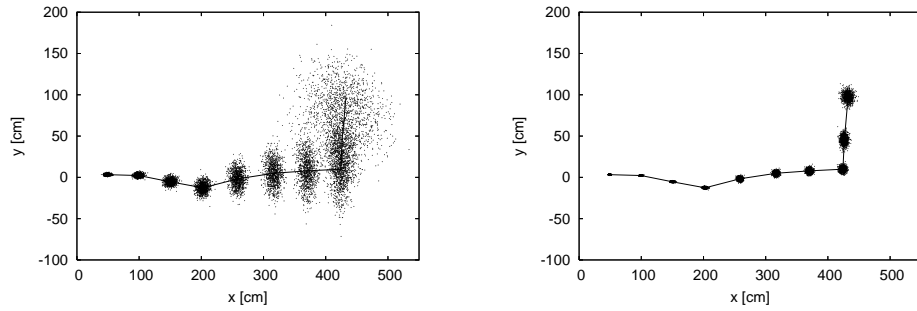


Figure 4.2: This picture illustrates two different proposal distributions. The robot moves along the black line. Left: samples generated according to the odometry motion model. Right: samples generated according to the scan-matcher error model.

thus limiting the use of the approach to small size environments. This limitation is mainly due to the memory requirements, as every particle needs to keep a map of the environment. The approach of Eliazar and Parr [Eliazar & Parr, 2003] alleviates this problem by using a better map representation (see Subsection 4.2.3 for more details).

In the domain of landmark maps, the memory problem is less evident, thereby in the approach by Montemerlo *et al.* [Montemerlo *et al.*, 2002] known as FastSLAM, the same proposal was used. This approach was then extended by the authors implementing FastSLAM 2.0. The main difference from this approach and the previous one was the choice of an informed proposal distribution  $p(x_t | x_{t-1}^{(i)}, u_t, z_t, m_{t-1}^{(i)})$ . This allows for a great reduction of the number of required particles, and made it also possible to give a proof of convergence, under assumptions similar to the ones made about the convergence of KF based approaches.

Within the domain of grid maps Hähnel *et al.* [Hähnel *et al.*, 2003a] proposed to use the error model of the scan-matching problem as a proposal distribution. This reduces the number of particles required of one order of magnitude, with respect to approaches which are based on the odometry motion model. In Figure 4.2, the two proposal distributions are shown. However, the main weakness of the approach of Hähnel is that it uses a fixed proposal distribution. This fixed proposal does not take into account the possible scan matcher failures. In order to grant an adequate coverage of the feasible robot trajectories, the error model has been “inflated” by the introduction of artificial noise.

This weakness was then resolved by Grisetti *et al.* [Grisetti, Stachniss, & Burgard, 2005]. The key idea of their approach is that the informed proposal distribution cannot be fixed as it strongly depends on the map of the environment (see Figure 4.3). To this end, they developed an adaptive proposal by approximating the distribution around the maximum likelihood solution of the scan matcher by a Gaussian

$$p(x_t | m_{t-1}^{(i)}, x_{t-1}^{(i)}, z_t, u_{t-1}) \simeq \mathcal{N}(\mu_t^{(i)}, \Sigma_t^{(i)}). \quad (4.4)$$

where, for each particle  $i$ , the parameters  $\mu_t^{(i)}$  and  $\Sigma_t^{(i)}$  can be determined by evaluating the likelihood function for a set of points  $\{x_j\}$  sampled around the corresponding local maximum.

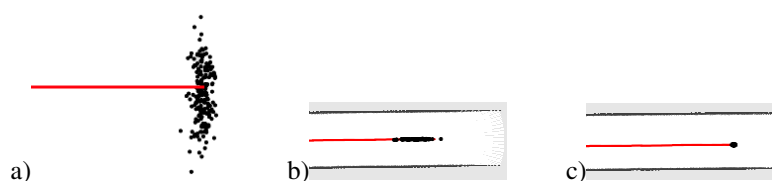


Figure 4.3: Particles distributions typically observed during mapping. In a featureless open space the proposal distribution is the raw odometry motion model (a). In an open corridor, the particles distribute along the corridor (b). In a dead end corridor, the uncertainty is small in all of the directions (c).

In featureless environment, the lack of meaningful structure does not allow a proper particle sampling and weighting. To overcome this situation, Grzonka *et al.* [Grzonka *et al.*, 2007] introduced a look-ahead proposal distribution. The pose prediction is computed, in each iteration, based on the  $k$  next sensory inputs instead of just one. These  $k$  measurements are used to better localize each particle within its own map. Concretely, for each mapping particle at time  $t$ , they draw  $l$  localization particles and localize them  $k$  steps ahead within their map. The resulting pose posterior at time  $t+k$  is then used to sample the successor pose of the mapping particle at time  $t$ . A similar idea has also been developed by Beevers and Huang [Beevers, 2007]. Their approach consists to draw new samples for the last  $L$  poses directly from the joint optimal block proposal distribution:  $p(x_{tL+1:t} | u_{1:t}, m_{t-L}, z_{tL+1:t}, x_{tL}^{(i)})$ . The basic idea is to sample from that distribution at each time step, replacing the most recent  $L$  poses of each particle with the newly sampled ones. The result is a particle filter that yields much better samples since future information is directly exploited by the joint proposal. Thus, degeneracies in the weights of particles are much less likely to occur.

A bad choice of the proposal affects not only the number of particles, but also the map quality. The phenomenon is explained in [Grisetti, 2006], performing the following simulated experiment: a mobile robot was moving along a loop in a medium size environment for a long time. The outcome of grid-based RBPf algorithms is compared by using two different proposal distributions: the uninformed scan-matching error distribution by Hähnel *et al.* [Hähnel *et al.*, 2003a], and the informed one proposed by Grisetti *et al.* [Grisetti, Stachniss, & Burgard, 2005]. After a while both the approaches generated a correct map. Then the quality of the estimate started to decrease. While using the proposal in [Hähnel *et al.*, 2003a] the quality of the estimate decreases after the robot repeats a few loops, in the map generated using [Grisetti, Stachniss, & Burgard, 2005], this phenomenon is less evident, as can be seen in Figure 4.4.

## 4.2.2 Resampling Strategies

An key issue in particle filtering SLAM is the consistency of the SLAM filter. This problem has been deeply analyzed by Bailey *et al.* [Bailey, Nieto, & Nebot, 2006] showing experimentally that in general, current particle filtering SLAM algorithms are inconsistent. A filter is inconsistent when the particle set does not cover the meaningful space of the true distribution, underestimating its own error.

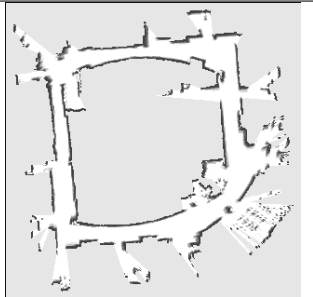
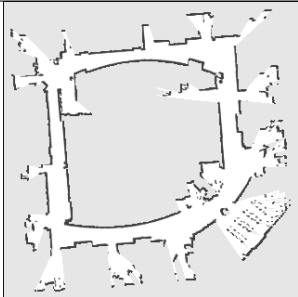
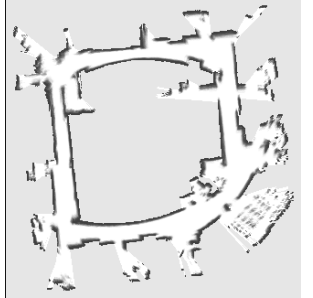
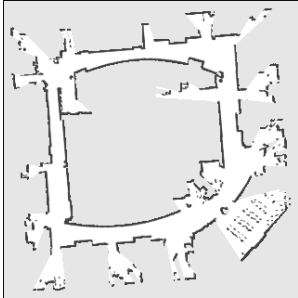
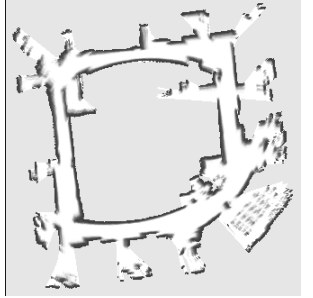
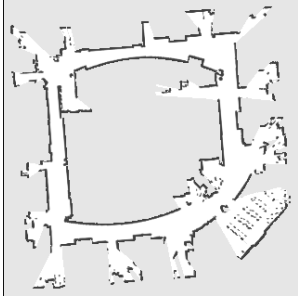
# of loops	[Hähnel <i>et al.</i> , 2003a]	[Grisetti, Stachniss, & Burgard, 2005]
5		
25		
50		

Figure 4.4: The RBPf results of a long navigation, using two different proposal distributions.



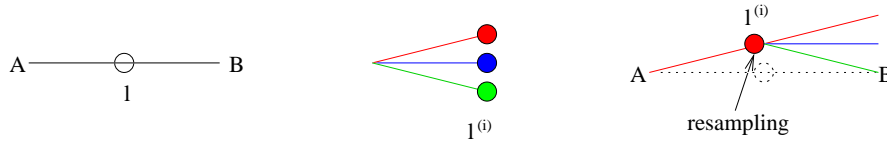


Figure 4.5: The effect of an unneeded resampling step. The left picture depicts the correct map, and trajectory. The robot moves from  $A$  to  $B$ , and it observes the white object at location  $l$ . In the middle picture, a possible 3 particles RBPf behavior is depicted. The red and the green trajectory are wrong, while the blue one is correct. In the right picture, a resampling occurs, and only the wrong red trajectory survives. The real trajectory and landmark are represented by the dotted lines. All the robot trajectories descending from the red one share the same wrong map. Notice that even if the final robot location of the green trajectory is correct, the error inherited by the red path persists.

This phenomenon is in large part due to degeneracies caused by frequent resampling, a problem known in literature with the name of *particle depletion* [van der Merwe *et al.*, 2000]. While degeneracy affects all particle filter, it is more severe and evident in the case of SLAM. Consider a resampling step in a typical particle filter, let say for a localization problem. Suppose we have a highly skewed weight distribution. After the resampling, only few particles survive highly underestimating the error. However, if the particles are close enough to the correct solution we can still recover from this situation thanks to the spreading due to the proposal distribution. Consider now the same situation, but in a SLAM context. The same resampling step at time  $t$  which replicates  $n$  times a particle causes the trajectory of that particle to be represented  $n$  times in the sample set. Since the map is constructed according to the trajectory, the maps of the  $n$  particles can differ only *after* time  $t$ , before they are the same. Here there is no possibility to recover from this bad situation, because of the incrementality of the proposal distribution. This fact is alleviated with the uses of future information in the proposal (see the previous Section), but still not solved. The phenomenon is illustrated in Figure 4.5.

To this end, some approaches have been explored to reduce this effect. The first attempt to solve the particle depletion problem is to improve the proposal distribution, such that the weight variance is minimized. This is obtained through the use of the so-called optimal proposal [Doucet, 1998].

Another possibility is to reduce the frequency of the resampling. In the basic particle filter, the resampling is performed at ever time step. However, Liu [Liu, 1996] showed that if all particle are approximately equally weighted, the efficiency of sample representation is decreased by a resampling. This idea has been introduced in the RBPf for SLAM by Grisetti *et al.* [Grisetti, Stachniss, & Burgard, 2005]. In their work, the authors used the effective number of particles  $N_{eff}$  as a measure of the quality of the particle estimate

$$N_{eff} = \frac{1}{\sum_{i=1}^N (w^{(i)})^2}.$$

In order to determine whether or not a resampling should be carried out they follow the

approach proposed by Doucet [Doucet, 1998]. Each time  $N_{eff}$  drops below the threshold of  $N/2$  where  $N$  is the number of particles, a resampling is performed.

The approach of Stachniss *et al.* goes in another direction. While improving the resampling strategy *prevent* the loss of particle diversity, their approach try to *recover* it after a loss. To do so, the authors store the “state” of the particle filter when the robot enters into a loop. After repeatedly traversing the loop to improve the map (a process normally resulting in loss of diversity), the filter state is restored by splicing the loop trajectory estimate onto each saved particle, effectively restoring the diversity of the filter prior to loop closing.

Similarly, Beevers and Huang [Beevers, 2007] propose an MCMC strategy to restore particle diversity after a resampling. Their approach lies in the *resample-move* framework, typical of target-tracking literature [Gilks & Berzuini, 2001]. The basic idea is to incorporate an MCMC step to “move” the trajectory of each particle over a fixed-lag time, by means of a Gibbs sampler. After the move, the particles are still distributed according to the desired posterior but degeneracies over the lag time have been averted. The samples are already approximately distributed according to the desired posterior before the move, so the usual burn-in time of MCMC samplers can be avoided.

### 4.2.3 Representation

A final problem affecting the grid based RBPF approaches is represented by the amount of memory required for storing a grid map for each sample. In the open source implementation of Hähnel *et al.* [Hähnel *et al.*, 2003a], the problem is solved by representing only the robot trajectories. Each time the particle map was needed (and this happens each time the likelihood has to be evaluated) it was constructed from scratch using the scans and the trajectory. Although this approach effectively reduces the memory requirements, it presents a serious computational problem: the time required for constructing the maps grows with the length of the trajectory.

Parr *et al.* [Eliazar & Parr, 2003] proposed a different schema, called distributed particle SLAM (DP-SLAM). The work is based on the consideration that the trajectories generated by a particle filter can be represented as a tree. The number of leaves of such a tree is the number of particles. Considering a path from a leaf to the root of such a tree, it is possible to detect a set of nodes in which a branch occurs. The authors propose to pack the map updates occurring among consecutive branches, and to prune the nodes unreachable from the leaves. In this way, the number of leaves of the “compressed” tree as well as its depth is bounded by the number of particles. The memory complexity is minimized, since shared portions of trajectories shares the same map updates. However, if all the trajectories differ from the beginning, it is still needed to store a grid map for each particle. Figure 4.6 depicts the trajectory tree and the compressed tree built from it.

## 4.3 Discussion

Rao-Blackwellized particle filters are indeed an effective tool for solving the Simultaneous Localization and Mapping problem. Since their introduction in the work of Murphy [Murphy, 1999] several improvements have been made, making the approach very competitive, also compared to other filtering techniques like the Kalman filter.

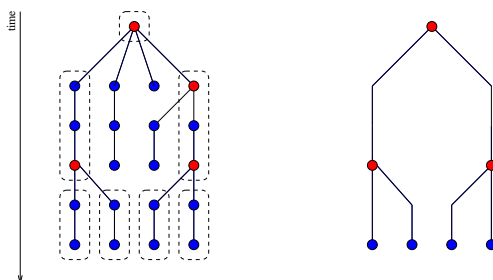


Figure 4.6: This figure depicts the trajectory tree, and the compressed DP-SLAM tree. On the right the original trajectory evolution is depicted, on the right the compressed tree obtained by i) deleting the nodes which are not reachable from the last level nodes, and ii) collapsing the paths among consecutive branches. The tree is bounded both in depth and in width by the number of particles.

First of all, their intrinsic capability of *factoring* the target distribution into a robot trajectory and map makes the approach very competitive for estimating grid maps. In Kalman filtering, for example, the estimation of grid maps is only possible through a delayed-state formulation [Eustice, Singh, & Leonard, 2005], where the map is represented through a set of points, or by explicitly represent every point as a landmark, making the inference intractable.

If a sequential proposal distribution is chosen, RBPFs allows to *incrementally* estimate the distribution over the robot trajectories. The  $t$  step generation of samples can be obtained from the previous step generation, by augmenting the each sample trajectory with the actual robot movement.

Since particle filters represent the probability distribution through a set of samples, their use is not restricted to applications in which the estimated distribution is Gaussian. This allows to estimate *multimodal* distributions, making them a useful tool for dealing with multiple hypotheses.

Using an RBPF does not require to *linearize* the observation or the motion model of the system, in fact relaxing most of the assumptions which are required to hold when using other approaches.

However, there are still some drawbacks in using RBPF for SLAM. In order to be consistent, the filter needs to cover the meaningful part of target distribution. As this distribution is approximate with a finite number of samples, the higher is the uncertainty in the distribution, the higher is the number of samples needed to approximate it. The particle filters have a theoretical justification as the number of particles goes to infinity. In practice such a number usually grow exponentially with the size of the biggest loop in the environment. However, observe that the SLAM problem in its general form is intrinsically exponential, because of the data association. In principle one wants to track all of the possible associations, whose number grows exponentially with the made observations. Despite these theoretical considerations the bounds in the uncertainty affecting both robot trajectory and observations allow to restrict the search space, thus enabling the development of algorithms which works in practical situations.

Since each particle represents its own map, a straightforward RBPF using a grid representation independently stores those maps, resulting in significant memory requirements. While effective solutions for representing a minimum number of maps have been proposed, the worst case still requires to independently represent a complete map for each particle.

The improvements made in the literature lessen the particle depletion and some memory issue, making the filter applicable in typical scenarios. However, those techniques only focus on one of the problem at time, trying to solve it. The analysis is made on the symptom of the problem (i.e. memory requirement, time complexity) and ad-hoc technique for solving it are adopted.

In [Chapter 5](#), we take the opposite way. We first make a deep analysis about how the process itself evolve and its structural properties. The exploitation of the process inner structure allows us to derive a fast approximated approach to Rao-Blackwellized mapping, which also uses a hierarchical patch based representation for compactly storing a distribution over maps and trajectories. We discuss how to effectively update such a structure performing approximated inference. The advantages introduced by the new trajectory/pose representation leads to a reduction of the memory requirements of several orders of magnitude with respect to approaches which individually stores the different grid maps. The approximated inference allows to update the structure one order of magnitude faster than previous approaches, making it possible to build real time maps of large environments.

# Chapter 5

## Introspective Filter

### 5.1 Introduction

In [Chapter 4](#), we introduced the Rao-Blackwellized particle filter for the Simultaneous Localization and Mapping problem. We described several state-of-the-art solutions to reduce the time and memory complexity of the classical filter.

In [[Grisetti, Stachniss, & Burgard, 2005](#)], they proposed an approach for computing an improved proposal based on both the current motion and the current observation. Each particle has its own grid map, and the proposal is computed for each particle. Such a computation requires to perform scan matching for each particle with respect to its own map, and to sample around the maximum provided by the scan matcher. Subsequently the likelihoods of this set of points are used for computing a Gaussian approximation of the proposal from which to draw the next robot pose hypothesis. Although using the above discussed improvements leads to a great reduction of the number of particles required for achieving a correct map, as the environment size grows, this number increases. Accordingly, the computational effort required by this approach can prevent its real time use, while the memory complexity can prohibit its use on actual computers.

In [[Eliazar & Parr, 2003](#)], an improved map representation is used to reduce the memory requirements of the filter. The work is based on the consideration that the trajectories generated by a particle filter can be represented as a tree. The number of leaves of such a tree is the number of particles. Considering a path from a leaf to the root of such a tree, it is possible to detect a set of nodes in which a branch occurs. The authors propose to pack the map updates occurring among consecutive branches, and to prune the nodes unreachable from the leaves. In this way, the number of leaves of the compressed tree as well as its depth is bounded by the number of particles. The memory complexity is minimized, since shared portions of trajectories shares the same map updates.

However, those approaches only analyze a small portions of the problem (the map in [[Eliazar & Parr, 2003](#)] and the particle number in [[Grisetti, Stachniss, & Burgard, 2005](#)]). In this Chapter, we perform a deeper introspective analysis of the whole problem, obtaining an approximate RBPF with

- a memory efficient way for representing a distribution over maps,

- a computationally efficient approach for updating the trajectory samples.

Experiments have shown that this approach is one order of magnitude faster than the approach discussed in [Chapter 4](#), and it requires several orders of magnitude less memory, making it possible to compute in real time even larger maps.

The filter improvement is done by first performing an analysis of its evolution during several runs in [Section 5.2](#). The goal of this analysis is to understand the relationships among the different situations of the mapping process, and the different filter behaviors. The data structures which allow to compactly represent trajectories and maps are presented in [Section 5.3](#), and in [Section 5.5](#) the heuristics for determining the mapping system situation are discussed. Subsequently, we devise an approach that allows to draw trajectory samples in an efficient way ([Section 5.4](#)). Finally, we discuss the details on how to update the above structures, by specifying how the basic operations of the algorithm are performed ([Subsection 5.3.1](#)) Experiments and results are discussed at the end of the chapter.

## 5.2 Introspective Analysis of Rao-Blackwellized Mapping

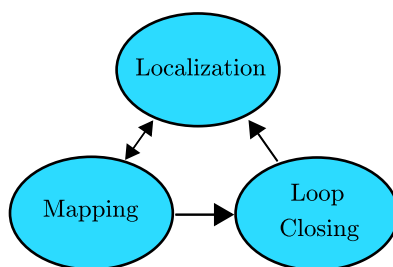


Figure 5.1: The mapping process automaton.

In this section we analyze the evolution of the filter. While the analysis is performed with a particular focus to a Rao-Blackwellized particle filter solution, its result is valid for any filtering approach to the problem. Let consider the examples depicted in [Figure 5.2](#) and [Figure 5.3](#). On the left part of the image, we can see that the robot is **exploring** previously unseen terrain and collecting new data. After a while, it reenters in a known terrain after a run in an unknown region. This situation, called **loop closing**, is shown in the center of the figure. Finally, on the right side of the image, we can see that the robot is navigating within a part of the environment already visited, thus it is in a **localization** stage.

In [Figure 5.1](#) the mapping process is depicted as a finite state automaton having three states. The filter updates have a different effect depending on the current state and the transition.

In each of these states the filter behaves in a different manner. In the following, we will describe the different situations and the behavior of both the RBPF and the KF. The results are then summarized in [Table 5.1](#) for the RBPF and in [Table 5.2](#) for the KF.

**Exploration** When the robot moves in unknown terrain, the overall uncertainty grows, since the spreading introduced by the motion model can only be bounded by the observa-

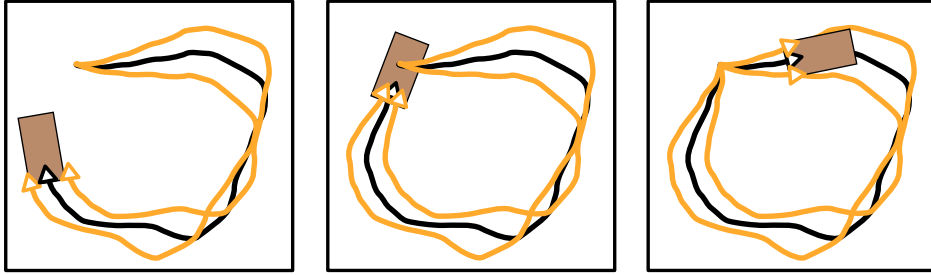


Figure 5.2: Introspective analysis of RBPF: exploration (left), loop closing (center) and localization (right). The triangles represent the different robot poses. The best particle is shown in black, while the others in dark yellow. The brown rectangle depicts the bounding box of the current reading

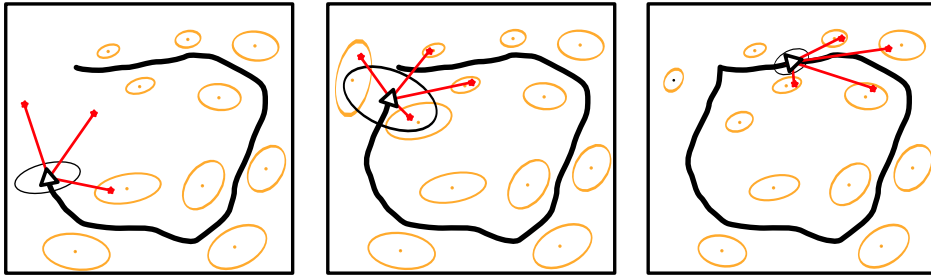


Figure 5.3: Introspective analysis of KF: exploration (left), loop closing (center) and localization (right). The black triangle represents the robot pose with its covariance ellipse. The map landmarks with their respective covariance ellipses are depicted in dark yellow. The red stars represent the current observation.

tions. Such observations are related to a partially known region. The uncertainty increase is due to the errors accumulated along the trajectory. In the RBPF, this results in growing differences among the sample weights, which leads to a slow decrease of  $N_{\text{eff}}$ . The slow decrease is due to the fact that the current observation is explained by part of the map that is similar among the different particles. In the KF, we see an increase of the covariance matrix of the robot pose,  $\Sigma$ . The newly created landmarks are correlated to the current pose, resulting in high correlation links with nearby landmarks. However, this correlation decays exponentially with landmarks far away in time and the covariance update can be approximated by modifying only portion of its matrix. This last observation is also at the core of some results about the sparsity of the information matrix in SLAM.

**Localization** Conversely, when the robot is moving in known terrain, it keeps localized in the map. Its uncertainty is bounded by the accuracy of the sensor and the map estimate by

	<b>Particles Behavior</b>	$N_{\text{eff}}$
<b>Exploration</b>	The particle cloud spreads according to the optimal proposal	slow decrease
<b>Localization</b>	Each robot pose hypothesis keeps localized in its own map.	mainly constant
<b>Loop Closing</b>	The particles are replicated according to their weight. The ones which correctly close a loop are retained, the others are suppressed.	fast drop

Table 5.1: The behavior of the RBPF during mapping

	<b>Data Association</b>	$\Sigma$
<b>Exploration</b>	One part of the observations is associated to highly correlated landmarks, the other one is used to create new landmarks.	slow increase
<b>Localization</b>	Observations are associated to highly correlated landmarks.	mainly constant
<b>Loop Closing</b>	Observations are associated to loosely correlated landmarks.	fast drop

Table 5.2: The behavior of the KF during mapping

the Cramer-Rao bound [Censi, 2007b]. When the sensor is accurate, the uncertainty of the map conditioned to the robot position is very small, thus this situation can be casted as a localization problem, which is simpler to deal. In the RBPF, this means that it is possible to compute the distribution of the robot poses only for one particle and then translate that distribution to the other particles. Since in this stage the different maps are locally similar, the contribution of the updates to the particle weight is more or less constant for the different samples. This strongly limits the  $N_{\text{eff}}$  decrease, resulting in an observed constant behavior. In the KF, we have that the landmarks estimate shows only minor update and is mainly constant, so that is possible to approximate the measurement update phase and only update the robot pose part. In this case, the robot uncertainty is mainly constant and bounded by the Cramer-Rao.

**Loop Closing** A loop closure results in a decrease of the uncertainty of the robot pose in particular and of the whole system in general, constraining the uncertainty at the loop closure to be smaller than the uncertainty at the loop entrance. In the RBPF, the likelihood of each particle is evaluated according to its own map. In this situation only a fraction of the robot pose hypotheses are in the right position. These hypotheses will be rewarded by a high weight, while the others are likely to be suppressed by a subsequent resampling stage. This



results in an entropy decrease, since the weights distribution becomes more peaked. After a loop has been detected only a few particles have a high weight and a fast drop of  $N_{\text{eff}}$  is observed. If using an adaptive resampling schema the low  $N_{\text{eff}}$  value triggers a resampling action. In the KF, we have that the current observation is associated with two main landmarks cliques (see Figure 5.3 center). These two cliques are composed by highly correlated landmarks, but they are almost uncorrelated between each other. This has two main effect in the filter behavior. The first is that data association became more difficult as each clique can push the system towards different part of the solution space, making also robust data association algorithm like the joint compatibility test harder. The second is that, after the loop is closed, these two groups of landmarks became correlated, resulting in a forced full update of the covariance matrix.

### 5.3 Map Representation

In order to exploit the different situations devised in the previous section, the map representation has to be carefully planned. Firstly, we need an accurate description, so that a pose tracker algorithm can provide precise results. This is particularly relevant when the mapper is in the *localization* situation. Secondly, we want to detect in which situation the mapper is, thus obtaining a form of situation assessment. Lastly, we need a representation when sub-part of the map can be easily accessed and composed together.

In order to achieve all of this requisites, we decide to represent the environment in a hybrid topological-metrical map. Our choice is to represent a map as a set of local maps (patches) embedded into a global network (graph). The nodes of the graph are the patch references, while the edges are the connections among patches. Given that the patches can be assumed locally consistent, it is possible to represent a global map by specifying the location of the patches in a global reference frame. We can therefore represent a sampled distribution over maps, by storing:

- all the patches  $\mathcal{P}_1, \dots, \mathcal{P}_N$ .
- for each trajectory sample  $x_{0:t}^{(i)}$ , the locations of the patches  $l_1^{(i)}, \dots, l_N^{(i)}$  according to that trajectory.

We can generate a standard grid representation for each particle by combining all the relevant patches. This representation is much more compact than storing individual grid maps. Let  $\phi_i$  be a list of references to the patches. The corresponding map can be computed by

$$m^{(i)} = \bigcup_n l_n^{(i)} \oplus \mathcal{P}_{\phi_i(n)}. \quad (5.1)$$

In general, there exists more than one possible topological structure. The environmental ambiguities in the loop closure map can result in a multi-modal particle distribution. This can be modelled by defining the concept of “particle cluster”. A particle cluster is a set of trajectory samples which can be obtained by grouping them in small regions so that the maps within one cluster share the same topology. The reader can think of a cluster as the set of trajectory samples that belong to a single mode in the distribution.

In other words a cluster is a subset of particles whose maps share the same topological graph. Of course, each particle belongs to a single cluster. After it is created, a cluster

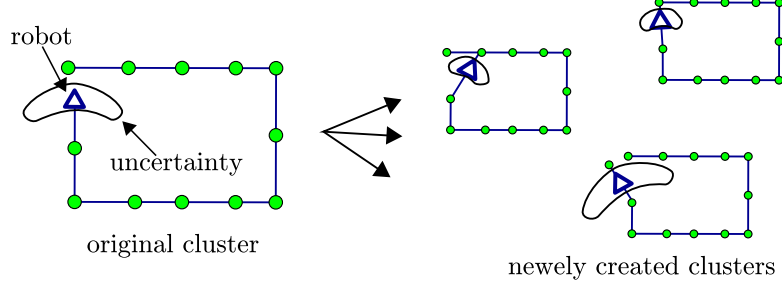


Figure 5.4: Particle cluster example: (left) The posterior about the robot poses. (right) The corresponding topologies and posterior after clustering.

is evolved independently from the others. The only interaction is in the resampling steps, that are simultaneously performed on all the particles in the system. Due to the fact that the clusters are evolved independently, within each cluster we can have different patches. Moreover, since the particles in a cluster share the same topology we don't need to store the topological structure in each particle, but only once, in the particle's cluster.

In [Figure 5.4](#) the idea of how to compute a map cluster from a set of particles is depicted.

A generic trajectory PDF can be factorized in a set of clusters, each of them having its own map patches. The concept of cluster does not explicitly appear in the mathematical formulation of the algorithm, but it is a support feature used for enforcing the local consistency of the maps by tracking different hypotheses.

Within a particle cluster, the local maps of each particle are locally similarity. Therefore, they can share their patches. This results in a much more compact representation compared to storing individual grid maps. In our current implementation, we used a graph structure and each node is a reference to the corresponding patch. To implement our representation, we store for each particle the state vector  $s_t^{(i)}$

$$s_t^{(i)} = \left\langle \underbrace{x_t^{(i)}}_{\text{robot pose}}, \underbrace{k}_{\text{cluster ID}}, \underbrace{l_1^{(i)}, \dots, l_{N_k}^{(i)}}_{\text{patch locations}} \right\rangle, \quad (5.2)$$

whereas each cluster  $\mathcal{C}_k$  is represented by

$$\mathcal{C}_k = \left\langle \underbrace{\mathcal{P}_1, \dots, \mathcal{P}_{N_k}}_{\text{pointer to patches}}, \underbrace{\{e_{l,m}\}}_{\text{graph edges}} \right\rangle. \quad (5.3)$$

Note that  $N_K$  does not grow with the length of trajectory traveled by the robot. It grows with the number of relevant patches which is related to the size of the environment.

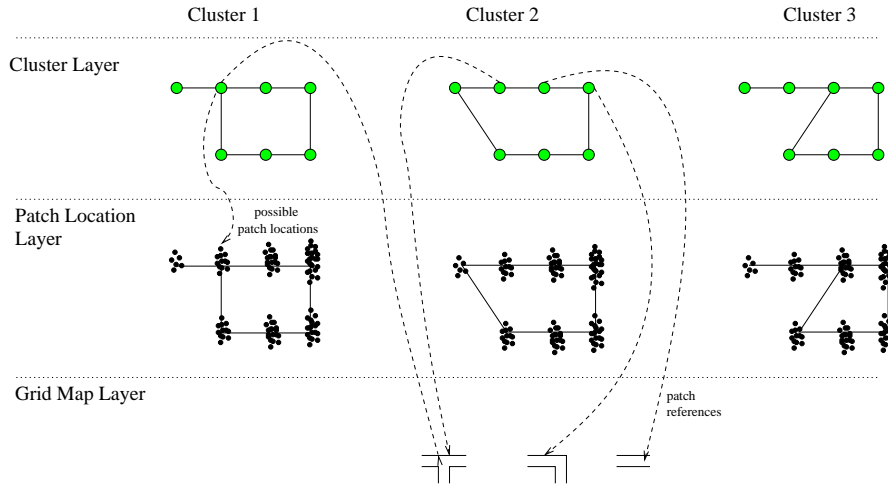


Figure 5.5: The map representation used in our approach.

In practice we extend the notion of a hierarchical representation made of two layers, to a three layered representation, shown in [Figure 5.5](#). The top layer (cluster layer), takes into account the topologies resulting by enforcing the local map similarity within a cluster. The middle layer represents the distribution of the centers of the patches, within one cluster. The final layer holds the grid patch representation. The compactness is achieved by the fact that the patches are shared among clusters and particles.

### 5.3.1 Basic Operations

In this section we describe some details about the basic operations performed in updating the previously described map representation. For each cluster  $\mathcal{C}_k$ , we need to store in which node  $v_k$  the robot actually is. We call this variable the *active vertex*, and it represents the vertex around which the local map is constructed. Before describing the algorithm we need to give some insight about the basic operations. These operations include: selecting the active vertex in the map, constructing a new patch and constructing a local grid map.

#### Changing the Active Vertex

When the robot moves, its location in the map changes, and the active vertex should change accordingly. When the robot is in vertex  $v_k$ , a local graph of the patch locations around  $v_k$  is constructed by selecting the neighborhood visit of the cluster's graph. This visit returns a set of vertexes  $\mathcal{V}_t = \{v_1, \dots, v_n\}$ . Each of these vertexes is associated to a map patch. Let  $i$  be a generic particle in the cluster, and consider a vertex  $v_q \in \mathcal{V}_t$ . Given the pair  $\langle i, v_q \rangle$  it is possible to instantiate the location  $l_q^{(i)}$  of the map patch corresponding to  $v_q$ . Generalizing, given a particle index  $i$  and a set of vertexes  $\mathcal{V}_t$  we can consider the corresponding set of patches locations  $\mathcal{L}_t = \{l_1^{(i)}, \dots, l_n^{(i)}\}$ . A straightforward approach for choosing the active vertex is to randomly select a particle  $i$  in the cluster, and choose as a new active vertex the

one which minimizes the distance among the particle's robot pose  $x_t^{(i)}$  and  $l_q^{(i)} \in \mathcal{L}_t$ . If the minimum distance is above a given threshold a new patch and a new vertex are added to the map of each particle in the cluster, and the newly created vertex becomes the active one.

### Creating a New Patch

Each time the active vertex changes, a local empty map is created. Such a map is updated according to the scan-matcher corrected robot poses, and the center of mass of the occupied cells is selected as the patch reference frame. All of the patch points are expressed in the reference frame of the patch center of mass. In this case the active vertex is moved to the newly created patch, and the new patch is added to the cluster. The newly created vertex is subsequently connected with the previous active vertex by adding an edge.

### Constructing a Local Map

There exists a local grid map for each cluster. It is rebuilt each time the active vertex changes. The construction of the local map works as follows: a random particle  $i$  within the cluster is selected. The graph around the active vertex is visited up to a given depth. This results in the selection of a set of vertexes. The location of the corresponding patches within the particle  $i$ ,  $l_{r1}^{(i)}, \dots, l_{rN}^{(i)}$  becomes the new patches locations. The patches are painted in an empty grid map one over the other, according to their locations.

### Patch Directory

As an additional optimization we notice that although the criterion for detecting possible loop closures discussed in [Section 5.5](#) is effective, it can be time consuming, because of the ray tracing operation on grid maps, which has to be performed for each particle. In order to limit the number of loop closure checks, the analysis is performed depending on the outcome of a heuristic. This heuristic relies on a low resolution grid map: the *patch directory*. Each cell of the patch directory contains a set of pairs of indices: a particle index, and a patch index. The pair  $\langle i, k \rangle$  is contained in the cell  $(x, y)$  if the patch  $k$  translated according to its location  $l_k^{(i)}$  relative to the particle  $i$ , covers the map cell  $(x, y)$ . At each time we can check whether a loop closure can occur, by computing a bounding box  $\mathcal{B}_t$ , which is the union of the bounding boxes  $b_t^{(i)}$  of the reading  $z_t$ , translated according to the particles. Subsequently, we consider all of the patch indices which fall within this bounding box  $k_1, \dots, k_M$ . By computing the set difference among this set and the patch indices in the local map, we have the indices of the map patches which are not in the local map, but are covered by a sensor reading for some particle. If this set is not empty, then the deep loop check described in [Section 5.5](#) is performed. In order to capture a locally consistent cluster we keep an independent *patch directory* for each cluster.

## 5.4 Situation Based Rao-Blackwellized Particle Filters

A key problem of RBPF is the computational complexity of the informed proposal distribution. The computations need to be carried out for each sample individually. Moreover, each

particle maintains a full grid map of the environment which requires to store large structures into memory. As a result, such a mapping system is able to run online only for small particle set, preventing its use in large environments.

In general the resulting maps around the robot pose are locally similar, since the samples are drawn from a peaked proposal. This means that, considering the same region of the environment, and two different particles at the same time instant, it is possible to find a translation which makes the maps of the two particles overlap.

It can be observed that, during the majority of the time, the mapping system is either in a localization situation, or in an exploration one. The loop closing situations occur not so often. When not closing a loop the shape of the proposal distribution depends only on:

- the robot pose within the local map;
- the local map around the robot;
- the current observation.

Therefore, we can compute the proposal distribution for a particle starting from the proposal computed for another particle, by simply translating the first proposal according to the position of the two particles local maps.

In the following we discuss how to efficiently update the distribution over robot trajectory. This can be done by exploiting the a priori knowledge about the mapping system. For efficiently estimating such a proposal the following assumptions have to hold:

**Assumption 5.1** *The current situation is known, which means that the robot is able to determine if its exploring new terrain, localizing in a known area or closing a loop.*

**Assumption 5.2** *The corresponding local maps of two samples are similar if considered in a particle-centered reference frame. In the following, we refer to this property as local similarity of the maps.*

**Assumption 5.3** *An accurate algorithm for pose tracking is used and the observations are affected by a limited sensor noise.*

A heuristic that enforces **Assumption 5.1** is presented in **Section 5.5**. **Assumption 5.3** is normally satisfied if using a laser range finder together with a scan matching procedure. For **Assumption 5.2** to hold the different modes of the robot trajectory distribution have to be independently updated, as explained in **Section 5.3**. Under the above assumptions we derive the equations for drawing the particles according to the goal distribution, and for updating the weights in the three situations of **Table 5.1**.

### 5.4.1 Exploration

For proximity sensors like laser range finders, the observations of the robot cover only a local area around the robot. As a result, we only need to consider the surroundings of the robot when computing the proposal distribution and the importance weight of the RBPF. Let  $\tilde{m}_{t-1}^{(i)}$

be the local map of particle  $i$  around its previous pose  $x_{t-1}^{(i)}$ . In the surroundings of the robot, we have

$$p(x_t | m_{t-1}^{(i)}, x_{t-1}^{(i)}, z_t, u_{t-1}) = p(x_t | \tilde{m}_{t-1}^{(i)}, x_{t-1}^{(i)}, z_t, u_{t-1}) \quad (5.4)$$

Let  $\oplus$  and  $\ominus$  be the standard pose compounding operators (see [Lu & Milios, 1997b]):  $a \ominus b$  is an operator that translates all the points in the domain of the function  $a$  so that the new origin of the domain of  $a$  is  $b$ <sup>1</sup>. The local similarity between maps (Assumption 5.2) allows us to write

$$\tilde{m}_{t-1}^{(i)} \ominus x_{t-1}^{(i)} \simeq \tilde{m}_{t-1}^{(j)} \ominus x_{t-1}^{(j)}. \quad (5.5)$$

### Proposal Distribution

Considering the insight obtained from Assumption 5.2, we obtain that the proposal distributions of different particles are similar if transformed to an egocentric reference frame

$$p(x_t \ominus x_{t-1}^{(j)} | \tilde{m}_{t-1}^{(j)}, x_{t-1}^{(j)}, z_t, u_{t-1}) \simeq p(x_t \ominus x_{t-1}^{(i)} | \tilde{m}_{t-1}^{(i)}, x_{t-1}^{(i)}, z_t, u_{t-1}) \quad (5.6)$$

Therefore, we can compute the proposal distribution of particle  $j$  by computing the proposal distribution in the reference frame of another particle, say  $i$ , and then translating it to the reference frame of particle  $j$

$$p(x_t | \tilde{m}_{t-1}^{(j)}, x_{t-1}^{(j)}, z_t, u_{t-1}) \simeq p(x_{t-1}^{(j)} \oplus (x_t \ominus x_{t-1}^{(i)}) | \tilde{m}_{t-1}^{(i)}, x_{t-1}^{(i)}, z_t, u_{t-1}) \quad (5.7)$$

This computation is illustrated in Figure 5.6. It shows how to transform a proposal distribution among particles. In this way, the complex proposal computation needs to be performed only once and its result is shared among all the particles.

### Importance Weighting

It can be observed that the two proposals are close to each other. Equation 5.7 tells us that, when exploring unknown environments, we can compute the informed proposal for only one particle. The proposal for other particles can be subsequently obtained by simple coordinate transformations. There exists empirical evidence that the difference in the particles weights slowly increases, resulting in a slow  $N_{\text{eff}}$  decay. This decrease strongly depends on the amount of overlapping among the current observation and the known part of the map. This overlap is particle dependent.

Evaluating the weights of a particle in this case requires to evaluate the weights according to

$$w_t^{(i)} = w_{t-1}^{(i)} p(z_t | m_{t-1}^{(i)}, x_{t-1}^{(i)}, u_{t-1})$$

<sup>1</sup> In the following of this document we make an extensive use of the motion composition operators  $\oplus$  and  $\ominus$ . If  $a$  and  $b$  are robot poses  $c = a \ominus b$  is the relative movement which translates the robot from  $b$  to  $a$ . Accordingly the  $\oplus$  operator applies a relative movement to a robot pose, so that the following holds  $b \oplus c = a$ .

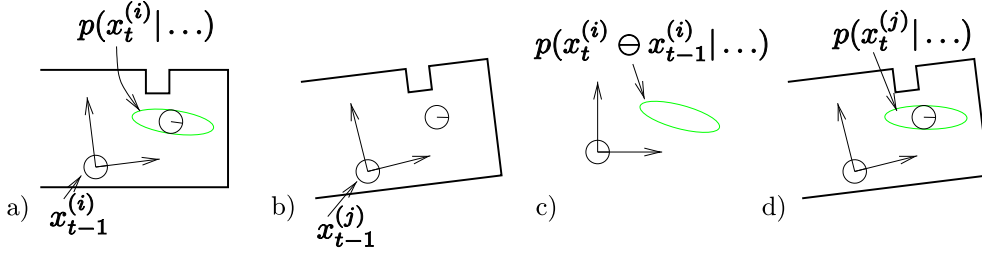


Figure 5.6: Proposal distribution in the exploration situation: Image (a) depicts the pose of the robot and its local map for one particle. The proposal computed is represented by the green/gray ellipse. Image (b) depicts the local map for another particle. The proposal computed in the robot reference frame is shown in (c), and (d) illustrates the proposal of the  $i$ -th particle translated into the reference frame of the  $j$ -th.

Performing the weights updates according to the above equation requires to sample the likelihood around each robot pose. Furthermore, we notice that if we drop the normalization step, unless a resampling is made, the weight of one particle does not depend from the others, therefore the decrease of the absolute value of  $N_{\text{eff}}$  does not depend on the number of samples. By raising the number of samples we can avoid the  $N_{\text{eff}}$  value to trigger a resampling. In other words we can choose a number of particles high enough so that the decrease of  $N_{\text{eff}}$  that occurs when navigating in an unknown region never triggers the resampling. This consideration lead us to compute the weight once for the reference particle, and then propagating it to all of the particles which are drawn according to the proposal computed for such a reference particle.

### Approximation Details

The level of approximation introduced by the above technique is strictly related to the size of the local maps,  $\tilde{m}^{(i)}$ . In the following we will show that if the local map consists only of the previous reading, the method is exact.

**Theorem 5.1 (Approximation Theorem)** *Let the local map consist only of the previous reading and [Assumption 5.3](#) hold. The similarity becomes equality and [Equation 5.6](#) becomes*

$$p(x_t \ominus x_{t-1}^{(j)} | \tilde{m}_{t-1}^{(j)}, x_{t-1}^{(j)}, z_t, u_{t-1}) = p(x_t \ominus x_{t-1}^{(i)} | \tilde{m}_{t-1}^{(i)}, x_{t-1}^{(i)}, z_t, u_{t-1}) \quad (5.8)$$

**Proof of the Approximation Theorem** Let rewrite [Equation 5.8](#) into

$$p(x_t \ominus x_{t-1}^{(j)} | z_{t-1}, x_{t-1}^{(j)}, z_t, u_{t-1}) = p(x_t \ominus x_{t-1}^{(i)} | z_{t-1}, x_{t-1}^{(i)}, z_t, u_{t-1}) \quad (5.9)$$

where the only *particle dependent* part is represented by  $x_{t-1}^{(j)}$ . Let now consider how the distribution is computed. The typical solution, and the one adopted in this work, is to perform a Gaussian approximation centered in the maximum reported by a scan matcher algorithm. In such a setting, the variable  $x_{t-1}^{(j)}$  only influences the mean of the resulting distribution. The scan matcher distribution is the same in both cases, allowing us to write

$$p(s_t | z_{t-1}, x_{t-1}^{(i)}, z_t, u_{t-1}) = p(s_t | z_{t-1}, x_{t-1}^{(j)}, z_t, u_{t-1}) \quad (5.10)$$

where  $s_t^{(i)} = x_t \ominus x_{t-1}$  is the displacement computed by the scan matcher. ■

**Corollary 5.1** *Let the local map consist only of the previous reading and [Assumption 5.3](#) hold. The proposal distribution of a particle can be exactly obtained by computing it in the reference frame of another particle and then translating it to the reference frame of the first one.*

$$p(x_t | \tilde{m}_{t-1}^{(j)}, x_{t-1}^{(j)}, z_t, u_{t-1}) = p(x_{t-1}^{(j)} \oplus (x_t \ominus x_{t-1}^{(i)}) | \tilde{m}_{t-1}^{(i)}, x_{t-1}^{(i)}, z_t, u_{t-1}) \quad (5.11)$$

**Proof** The fact comes directly from [Theorem 5.1](#) and the fact that  $s_t^{(i)} = x_t \ominus x_{t-1}^{(i)}$ . ■

In general, the smaller is the local map, the better is the approximation of this approach. However, for smaller local maps, the uncertainty of the distribution is slightly bigger, thus more particles are needed. In practice, the size of the maps is chosen according to the precision requested from the application and the available computational power.

## 5.4.2 Localization

Whenever the robot moves through known areas, each particle stays localized in its own map according to [Assumption 5.3](#). In such a situation, the pose distribution is very peaked and can be approximated with a Dirac distribution. To update the pose of each particle while the robot moves, we choose the pose  $x_t$  that maximizes the likelihood of the observation around the pose predicted by the odometry reading.

$$x_t^{(i)} = \operatorname{argmax}_{x_t} p(x_t | \tilde{m}_{t-1}^{(i)}, x_{t-1}^{(i)}, z_t, u_{t-1}) \quad (5.12)$$

Suppose that we know a pair of corresponding points in the two local maps  $l^{(i)}$  and  $l^{(j)}$ . Since we assume that the robot moves through known terrain, these reference frames have been established when the robot has visited the corresponding area for the first time. By exploiting [Assumption 5.2](#) and [Equation 5.5](#), we can rewrite the similarity among local maps as follows:

$$\tilde{m}_{t-1}^{(i)} \ominus l^{(i)} \simeq \tilde{m}_{t-1}^{(j)} \ominus l^{(j)}. \quad (5.13)$$



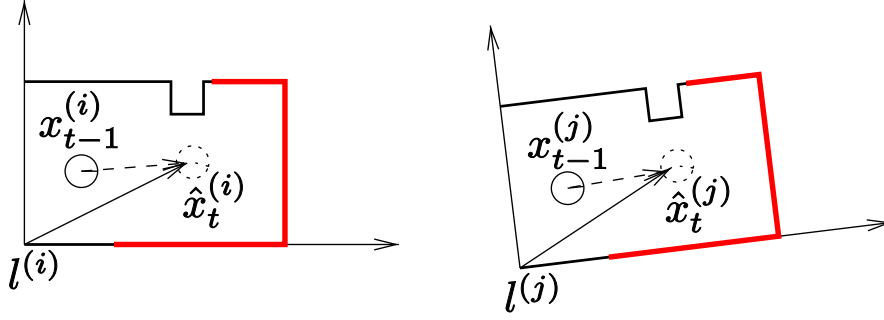


Figure 5.7: Proposal distribution in the localization situation. Left: The robot moves according to the dashed arrow. The maximum likelihood pose computed for a robot pose. In red are the observations. Right: We can obtain the maximum likelihood pose for other particles by applying the robot translation vector to a local map relative reference frame.

### Proposal Distribution

The above relationship allows to rewrite the right term of Equation 5.12, for the two particles as

$$p(x_t \ominus l^{(j)} \mid \tilde{m}_{t-1}^{(j)}, x_{t-1}^{(j)}, z_t, u_{t-1}) \simeq p(x_t \ominus l^{(i)} \mid \tilde{m}_{t-1}^{(i)}, x_{t-1}^{(i)}, z_t, u_{t-1}). \quad (5.14)$$

By manipulating Equation 5.6 we can express the prior of a robot pose hypothesis with respect to the current observation and its own map, as a function of the proposal computed for another particle:

$$p(x_t \mid \tilde{m}_{t-1}^{(j)}, x_{t-1}^{(j)}, z_t, u_{t-1}) \simeq p(l^{(j)} \oplus (x_t \ominus l^{(i)} \mid \tilde{m}_{t-1}^{(i)}, x_{t-1}^{(i)}, z_t, u_{t-1})) \quad (5.15)$$

By combining Equation 5.14 and Equation 5.15 we have

$$\begin{aligned} x_t^{(j)} &= \operatorname{argmax}_{x_t} p(x_t \mid \tilde{m}_{t-1}^{(j)}, x_{t-1}^{(j)}, z_t, u_{t-1}) \\ &\simeq \operatorname{argmax}_{x_t} (l^{(j)} \oplus (p(x_t \ominus l^{(i)}), \tilde{m}_{t-1}^{(i)}, x_{t-1}^{(i)}, z_t, u_{t-1})) \\ &= l^{(j)} \oplus (x_t^{(i)} \ominus l^{(i)}) \end{aligned} \quad (5.16)$$

The previous equation tells us that, if the maps are locally similar and we know two corresponding points in the local maps of different samples we can estimate Equation 5.12 for a generic sample. This can be done for one sample by evaluating Equation 5.12 for a particle  $i$ . Once a value  $x_t^{(i)}$  is obtained for the  $i^{\text{th}}$  sample, it is possible to compute the result of the evaluation of Equation 5.12 for another sample  $j$  by a simple combination of relative movements expressed by Equation 5.16.

In Figure 5.7 the above procedure is sketched.

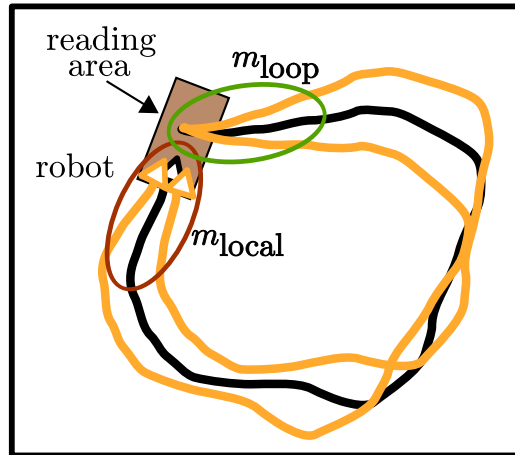


Figure 5.8: Local vs. loop map. The local map is used for the proposal distribution. The loop map for the weight computation.

### Importance Weighting

The weight update in this stage is done as in the *exploration* situation: the weight variation of the reference particle is propagated to all of the particles drawn from the reference particle proposal.

### 5.4.3 Loop Closing

In contrast to the two situations described before, the computation of the proposal is more complex in case of a loop-closure. This is due to the fact that [Assumption 5.2](#) (local similarity) is typically violated even for subsets of particles. Let us assume that the particle cloud is widely spread when the loop is closed. Typically, the individual samples reenter the previously mapped terrain at different locations. This results in different hypotheses about the topology of the environment and definitively violates [Assumption 5.2](#). Dealing with such a situation, requires additional effort in the estimation process.

#### Proposal Distribution

Whenever a particle  $i$  closes a loop, we consider that the map  $\tilde{m}_t^{(i)}$  of its surroundings consists of two components:  $m_{\text{loop}}^{(i)}$  referring to the map of the area the robot seeks to reenter;  $m_{\text{local}}^{(i)}$  referring to the map constructed from the most recent measurements, without the part of the map that overlaps with  $m_{\text{loop}}^{(i)}$ . In [Figure 5.8](#) is sketched the difference between the two maps.

$$p(z_t | x_t, m_{t-1}^{(i)}) = p(z_t | x_t, m_{\text{local}}^{(i)}, m_{\text{loop}}^{(i)}) \quad (5.17)$$

Since those two maps are disjoint and under the assumption that the individual grid cells are independent, we can use a factorized form for our likelihood function

$$p(z_t | x_t, m_{\text{local}}^{(i)}, m_{\text{loop}}^{(i)}) \propto p(z_t | x_t, m_{\text{local}}^{(i)})p(z_t | x_t, m_{\text{loop}}^{(i)}) \quad (5.18)$$

Given such decomposition, the samples are drawn by only considering the local part of the map, thus using the following proposal distribution

$$p(x_t | z_t, x_{t-1}^{(i)}, m_{\text{local}}^{(i)}, u_{t-1}) \quad (5.19)$$

Given that [Assumption 5.2](#) still holds when restricted to the local map  $m_{\text{local}}^{(i)}$ , the particles are drawn in the same way as they are in the *exploration* situation described in [Subsection 5.4.1](#)

### Importance Weighting

According to the Importance Sampling Principle, the choice of  $p(x_t | z_t, x_{t-1}^{(i)}, m_{\text{local}}^{(i)}, u_{t-1})$  as a proposal distribution leads to the following weight computation

$$\begin{aligned} w_t^{(i)} &= w_{t-1}^{(i)} \frac{p(x_t^{(i)} | z_t, x_{t-1}^{(i)}, m_{\text{local}}^{(i)}, m_{\text{loop}}^{(i)}, u_{t-1})}{p(x_t^{(i)} | z_t, x_{t-1}^{(i)}, m_{\text{local}}^{(i)}, u_{t-1})} \\ &= w_{t-1}^{(i)} \frac{\eta_1^{(i)} p(z_t | x_t^{(i)}, m_{\text{local}}^{(i)}) p(z_t | x_t^{(i)}, m_{\text{loop}}^{(i)}) p(x_t^{(i)} | x_{t-1}^{(i)}, u_{t-1})}{\eta_2^{(i)} p(z_t | x_t^{(i)}, m_{\text{local}}^{(i)}) p(x_t^{(i)} | x_{t-1}^{(i)}, u_{t-1})} \\ &= w_{t-1}^{(i)} p(z_t | x_t^{(i)}, m_{\text{loop}}^{(i)}) \frac{\eta_1^{(i)}}{\eta_2^{(i)}} \end{aligned} \quad (5.20)$$

The sample weights can be evaluated by computing the normalizing factors  $\eta_1$  and  $\eta_2$ , and the likelihood of the reading given the loop map. Using the weight update [Equation 5.20](#) is computationally expensive, due to the need of computing  $\eta_1/\eta_2$ . For this reason, in our approximated approach, we just drop the term  $\eta_1/\eta_2$ , having

$$w_t^{(i)} = w_{t-1}^{(i)} p(z_t | x_t^{(i)}, m_{\text{loop}}^{(i)}) \quad (5.21)$$

We observed in practical experiments that the normalizing constants in [Equation 5.20](#) have only minor influence in the weight computation. In [Figure 5.9](#) we compared the variations of the particle weights  $w_t^{(i)}/w_{t-1}^{(i)}$  during a loop closure, using the partial weight evaluation and full one of [Equation 5.20](#).

Additionally we measured the KLD distance of the weight distribution computed using the two weight update equations and the approximated one. The KLD distance is a measure of similarity among probability distribution: its value is 0 when the two distributions are identical, while it is infinity when the two distributions are totally different. A value of 0.02 means that dropping the term  $\eta_1/\eta_2$  has not a big effect.

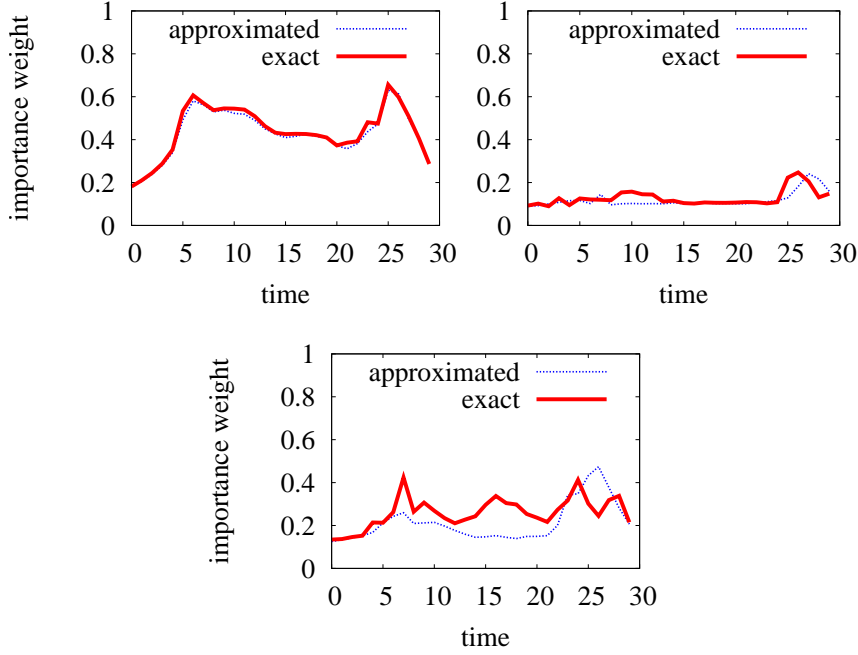


Figure 5.9: Weight approximation. The graphs show the evolution of the weight variation  $w_t^{(i)}/w_{t-1}^{(i)}$  of three different particles during a loop closure. The blue line shows the weight variation computed using Equation 5.20, while the red line shows the weight variation computed using Equation 5.21. The observation along the trajectories of the particles under the two different weighting strategies and the sequence of the observations are the same. The top images show typical results, the bottom one depicts one of the worst result during our experiments.

**Likelihood Evaluation in a Single Loop Closure Point** Let assume that  $m_{\text{loop}}$  is the same for all of the particles, up to a translation. Let  $t_E$  be the time index in which the robot first acquired the part of the environment described by the patch, and  $t_C$  the time in which it re-enters. Our representation ensures the following equation to hold:

$$m_E^{(i)} \ominus x_{t_E}^{(i)} \simeq m_E^{(j)} \ominus x_{t_E}^{(j)}. \quad (5.22)$$

Here  $i$  and  $j$  are two particle indices, while  $x_{t_E}^{(i)}$  and  $x_{t_E}^{(j)}$  represent the corresponding robot pose hypotheses at the loop entrance. Let the robot trajectories of the two particles along the loop be  $x_{t_E:t_C}^{(i)}$  and  $x_{t_E:t_C}^{(j)}$ . Since we know that, with respect to the closing patch,  $x_{t_E}^{(i)}$  and  $x_{t_E}^{(j)}$  are the same point, we can evaluate the likelihood of the particle  $j$  by attaching its trajectory  $x_{t_E:t_C}^{(j)}$  to  $x_{t_E}^{(i)}$ , and computing the likelihood with respect to the map computed for the particle  $i$ . This is done in the following way:

1. compute the overall displacement for the particle for which the likelihood have to be

evaluated  $\delta^{(j)} = x_{t_C}^{(j)} \ominus x_{t_E}^{(j)}$ .

2. evaluate the pose of the particle  $j$  according to the particle  $i$   $x_{t_C}^{(j,i)} = x_{t_E}^{(j)} \oplus \delta^{(j)}$
3. evaluate the approximated weight as  $p(z_t | x_{t_C}^{(j,i)}, m_E^{(i)})$

### Likelihood Evaluation under Small Uncertainties in the Loop Closure Point

The procedure described above is correct *only if we know the loop closing patch*. In the general case the uncertainty at the loop entrance can be big, and there can be different closure points according to the robot pose distribution during closure. We can still restrict the evaluation to the known loop closure case, but we need to split the evaluation of the particle's likelihoods in several known-closure evaluations. In our representation the only information needed for executing the previously described procedure is the index of the patch in which the closure occurs.

However,  $m_{\text{loop}}$  can spread over different patches. There are situations in which the closure map of different particles is constituted by the same indices, since different robot pose hypotheses are approaching to the closure from different directions. In order to restrict ourselves to the known loop closure point case we have to group the particles into sets so that all of the particles belonging to the same set can be considered to enter in the loop from nearby locations. The idea is, for each particle  $i$  and each patch in  $m_{\text{loop}}^{(i)}$  to consider the distances among the locations of the patches and the robot pose  $x_{t_C}^{(i)}$ . We compute a fingerprint for the closure map of a particle by sorting the indices of the local map patches according to the measured distances. Subsequently we can partition the particles in groups having the same fingerprint. Finally we compute a loop map for a randomly selected particle within each cluster. The likelihoods of the other particles are evaluated within each group using the previously described procedure. The process of building a fingerprint is sketched in [Figure 5.10](#).

### Loop Assessment

When a loop is found, the difference among the weights of the particles starts to increase. Accordingly  $N_{\text{eff}}$  drops. When the value of  $N_{\text{eff}}$  is below a given threshold we perform a resampling action. The resampling kills most of the wrongly aligned particles and replicates the good ones. At this point the loop is assessed, and some of the groups can have no particles in it. At this point all of the non-empty groups are translated into particle clusters. The graph of each newly created cluster is modified in order to reflect the topological modification in the map. This is done by adding an edge between the current vertex and the vertex associated to the loop entrance in the maps. Subsequently the new clusters are added to the representation. Observe that, in case of ambiguities in the loop closure, all the different topological hypotheses are tracked.

## 5.5 Situation Assessment

All of the derivations made in the previous section require that the robot knows if it is in an exploration state, it is localizing or it is currently closing a loop ([Assumption 5.1](#)). Here, we

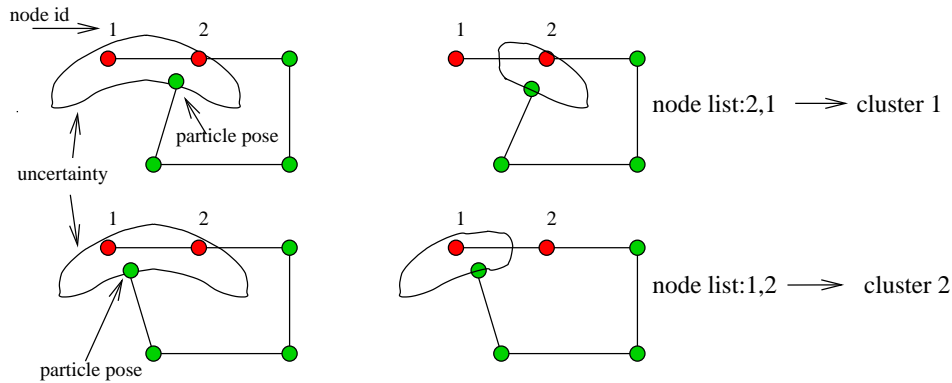


Figure 5.10: Fingerprint computation. a) Two trajectory/map instances before closing a loop. b) The computation of the two fingerprints and the corresponding grouping. The clustering of the topologies is done through the fingerprints.

describe how to distinguish the different cases. Detecting the first two situations can be done in a straightforward way by comparing the area covered by the current observation and the map constructed so far.

Slightly more complicated is to decide whether or not the robot is closing a loop. To make this decision, we apply the approach proposed by Stachniss *et al.* [Stachniss, Hähnel, & Burgard, 2004] in the context of exploration with active loop-closure. To determine whether there exists a possibility to close a loop we consider two different representations of the environment. Each particle  $s$  maintains an occupancy grid map  $m^{[s]}$  and a topological map  $\mathcal{G}^{[s]}$  during the mapping task. The vertices in  $\mathcal{G}^{[s]}$  represent positions visited by the robot and the trajectory of particle  $s$  corresponds to the edges in  $\mathcal{G}^{[s]}$ . New nodes are created and added to the graph structure whenever the robot moved for a certain distance or it cannot observe any previously created node.

Figure 5.11 shows such a graph for one particular particle during different phases of the mapping process. In each image the topological map  $\mathcal{G}^{[s]}$  is depicted on top of metric map  $m^{[s]}$ . To motivate the idea of our approach, consider the left image of Figure 5.11. Here the robot is almost closing a loop. This can be detected by the fact that the length of the shortest path between the current pose of the robot and previously visited locations in the topological map  $\mathcal{G}^{[s]}$  is large, whereas it is small in the grid-map  $m^{[s]}$ . Thus, to determine whether or not a loop closure can take place we compute for each sample  $s$  the set  $\mathcal{I}(s)$ . This set of positions of interest contains all nodes that are close to the current pose  $x_t^{[s]}$  of particle  $s$  based on the grid map  $m^{[s]}$  but are far away given the topological map  $\mathcal{G}^{[s]}$ :

$$\mathcal{I}(s) = \{x_{t'}^{[s]} \in \text{nodes}(\mathcal{G}^{[s]}) \mid \text{dist}_{m^{[s]}}(x_{t'}^{[s]}, x_t^{[s]}) < c_1 \wedge \text{dist}_{\mathcal{G}^{[s]}}(x_{t'}^{[s]}, x_t^{[s]}) > c_2\} \quad (5.23)$$

Here  $\text{dist}_m(x_1, x_2)$  is the length of the shortest path from  $x_1$  to  $x_2$  given the grid map and  $\text{dist}_{\mathcal{G}}(x_1, x_2)$  the shortest path based on the topological map. The terms  $c_1$  and  $c_2$  are constants that must satisfy the constraint  $c_1 < c_2$ .

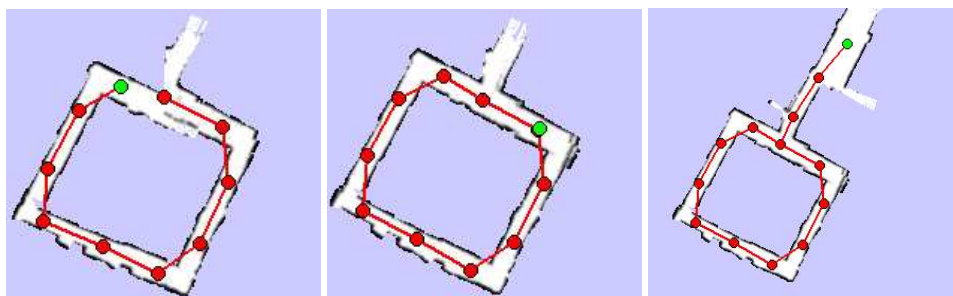


Figure 5.11: The red/gray circles and lines in these three images represent the nodes and edges of  $\mathcal{G}^{[s]}$ . In the left image,  $\mathcal{I}(s)$  contained two nodes and in the middle image the robot closes the loop to reduce its uncertainty. After this it continues to explore new terrain (right image).

If  $\mathcal{I}(s) \neq \emptyset$ , there exist metrical shortcuts from  $x_t^{[s]}$  to the positions in  $\mathcal{I}(s)$ . These shortcuts represent edges that would generate a cycle in the graph structure of  $\mathcal{G}^{[s]}$  and therefore constitute an opportunity to close a loop (compare Figure 5.11). To determine when to switch back from the *loop-closing* state to the *localization* state, we analyze the evolution of the likelihood computed in the closing map for a given time interval. Once the likelihood of the best particle is higher than a threshold for a long time, we assess the loop closure and switch back to a *localized* state.

## 5.6 Overall Algorithm

In this section we present the overall algorithm. For clarity of presentation we omit the memory management operations, like patch sharing, however in our implementation both the patches, and the cluster's graphs are represented as shared structures.

In Algorithm 4 we present the top level routine which is used for updating the filter state. The purpose of the algorithm is to give an high level intuition on how the things work, rather than a procedure literally describing all of the steps.

In addition to the previously described structures, for each cluster  $\mathcal{C}_k$  we update the following variables:

- $d_k$ : the patch directory, which is a coarse resolution grid map, in which each cell contains the set of patch indexes that cover the cell.
- $j_k$ : the patch index around which the local map is constructed.
- $r_k$ : the index of the best particle in the cluster.
- $tm_k$ : the temporary grid map used for constructing the new patches when augmenting the map.
- $m_k$ : the cluster's local grid map;

**Algorithm 4** is executed each time the robot has traveled for a minimum distance, and it has acquired a new laser scan. For each cluster, a scan matcher guess of the reference particle is computed, by considering the cluster's local map. Using such a guess the state transition (localization, augmentation or loop closing) of the mapping system is determined, and the corresponding update is performed. The resampling is performed whenever the overall  $N_{\text{eff}}$  goes below a given threshold. As a result of the resampling, the clusters which are not referenced by any particle are suppressed.

Due to the complexity of the algorithm we show the fragments for handling the different situations: *localizationUpdate(...)*, *augmentUpdate(...)* and *loopUpdate(...)*, in **Algorithm 5**, **Algorithm 6** and **Algorithm 7**. The first two fragments explicitly follow the ideas in **Section 5.4**. Conversely, the *loopUpdate(...)* fragment performs an heterogeneous sequence of operations, which have been explained in several sections of this chapter.

In the following we organize those operations, and discuss the loop closure handling described in **Algorithm 7**. Observe that the following operations are executed for each cluster in which a closure is detected. First, the fingerprints of the closure are computed. The fingerprints partition the indexes of the particles pointing to the original cluster's particle indexes ( $\mathcal{I}_k$ ) in sets sharing the same topological structure:  $\{f_1, \dots, f_q\}$ . Each of those sets is then treated independently, by performing the following steps:

- A loop map  $m_{\text{closure}}$  is computed, according to a randomly selected particle belonging to the partition.
- The weights of the particles within a partition are then evaluated according the partition's closure map  $m_{\text{closure}}$ .
- New edges are computed, in order to reflect in the topology the loop closure.
- A new cluster  $\mathcal{C}^{\text{new}}$  is created from the originating cluster  $\mathcal{C}_k$ , and the computed edges.
- The new particles are computed by the old ones, and are setted to belong to  $\mathcal{C}^{\text{new}}$ .

Finally, the original cluster is replaced by the set of clusters created by the loop closure, and the structures are updated accordingly.

### Computational Complexity

The asymptotic computational complexity can be derived by considering the time of each operation in the overall algorithm. Analyzing the latter, we find that:

$$\begin{aligned}
 t_{\text{Overall}} \sim & k(t_{\text{ScanMatcher}} + t_{\text{ComputeTransition}} + t_{\text{FixedDepthVisit}} + \\
 & + \max\{t_{\text{Localization}}, t_{\text{Augmenting}}, t_{\text{LoopClosing}}\}) + \\
 & + t_{N_{\text{eff}}} + t_{\text{Resampling}}
 \end{aligned} \tag{5.24}$$

where  $k$  stands for the maximum number of clusters and the subscripts to the diverse procedures. The time for scan matching, compute transition, and visiting the graph are bounded, as they refer to the local property of the environment and thus they do not depend on its structure (such as length or number of cycles). We now analyze the time of the different situations. In localization we have the following complexity

$$t_{\text{Localization}} \sim |I_k| + |\text{LocalMap}| + |\text{Scan}| \tag{5.25}$$



**Algorithm 4:** Approximated Particle Filter**Input:**

*ClusterSet* :  $\mathcal{C}_{t-1} = \{\mathcal{C}_1, \dots, \mathcal{C}_q\}$  the previous step clusters

*DirectorySet* :  $\mathcal{D}_{t-1} = \{d_1, \dots, d_q\}$  the previous step patch directory

*PatchIndexSet* :  $\mathcal{J}_{t-1} = \{j_1, \dots, j_q\}$  the local patch index of each cluster

*ParticleIndexSet* :  $\mathcal{R}_{t-1} = \{r_1, \dots, r_q\}$  the reference particle index in each cluster

*MapSet* :  $\mathcal{TM}_{t-1} = \{tm_1, \dots, tm_q\}$ , the temporary maps for the clusters

*MapSet* :  $\mathcal{M}_{t-1} = \{m_1, \dots, m_q\}$ , the local maps for the clusters

*SampleSet* :  $\mathcal{S}_{t-1}$ , the previous step particles

*Observation* :  $z_t$ , the current laser scan

*Motion* :  $u_t$ , the current odometry measure

**Output:**

*ClusterSet* :  $\mathcal{C}_t$

*DirectorySet* :  $\mathcal{D}_t$

*PatchIndexSet* :  $\mathcal{J}_t$

*ParticleIndexSet* :  $\mathcal{R}_t$

*MapSet* :  $\mathcal{TM}_t$

*MapSet* :  $\mathcal{M}_t$

*SampleSet* :  $\mathcal{S}_t$

```

1 forall  $\mathcal{C}_k \in \mathcal{C}$  do
2   ParticleIndexSet :  $\mathcal{I}_k = \{i | s_{t-1}^{(i)} = \langle x_{t-1}^{(i)}, k, l_1^{(i)}, \dots, l_{N_k}^{(i)} \rangle\}$ 
3   Pose :  $\hat{x}_k = \text{scanmatch}(x_{t-1}^{(r_k)}, m^{(r_k)}, z_t, u_t)$ 
4   Transition :  $\tau = \text{computeTransition}(x_k, \mathcal{C}_k, d_k, j_k)$ 
5   PatchIndexSet : localIndexes = fixedDepthVisit( $\mathcal{C}_k, j_k$ )
6   if  $\tau == \text{localizing}$  then
7     | localizationUpdate()
8   else if  $\tau == \text{exploration}$  then
9     | explorationUpdate()
10  else if  $\tau == \text{loopClosing}$  then
11    | loopClosing()
12  end
13 end
14 double :  $N_{\text{eff}} = \text{computeNeff}(\mathcal{S}_t)$ 
15 if  $N_{\text{eff}} < T$  then  $\mathcal{S}_t = \text{resample}(\mathcal{S}_t)$ 

```

**Algorithm 5:** localizationUpdate()

---

```

1 double :  $\delta_w = p(z_t, \hat{x}_k, m^{(r_k)})$ 
2 Pose :  $\delta_k = \hat{x}_k \ominus l_{j_k}^{(r_k)}$ 
3 forall  $i \in \mathcal{I}_k$  do
4   Pose :  $x_t^{(i)} = l_{r_k}^{(i)} \oplus \delta_k$ 
5   Sample :  $s_t^{(i)} = \langle x_t^{(i)}, k, l_1^{(i)}, \dots, l_{N_k}^{(i)} \rangle$ 
6   double :  $w_t^{(i)} = w_{t-1}^{(i)} \delta_w$ 
7    $\mathcal{S}_t = \mathcal{S}_t \cup \{s_t^{(i)}\}$ 
8 end
9 PatchIndex :  $j_k^{\text{new}} = \text{closestPatch}(\hat{x}_t^{(i)}, z_t, \text{patchIndexes}, \mathcal{C}_k)$ 
10 if  $j_k \neq j_k^{\text{new}}$  then
11   // Changing the Active Vertex
12   Map :  $m_k^{\text{new}} = \text{computeLocalMap}(\mathcal{C}_k, r_k, j_k^{\text{new}})$  // recompute local map
13   Map :  $tm_k^{\text{new}} = \text{emptyMap}$ 
14    $tm_k^{\text{new}} = \text{registerScan}(\hat{x}_k, z_t, tm_k)$ 
15    $\mathcal{T}\mathcal{M}_t = \mathcal{T}\mathcal{M}_t \cup \{tm_k^{\text{new}}\}$ 
16    $\mathcal{M}_t = \mathcal{M}_t \cup \{m_k^{\text{new}}\}$ 
17 else
18    $tm_k^{\text{new}} = \text{registerScan}(\hat{x}_k, z_t, tm_k)$ 
19    $\mathcal{T}\mathcal{M}_t = \mathcal{T}\mathcal{M}_t \cup \{tm_k^{\text{new}}\}$ 
20    $\mathcal{M}_t = \mathcal{M}_t \cup \{m_k^{\text{new}}\}$ 
21 end
22  $\mathcal{J}_t = \mathcal{J}_t \cup \{j_k^{\text{new}}\}$ 
23  $\mathcal{D}_t = \mathcal{D}_t \cup \{d_k\}$ 
24  $\mathcal{C}_t = \mathcal{C}_t \cup \{\mathcal{C}_k\}$ 

```

---

**Algorithm 6:** explorationUpdate()

---

```

// Adding New Patch
1 Map :  $tm_k^{\text{new}} = \text{registerScan}(\hat{x}_k, z_t, tm_k)$ 
2 Patch :  $\mathcal{P}^{\text{new}} = \text{tempToMap}(tm_k^{\text{new}})$ 
3  $tm_k^{\text{new}} = \text{emptyMap}$ 
4 Cluster :  $\mathcal{C}_k^{\text{new}} = \text{insert}(\mathcal{C}_k, \mathcal{P}^{\text{new}})$ 
5 PatchIndex :  $\hat{j}_k^{\text{new}} = \text{indexOfPatch}(\mathcal{C}_k^{\text{new}}, \mathcal{P}^{\text{new}})$ 
6 Map :  $m_k^{\text{new}} = \text{computeLocalMap}(\mathcal{C}_k^{\text{new}}, r_k, \hat{j}_k^{\text{new}})$ 
7 Pose :  $\delta_l = \text{patchCenter}(\mathcal{P}^{\text{new}}) \ominus \hat{x}_k$ 
8  $\mathcal{TM}_t = \mathcal{TM}_t \cup \{tm_k^{\text{new}}\}$ 
9  $\mathcal{M}_t = \mathcal{M}_t \cup \{m_k^{\text{new}}\}$ 
10  $\mathcal{J}_t = \mathcal{J}_t \cup \{j_k^{\text{new}}\}$ 
11  $\mathcal{D}_t = \mathcal{D}_t \cup \{\text{updateDirectory}(d_k, \hat{x}_k, z_t)\}$ 
12  $\mathcal{C}_t = \mathcal{C}_t \cup \{\mathcal{C}_k^{\text{new}}\}$ 

// Sampling new poses
13 Covariance :  $\Sigma$ 
14 Pose :  $\mu$ 
15 double :  $\delta_w = p(z_t, \hat{x}_k, m^{(r_k)})$ 
16  $\langle \Sigma, \mu \rangle = \text{computeProposal}(x_{t-1}^{(r_k)}, m^{(r_k)}, z_t, u_t)$ 
17 forall  $i \in \mathcal{I}_k$  do
18   Pose :  $x_t^{(i)} = x_{t-1}^{(i)} \oplus (\text{samplePose}(\Sigma, \mu) \ominus x_{t-1}^{(r_k)})$ 
19   PatchLocation :  $l^{(i)\text{new}} = x_t^{(i)} \oplus \delta_l$ 
20   Sample :  $s_t^{(i)} = \langle x_t^{(i)}, k, l_1^{(i)}, \dots, l_{N_k}^{(i)}, l^{(i)\text{new}} \rangle$ 
21   double :  $w_t^{(i)} = w_{t-1}^{(i)} \delta_w$ 
22    $\mathcal{S}_t = \mathcal{S}_t \cup \{s_t^{(i)}\}$ 
23 end

```

---

**Algorithm 7:** loopUpdate()

---

```

// Update the weights according to  $m_{\text{local}}$ 
1 double :  $\delta_w = p(z_t, \hat{x}_k, m^{(r_k)})$ 
2 SampleSet :  $\mathcal{S}'_t = \{\}$ 
3 forall  $i \in \mathcal{I}_k$  do
4   Pose :  $x_t^{(i)} = l_{r_k}^{(i)} \oplus \delta_k$ 
5   Sample :  $s_t^{(i)} = \langle x_t^{(i)}, k, l_1^{(i)}, \dots, l_{N_k}^{(i)} \rangle$ 
6   double :  $w_t^{(i)} = w_{t-1}^{(i)} \delta_w$ 
7    $\mathcal{S}'_t = \mathcal{S}'_t \cup \{s_t^{(i)}\}$ 
8 end
// New clusters creation
9 PatchIndexSet : localIndexes = fixedDepthVisit ( $\mathcal{C}_k, j_k$ )
10 SetPartition :  $\mathcal{F} = \{f_1, \dots, f_q\} = \text{fingerprintPartition}(d_k, \mathcal{S}'_t, z_t)$ 
11 forall  $f_i \in \mathcal{F}$  do
12   PatchIndexSet :  $\mathcal{K} = \text{computeClosure}(\mathcal{S}'_t, z_t, d_k) - \text{localIndexes}$ 
13   Cluster :  $\mathcal{C}^{\text{new}} = \text{connectEdges}(\mathcal{C}_k, \mathcal{K})$ 
14   ClusterIndex :  $u = \text{computeClusterIndex}(\mathcal{C}^{\text{new}})$ 
15   SampleIndex :  $h = \text{drawFromIndexSet}(f_i)$ 
16   PatchIndex :  $c = \text{computeClosestPatch}(x_t^{(h)}, \mathcal{K})$ 
17    $\mathcal{J}_t = \mathcal{J}_t \cup \{c\}$ 
18   Map :  $m^{\text{closure}} = \text{computeMap}(\mathcal{C}_k, \mathcal{K}, h)$ 
19   forall  $j \in f_i$  do
20     Pose :  $x_t^{(j,h)} = l_c^{(h)} \oplus (x_t^{(j)} \ominus l_c^{(j)})$ 
21     double :  $w_t^{(i)} = w_t^{(i)} p(z_t | m^{\text{closure}}, x_t^{(j,h)})$ 
22     Sample :  $s_t^{(j)\text{new}} = \langle x_t^{(i)}, u, l_1^{(i)}, \dots, l_{N_k}^{(i)} \rangle$ 
23      $\mathcal{S}_t = \mathcal{S}_t \cup \{s_t^{(j)\text{new}}\}$ 
24   end
25   SampleIndex :  $r_u = \text{bestSample}(f_i, \mathcal{S}'_t)$ 
26    $\mathcal{R}_t = \mathcal{R}_t \cup \{r_u\}$ 
27    $tm_u^{\text{new}} = \text{emptyMap}$ 
28    $\mathcal{TM}_t = \mathcal{TM}_t \cup \{tm_u^{\text{new}}\}$ 
29    $m_u^{\text{new}} = \text{computeLocalMap}(\mathcal{C}^{\text{new}}, r_u, c)$ 
30    $\mathcal{M}_t = \mathcal{M}_t \cup \{m_u^{\text{new}}\}$ 
31    $\mathcal{C}_t = \mathcal{C}_t \cup \{\mathcal{C}^{\text{new}}\}$ 
32 end

```

---

where  $|I_k|$  stands for the number of particles belonging to the  $k$ -th cluster,  $|LocalMap|$  is the dimension of the local map and  $|Scan|$  is the number of readings. Both, the local map dimension and the number of readings, are fixed and do not depend on the specific environment to be mapped, so the localization complexity is  $O(|I_k|)$ . In augmenting we have the following complexity

$$t_{Augmenting} \sim |I_k| + |LocalMap| + |Scan| + t_{ComputeProposal} \quad (5.26)$$

where  $|I_k|$  stands for the number of particles belonging to the  $k$ -th cluster,  $|LocalMap|$  is the dimension of the local map and  $|Scan|$  is the number of readings. Both, the local map dimension and the number of readings, are fixed and do not depend on the specific environment to be mapped. The time for computing the proposal is also bounded and do not depend on the environment, but nevertheless is very time consuming (This will be reconsider in the followings), so the augmenting complexity is  $O(|I_k|)$ . In the loop closing we have

$$t_{LoopClosing} \sim |I_k| + |LocalMap| + |Scan| + t_{FixedDepthVisit} + |F_k|(|ClosureMap| + t_{RebuildDirectory}) \quad (5.27)$$

where  $|I_k|$  stands for the number of particles belonging to the  $k$ -th cluster,  $|LocalMap|$  is the dimension of the local map,  $|Scan|$  is the number of readings,  $|F_k|$  is the number of generated fingerprints and  $|ClosureMap|$  is the dimension of the closure map. The local and closure maps dimension, the number of readings and the time for visiting the graph are fixed and do not depend on the specific environment to be mapped. The time for rebuilding the directory depends on the number of patches presented and so on the length of the path.  $|F_k|$  depends on the structure, but is bounded by  $|I_k|$ , thus resulting in a complexity of  $O(|I_k| \cdot |Patches|)$ , where  $|Patches|$  represents the current number of patches. The time for computing neff  $t_{Neff}$  is  $O(n)$  where  $n$  is the number of particles, and the time for resampling  $t_{Resampling}$  is  $O(n \cdot |Patches|)$ . Considering that  $k \cdot |I_k| = n$  and that resampling occurs only in loop closing events, the overall complexity is  $O(n)$  during the normal behavior and  $O(n \cdot |Patches|)$  in loop closing. Moreover, in typical settings, all the cost are small but the time for computing the proposal, the number of fingerprint generated are small compared to the number of particles we have a complexity of  $O(k)$  in normal behavior and of  $O(k \cdot |Patches|)$  in loop closing.

## 5.7 Experiments

The proposed approach has been evaluated with several experiments performed on datasets acquired using real robots and in simulation. In particular our approximated approach was able to estimate maps that were topologically correct and showed a good metric quality.

The test datasets are available to the SLAM community[Howard & Roy, 2003], and are becoming an important testbed for validating the performances of a SLAM algorithm. With the proposed technique it has been possible to build accurate maps in real time of environments whose size was  $250 \times 250$  meters. The mobile bases used for acquiring the data are ActivMedia Pioneer 2 AT, Pioneer 2 DX-8, and iRobot B21r, equipped with SICK PLS or LMS range finders.



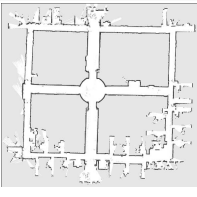
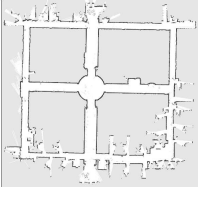
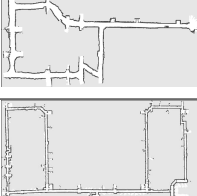
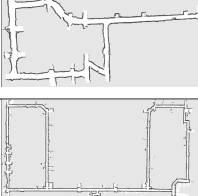
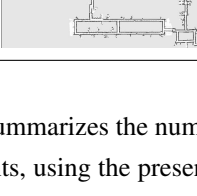
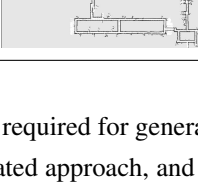
Environment	Approximated approach		Rao-Blackwellized particle filter	
	Result	particles	Result	particles
Intel Lab		30		15
ACES building		30		30
Bruceton mines		50		30
MIT Killian Court		1000		120

Table 5.3: This table summarizes the number of particle required for generating a good map, for several environments, using the presented approximated approach, and a straightforward RBPf implementation in [Grisetti, Stachniss, & Burgard, 2005]

### 5.7.1 Number of Particles

The proposed approach introduces several approximations to the implementation of a RBPf algorithm for grid maps. In order to capture the errors introduced by the used approximations one can conjecture that a higher number of particles has to be used. In order to quantitatively measure the validity of the introduced approach, we compared it with the RBPf implementation of Grisetti *et.al* [Grisetti, Stachniss, & Burgard, 2005]. Using the two approaches we compared the number of particles required for achieving a correct map. The results are summarized in Table 5.3. It turns out that the increase in the number of particle required by the approximated algorithm is less than one order of magnitude.

However, the performances decreases in environments in which the static assumptions are violated, or in situations in which the local assumption does not hold. In particular this is true in extremely cluttered and noisy environments, for instance when the robot perceives the grass in a field. In this case, minimal differences of the particles positions within the local map can lead to completely different proposal distribution. Errors can arise if a single proposal is computed and propagated to all of the particles. This is the reason of the big gap among the

Table 5.4: Comparison of memory and computational resources based on the MIT dataset using a PC with a 1.3 GHz CPU.

	#particles	runtime	max. memory
our approach	2.000	56 min	210 MB
our approach	1.500	51 min	200 MB
our approach	1.000	41 min	180 MB
our approach	500	35 min	165 MB
RBPF	150	(memory swapping)	2.9 GB
RBPF	80	300 min	1.5 GB
RBPF	50	190 min	1 GB

particles required by the two approaches for computing the MIT map. The windows and the mirrors in that environment makes it challenging for an RBPF mapper which uses a shared proposal distribution having a single mode. However, this problem will be solved in future versions, in which we plan to compute a multi-modal proposal in the Mapping Situation.

## 5.7.2 Mapping Experiments

The approach was validated on several logs available on [Howard & Roy, 2003]. The proposed technique allows to generate high quality maps of large scale environments, in one order of magnitude less time than previous state-of-the-art approaches. This is obtained by the combination of an improved proposal and approximated map representations. In Figure 5.12 and Figure 5.13 some of the generated maps are shown.

## 5.7.3 Performances

The second experiment is designed to show the advantages of our approach when compared to RBPF mapper without our optimizations. To compare the results, we used the open-source RBPF mapper [Stachniss & Grisetti, 2004]. We compared the overall time, needed to correct the MIT Killian Court dataset and the amount of memory used to store the environment representation. This was done using a (comparably slow) PC with a 1.3 GHz CPU and 1.5 GB RAM. The results of both mapping approaches are summarized in Table 5.4. Since the approximated proposal is not as accurate as the original one, we need more particles to achieve the same to robustness in filter convergence and quality of the resulting maps. However, we can maintain more than one order of magnitude more particles while requiring less runtime and memory. In all our experiments, this sufficiently accounted for the less accurately drawn samples.

The savings on runtime are mainly caused by transforming a computed proposal distribution so that it can be used for several particles instead of computing it from scratch. The memory savings are due to the fact that all particles within a cluster can share their map model. Furthermore, the memory usage and runtime of our approach grows much slower

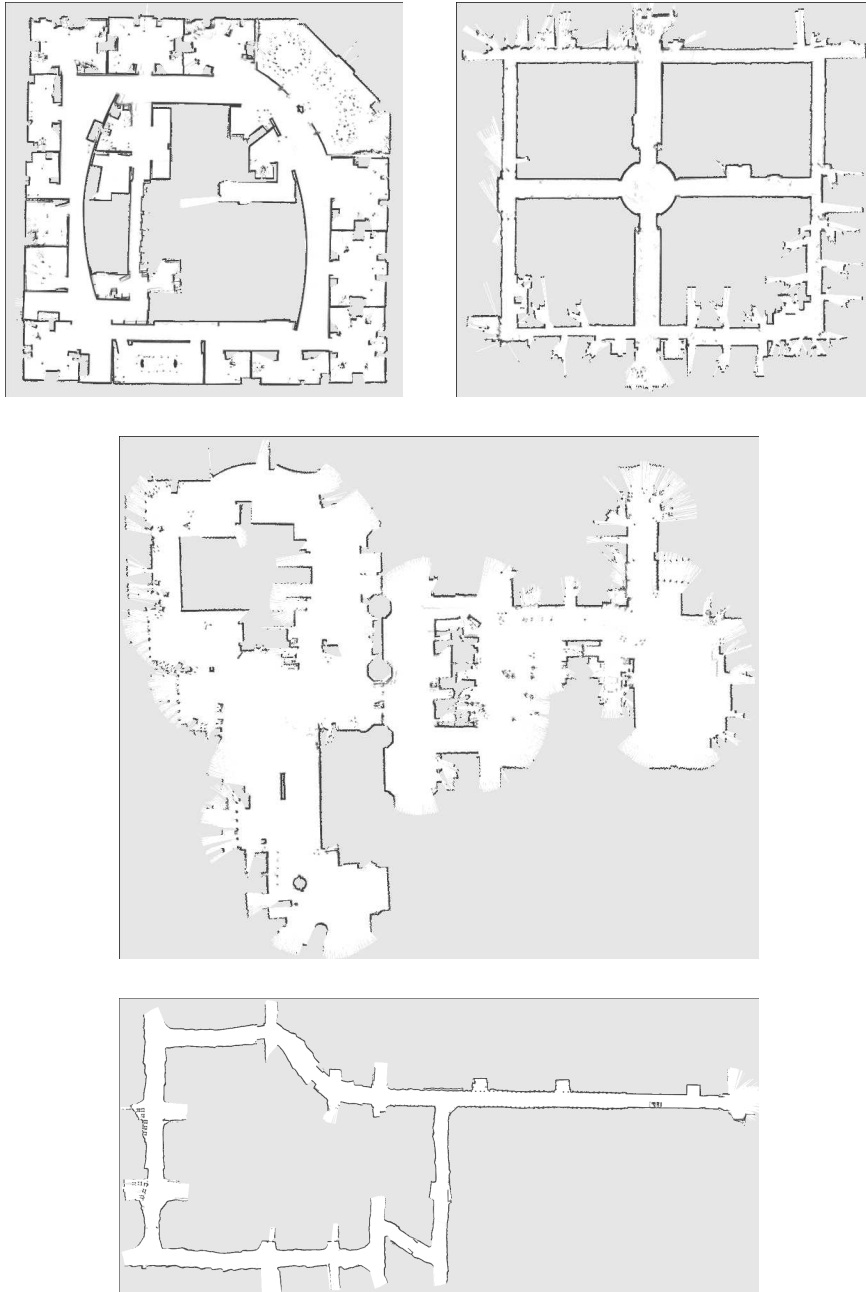


Figure 5.12: Some of the maps generated by the proposed approach: the Intel Lab, the ACES Building at the university of Texas, the Edmonton convention center, and the Bruceton mines.



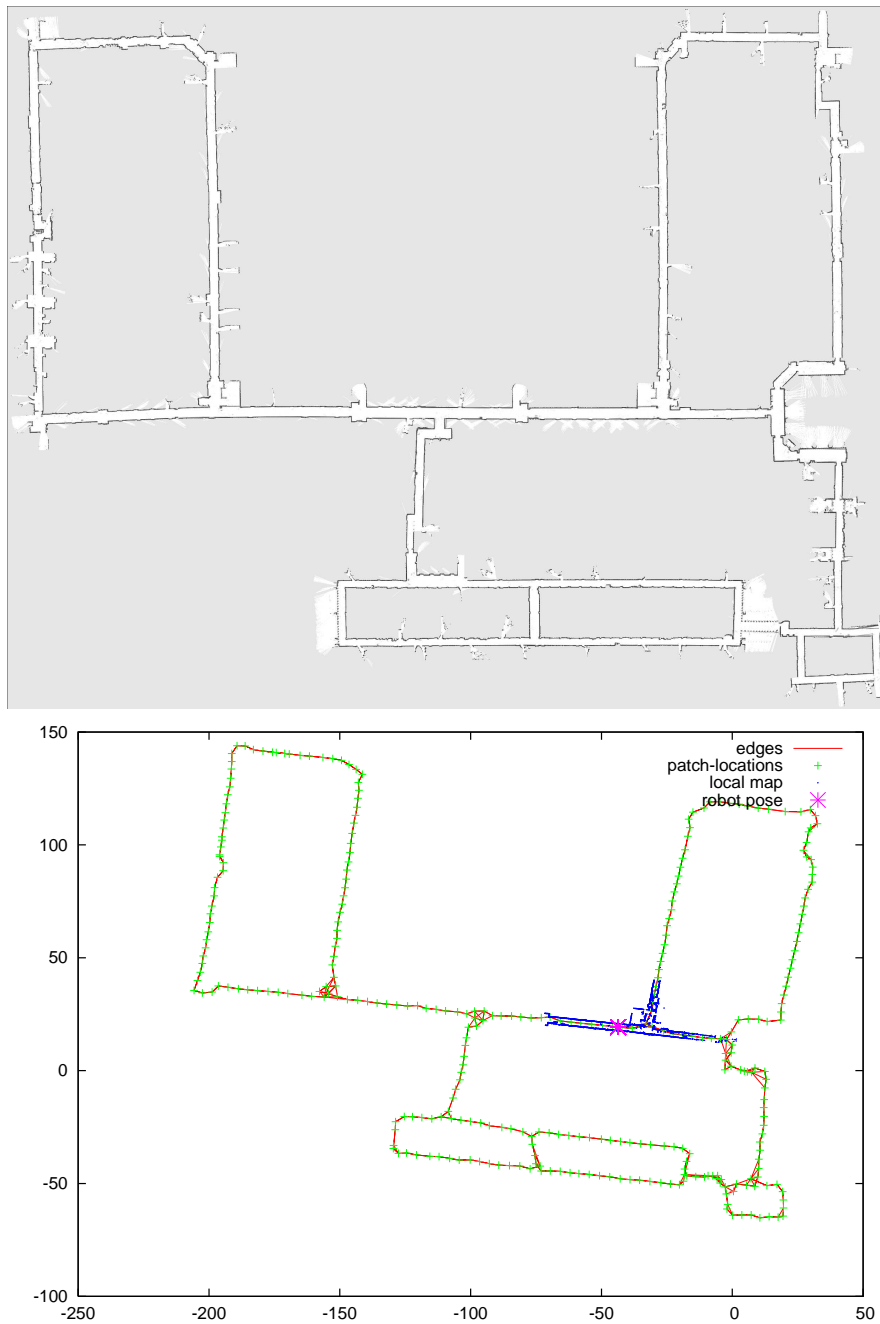


Figure 5.13: The map learned by our approach, using data acquired at the MIT Killian Court. The top picture shows the final result. The bottom picture shows the internal algorithm representation.

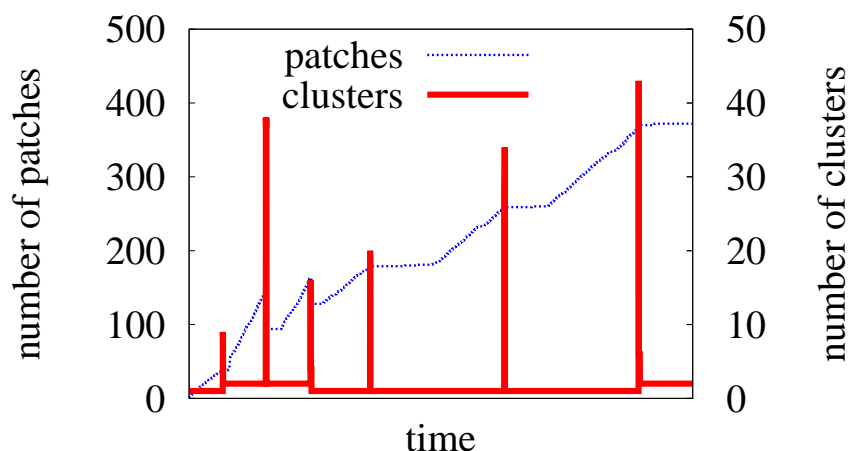


Figure 5.14: This plot depicts the number of patches in the memory and the number of clusters over time for the MIT dataset using 1500 particles.

when adding particles. The reason is that the complexity of our filter grows mainly with the number of topological hypotheses (particle clusters) which need to be maintained and not directly with the number of samples. Notice that the maximum memory usage shown of our approach is much higher than the typical one. There exist a few high peaks in the memory usage which arise from a loop-closure where several clusters are temporarily created but deleted after a few steps (compare Figure 5.14). The typical memory usage is around 30% of the maximum usage.

Figure 5.14 depicts the number of patches in the memory and the number of clusters during the estimation process of the MIT dataset with 1.000 particles. As can be seen, the number of clusters is typically small until a loop-closure occurs. At this point, the number of clusters increases, but only for a short period of time since unlikely clusters vanish quickly.

## 5.8 Connections with Previous Works

The approach presented in this chapter integrates techniques which have been proposed in the context of hierarchical approaches [Thrun, Burgard, & Fox, 2000],[Gutmann & Konolige, 1999] discussed in Subsection 3.3.8, and ideas coming from the Rao-Blackwellized Framework. In particular the idea of tracking the robot pose posterior using a particle filter has been initially proposed in [Thrun, Burgard, & Fox, 2000] and the map representation within a single cluster is similar to [Gutmann & Konolige, 1999]. Moreover, the idea of performing an explicit loop closure is common with both [Thrun, Burgard, & Fox, 2000] and [Gutmann & Konolige, 1999].

However, the technique discussed in this chapter presents several differences with respect to the above works. In our approach, the whole estimate is carried on using a Rao-Blackwellized particle filter in which each particle has its own map instance. This makes it possible to avoid the optimization step which revises the trajectories in the past. The uncer-

tainty reduction after a loop closure is achieved by the particle suppression produced by a resampling step. Although the proposal distribution is computed based on a single hypothesis per cluster, such a proposal is translated according to each sample in the cluster, in order to obtain a sample dependent proposal. This operation is allowed by the local consistency assumption. The loop closure events are recognized as the unique factors which can brake such an assumption. However, in this case the local consistency is recovered by a clustering operation which partitions the samples within a cluster into sets having a locally similar map. Moreover, we introduce a localization phase, in which the map hypotheses are not modified.

## 5.9 Conclusions

In this chapter, we presented an introspection analysis that allows efficient optimizations for Rao-Blackwellized SLAM on grid maps. We are able to update the complex posterior with substantially less resources by performing the computations only for a set of representatives instead of for all potential hypotheses. We proposed an alternative way for representing a distribution over grid maps which needs only a fraction of the memory resources used by previous approaches. Moreover, we proposed an efficient way for updating such a representation.

The key idea is based on an analysis of the mapping process which allows us to perform filter updates *conditioned* to the state of the mapping system: **localization, mapping or loop closing**. Using this insight, we are able to obtain clusters of particles that share a compact map representation as well as an informed proposal distribution to sample the next generation of particles.

With our optimizations, we are able to maintain between one and two orders of magnitude more samples and at the same time require less memory and computational resources compared to other state-of-the-art Rao-Blackwellized mapping techniques.

The approach has been implemented, tested, and evaluated based on real robots and standard log files used within the SLAM community to demonstrate the accuracy as well as the benefits of our system.



## **Part II**

# **Introspective Decomposition of Simultaneous Localization and Mapping**



## Chapter 6

# Motion Clustering and Incremental Estimation

### 6.1 Introduction

When the robot moves in unknown terrain, the uncertainty in its position grows, due to the noise in the odometry sensor and non deterministic effects of the world (e.g. wheel slippage). A first way to reduce this uncertainty is to use exteroceptive sensors to keep track of the robot position over time. When the robot is equipped with range sensors, a common solution is to match two consecutive scans. The most popular of scan matching methods is the Iterative Closest Point (ICP) algorithm [Zhang, 1994].

When operating in urban environments, the detection and the correct motion estimation of cars, people and other dynamic objects can be essential for the safety of the robot and humans nearby. In the presence of dynamic objects, ICP might fail since the presence of spurious dynamic objects can influence the computation of the robot movement. To tackle this problem, current techniques [Hähnel *et al.*, 2003b; Wolf & Sukhatme, 2005] try to detect the parts of the laser scan that are caused by dynamic objects and eliminate them from the robot's motion estimation. Despite the importance of the object detection and tracking problem, most techniques proposed thus far rely on rather ad hoc heuristics and manual parameter tuning.

In this chapter, we address the problem of vehicle tracking in both static and dynamic environments. We show how to compute a local transformation between consecutive robot poses and its uncertainty. This stochastic relationship can be then incorporated into an optimization framework, together with global relationship to provide a consistent map (see [Chapter 8](#) for more details on map optimization).

Since ICP can be considered a reliable technology for scan matching in static environment, the main focus of this chapter will be on how to estimate the robot motion in dynamic ones. Therefore, we present a technique for detecting and estimating the motion of dynamic objects in urban environments based on laser range data. [Figure 6.1](#) shows an example where our robot is facing an intersection while a car is passing by. We can see, that despite the moving object, the static part of the map is correctly aligned and the car detected. We cast

this latter problem as a clustering procedure, where associated points in consecutive laser scans are clustered according to their motion patterns. Common clustering techniques such as EM [Dempster, Laird, & Rubin, 1977] and K-Means [Duda, Hart, & Stork, 2001] assume that points in the data set are independent. Since there is a strong spatial correlation between laser points located nearby, this assumption can severely jeopardize the consistence of these procedures. Frequently, laser returns are generated by rigid objects, thus obeying the Gestalt principles of proximity and common fate [Koffka, 1935]. Semi-supervised clustering [Wagsta *et al.*, 2001; Basu, Banerjee, & Mooney, 2002; Basu *et al.*, 2006] addresses the independent data assumption by enforcing several constraints among cluster assignments. However, these constraints are usually given by an expert and not extracted from the data. We propose a semi-supervised clustering procedure based on Conditional Random Fields (CRFs) [Lafferty, McCallum, & Pereira, 2001]. This significantly simplifies the modeling process and allows the specification of more complex constraints to capture particular aspects of the data. In the particular application of motion clustering, motion patterns are represented by their parameters such as rotation and translation. Our method is able to determine the number of clusters and the corresponding motion patterns while concomitantly computing their parameters. One of the main problems in most of the clustering procedures is to determine the right number of clusters. We provide a simple solution for this problem based on the likelihood of the cluster assignment. This measure is computed automatically during inference in the probabilistic graphical model.

## 6.2 Conditional Random Fields

Since our technique is based on Conditional Random Fields (CRFs), we briefly review the main concepts and how to perform parameter learning and inference in these models.

CRFs are undirected graphical models developed for labeling sequence data [Lafferty, McCallum, & Pereira, 2001]. CRFs directly model  $p(\mathbf{x}|\mathbf{z})$ , the *conditional* distribution over the hidden variables  $\mathbf{x}$  given observations  $\mathbf{z}$ . This is in contrast to generative models such as Hidden Markov Models or Markov Random Fields, which apply Bayes rule to infer hidden states [Rabiner, 1989]. Due to this structure, CRFs can handle arbitrary dependencies between the observations  $\mathbf{z}$ , which gives them substantial flexibility in using high-dimensional feature vectors.

The nodes in a CRF represent hidden states, denoted  $\mathbf{x} = \langle \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n \rangle$ , and data, denoted  $\mathbf{z}$ . The nodes  $\mathbf{x}_i$ , along with the connectivity structure represented by the undirected edges between them, define the conditional distribution  $p(\mathbf{x}|\mathbf{z})$  over the hidden states  $\mathbf{x}$ . Let  $\mathcal{C}$  be the set of cliques in the graph of a CRF. Then, a CRF factorizes the conditional distribution into a product of *clique potentials*  $\phi_c(\mathbf{z}, \mathbf{x}_c)$ , where every  $c \in \mathcal{C}$  is a clique in the graph and  $\mathbf{z}$  and  $\mathbf{x}_c$  are the observed data and the hidden nodes in the clique  $c$ , respectively. Clique potentials are functions that map variable configurations to non-negative numbers. Intuitively, a potential captures the “compatibility” among the variables in the clique: the larger the potential value, the more likely the configuration. Using clique potentials, the conditional distribution over hidden states is written as

$$p(\mathbf{x} | \mathbf{z}) = \frac{1}{Z(\mathbf{z})} \prod_{c \in \mathcal{C}} \phi_c(\mathbf{z}, \mathbf{x}_c), \quad (6.1)$$

where  $Z(\mathbf{z}) = \sum_{\mathbf{x}} \prod_{c \in \mathcal{C}} \phi_c(\mathbf{z}, \mathbf{x}_c)$  is the normalizing partition function. The computation of



this partition function can be exponential in the size of  $\mathbf{x}$ . Hence, exact inference is possible for a limited class of CRF models only.

Potentials  $\phi_c(\mathbf{z}, \mathbf{x}_c)$  are described by log-linear combinations of *feature functions*  $\mathbf{f}_c$ , i.e.,

$$\phi_c(\mathbf{z}, \mathbf{x}_c) = \exp(\mathbf{w}_c^T \cdot \mathbf{f}_c(\mathbf{z}, \mathbf{x}_c)), \quad (6.2)$$

where  $\mathbf{w}_c^T$  is a weight vector, and  $\mathbf{f}_c(\mathbf{z}, \mathbf{x}_c)$  is a function that extracts a vector of features from the variable values. Using feature functions, we rewrite the conditional distribution in [Equation 6.1](#) as

$$p(\mathbf{x} | \mathbf{z}) = \frac{1}{Z(\mathbf{z})} \exp\left\{ \sum_{c \in \mathcal{C}} \mathbf{w}_c^T \cdot \mathbf{f}_c(\mathbf{z}, \mathbf{x}_c) \right\} \quad (6.3)$$

### 6.2.1 Inference

Inference in CRFs can estimate either the marginal distribution of each hidden variable  $\mathbf{x}_i$  or the most likely configuration of all hidden variables  $\mathbf{x}$  (i.e., MAP estimation), as defined in [Equation 6.3](#). Both tasks can be solved using *belief propagation* (BP) [[Pearl, 1988](#)], which works by sending local messages through the graph structure of the model. Each node sends messages to its neighbors based on messages it receives and the clique potentials, which are defined via the observations and the neighborhood relation in the CRF.

BP generates exact results in graphs with no loops, such as trees or polytrees. As our model iterates inferences in a chain graph, BP obtains exact results by propagating messages forward and backward in the chain. In our experiments, we compute the MAP clustering assignment using the max-product version of BP.

### 6.2.2 Pseudo-Likelihood Parameter Learning

The goal of CRF parameter learning is to determine the weights of the feature functions used in the conditional likelihood (see [Equation 6.3](#)). CRFs learn these weights discriminatively by maximizing the conditional likelihood of labeled training data. While there is no closed-form solution for optimizing [Equation 6.3](#), it can be shown that [Equation 6.3](#) is convex relative to the weights  $\mathbf{w}_c$ . Thus, the global optimum of [Equation 6.3](#) can be found using a numerical gradient algorithm. Unfortunately, this optimization runs an inference procedure at each iteration, which can be intractably inefficient in our case.

We therefore resort to maximizing the *pseudo-likelihood* of the training data, which is given by the sum of local likelihoods  $p(\mathbf{x}_i | \text{MB}(\mathbf{x}_i))$ , where  $\text{MB}(\mathbf{x}_i)$  is the Markov blanket of variable  $\mathbf{x}_i$ : the set of the immediate neighbors of  $\mathbf{x}_i$  in the CRF graph [[Besag, 1975](#)]. Optimization of this pseudo-likelihood is performed by minimizing the negative of its log, resulting in the following objective function:

$$L(\mathbf{w}) = - \sum_{i=1}^n \log p(\mathbf{x}_i | \text{MB}(\mathbf{x}_i), \mathbf{w}) + \frac{(\mathbf{w} - \tilde{\mathbf{w}})^T (\mathbf{w} - \tilde{\mathbf{w}})}{2\sigma^2} \quad (6.4)$$

Here, the terms in the summation correspond to the negative pseudo log-likelihood and the right term represents a Gaussian shrinkage prior with mean  $\tilde{\mathbf{w}}$  and variance  $\sigma^2$ . Without additional information, the prior mean is typically set to zero. In our approach, we use

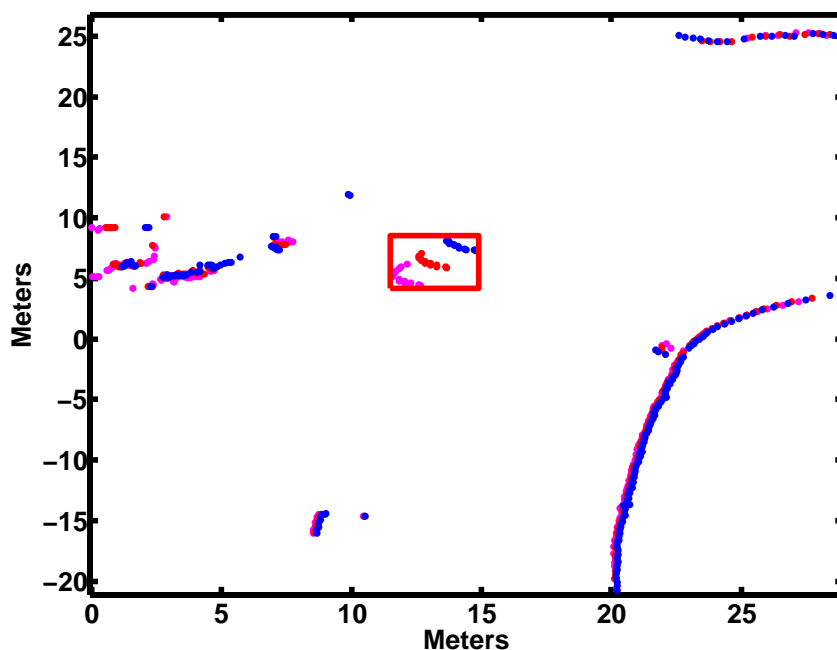


Figure 6.1: A sequence of three consecutive scans obtained by a car navigating in a urban environment. The car is facing an intersection while another one is passing by. The different colors indicate different scans. The laser returns caused by the moving car are indicated by the red square.

unconstrained L-BFGS [Liu & Nocedal, 1989], an efficient gradient descent method, to optimize Equation 6.4. The key advantage of maximizing pseudo-likelihood rather than the likelihood of Equation 6.3 is that the gradient of Equation 6.4 can be computed extremely efficiently, without running an inference algorithm. Learning by maximizing pseudo-likelihood has been shown to perform very well in different domains; see [Kumar & Hebert, 2003; Richardson & Domingos, 2006; Friedman, Fox, & Pasula, 2007; Ramos, Fox, & Durrant-Whyte, 2007].

### 6.3 Motion Estimation in Static Environment

In static environments, the motion of the robot is computed by finding a configuration in the solution space that, if applied to one scan, obtain a maximum overlap with the other one. More formally, the scan matching problem can be expressed as:

*Given two sets of 2D data (i.e. a reference scan,  $s$  and a current scan,  $g$ ), determine a 2D rigid motion (a translation  $T$  and a rotation  $R_\phi$ ) that makes the scan data overlapping the reference data.*

In typical scenarios, however, the perfect overlap is not possible, thus the objective becomes to transform  $s$  in such a way that the distance between the points in  $g$  and the transformed points in  $s$  is minimized. While several algorithms exist (see [Subsection 3.3.1](#) for more details) in this Chapter we focus on the ICP family of algorithms. A naive ICP algorithm has a simple iterative structure. First, a set of correspondent points between the scans is computed and then, the sensor displacement is estimated by minimizing the error of the correspondences. This process is repeated until convergence.

More formally, let  $\{s_1, \dots, s_N\}$  be the points in the current scan  $s$  and  $\{g_1, \dots, g_M\}$  the points in the reference one, each points belonging to  $\mathbb{R}^2$ . Let  $q \in \mathcal{SO}(3)$  be a robot configuration, with  $q_k$  the estimate at iteration  $k$ . The ICP algorithm is a simple iterative procedure that iterate two basic steps: association and minimization. In each iteration  $k$ , points are associated among each other by minimizing the following function

$$r_i = \arg \min_{s_j} \sum_{s_j \in s} \|q_k \oplus s_j - g_i\|^2. \quad (6.5)$$

Once the correspondences are obtained, an incremental solution is estimated by again minimizing the function

$$q_{k+1} = \arg \min_q \sum_{s_i, g_i} \|q \oplus s_i - g_i\|^2. \quad (6.6)$$

Usually, for non linearity issues, the estimation in [Equation 6.6](#) is done step by step. This means that every iteration of the ICP algorithm computes only the transformation between the configuration of step  $k$  and  $k + 1$ , thus minimizing

$$q_{min} = \arg \min_q \sum_{s_i, g_i} \|q \oplus (q_k \oplus s_i) - g_i\|^2. \quad (6.7)$$

and computing  $q_{k+1} = q_{min} \oplus q_k$ .

Since this incremental estimate is performed often, the angular displacement is usually small. For this reason, the nonlinear function can be effectively linearized and the optimization is performed using linear regression techniques. By doing so, [Equation 6.7](#) becomes

$$q_{min} \approx \arg \min_q \sum_{s_i, g_i} \|\hat{q}_0 \oplus (q_k \oplus s_i) - g_i + J_i(q - \hat{q}_0)\|^2. \quad (6.8)$$

$$= \arg \min_q \sum_{s_i, g_i} \|J_i q - (J_i \hat{q}_0 + g_i - \hat{q}_0 \oplus (q_k \oplus s_i))\|^2 \quad (6.9)$$

where  $J_i = \left. \frac{\partial \oplus}{\partial q} \right|_{\hat{q}_0, s_i}$  is the Jacobian of the compound operator and  $\hat{q}_0$  is the linearization point, often set to 0. By stacking everthing together, we obtain the typical form for linear

regression

$$\mathbf{E} = \mathbf{H}q \quad (6.10)$$

$$\mathbf{H} = \begin{bmatrix} J_1 \\ \vdots \\ J_N \end{bmatrix} \quad (6.11)$$

$$\mathbf{E} = \begin{bmatrix} J_1 \hat{q}_0 + g_1 - \hat{q}_0 \oplus (q_k \oplus s_1) \\ \vdots \\ J_N \hat{q}_0 + g_N - \hat{q}_0 \oplus (q_k \oplus s_N) \end{bmatrix} \quad (6.12)$$

whose solution is

$$q_{min} = [\mathbf{H}^T \mathbf{H}]^{-1} \mathbf{H}^T \mathbf{E} \quad (6.13)$$

However, especially when the robot is rotating on itself, this linearization can fail, thus in this thesis we use notions from rigid body physics to compute a closed form minimization step for the non linear problem. In physics terms, the scans are rigid bodies and each correspondence from a point  $g_i$  to a point  $s_j$  can be seen as a linear momentum applied at the point  $s_j$  of the rigid body  $s$ . If we consider discrete time step, this velocity will bring the point  $s_j$  on the point  $g_i$  in one step. The minimization is then computed by computing the overall momentum (translational and angular) using the diverse correspondences ones. As for the linear velocity, we have that

$$v_T = \frac{1}{N} \sum_{i=1}^N g_i - q_k \oplus s_i \quad (6.14)$$

For the rotational one, we need to first compute the angular momentum  $L$  and the inertia  $I$  of the rigid body. These quantities are computed with respect to a rotational center, which is the robot position  $\hat{q}_0$  (usually set to 0).

$$v_A = \frac{\|L\|}{I} \quad (6.15)$$

$$L = \sum_{i=1}^N (\hat{q}_0 \oplus s_i - \hat{q}_0) \times (g_i - \hat{q}_0 \oplus s_i) \quad (6.16)$$

$$I = \sum_{i=1}^N \|g_i - \hat{q}_0 \oplus s_i\|^2 \quad (6.17)$$

where we are only concern about the module of the rotational acceleration, since we already know that the vector is perpendicular to the motion plane.

### 6.3.1 Covariance Estimation

Since we want to use the local estimation obtained from the scan matcher algorithm into an optimization framework, a measure of the uncertainty of this estimation is needed. A first, straightforward solution is the one introduced in [Lu & Milios, 1994] and then improved in [Bengtsson & Baerveldt, 2001]. The key idea is that if we can estimate the displacement

by casting a linear regression problem, we can use the same theory to compute a covariance estimation as well. This can be easily done as

$$\Sigma_q = [\mathbf{H}^T \mathbf{H}]^{-1} \sigma^2 \quad (6.18)$$

An unbiased estimate of  $\sigma^2$  is given by

$$\sigma^2 \approx \frac{1}{N-3} \sum_{s_i, g_i} \|q_{min} \oplus s_i - g_i\|^2 \quad (6.19)$$

being  $N$  the number of correspondences and  $q_{min}$  the solution of the minimization step.

Since we are not relying on the linear regression framework, in this thesis we adopted the approach of Censi [Censi, 2007a]. Let be  $\hat{q}$  the solution of the scan matching algorithm  $A$  and  $z$  the observations. We have that  $q = A(z) = \operatorname{argmin}_q J(q, z)$ , where  $J(q, z)$  represents compactly the error term in Equation 6.6. The first order approximation of the covariance matrix would be

$$\Sigma_q \approx \frac{\partial A}{\partial z} \Sigma_z \frac{\partial A^T}{\partial z} \quad (6.20)$$

A solution for the partial derivative of the function computed by the algorithm,  $A$ , is not easily computed, since the function itself is not in a closed form. However,  $A(z)$  and  $z$  are bounded by an implicit function, since we know that the derivative of the error function is zero at the solution  $q$ . They show that using the implicit function theorem, is possible to express the partial derivative of  $A$  in terms of partial derivatives of  $J$ , obtaining

$$\Sigma_q \approx \left( \frac{\partial^2 J}{\partial q^2} \right)^{-1} \frac{\partial^2 J}{\partial z \partial q} \Sigma_z \frac{\partial J}{\partial z \partial q}^T \left( \frac{\partial^2 J}{\partial q^2} \right)^{-1} \quad (6.21)$$

## 6.4 Motion Clustering and Estimation

In this section the CRF model is extended to address the problem of semi-supervised clustering. Although we describe the methodology for the specific case of motion clustering, the technique is general and can be applied to any clustering problem where the data has some kind of dependence. See Appendix A for a more general description of CRF-Clustering.

In the following, we will explicitly use the rotational  $R$  and translational  $T$  parameter of the configuration  $q$  for clarity reason. Moreover, in the clustering framework, no distinction is made among the robot and the other moving objects. After clustering, the motion of the robot is computed in a way similar to a  $\chi^2$  test. We compute the Mahalanobis distance between the odometry reading and the estimated object motions and classify the motion with the lowest distance as the robot motion.

### 6.4.1 Model Definition

CRF clustering can be understood as a CRF model where local potentials are functions of the hidden variables and, conversely, the hidden variables depend on the local potentials. In the case of motion clustering, the hidden variables are the motion parameters (rotation and translation) and, for every laser beam, there is one hidden variable representing the cluster assignment. In general, there is one hidden variable for every point to be clustered and hidden

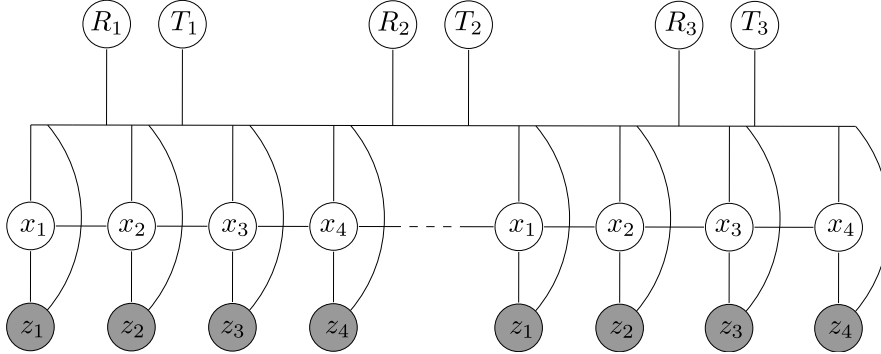


Figure 6.2: Graphical representation of the CRF-Clustering model. The hidden states  $x_i$  indicate the corresponding clusters for each of the pairs of points in the scan. The observations  $z_i$  correspond to local features such as the distance between the points in the pair after the transformation given by the a cluster parameters (rotation and translation).  $R_j$  and  $T_j$  indicate respectively rotation and translation for the clusters.

variables to characterize each cluster. Figure 6.2 shows the graph representation of the model. The hidden variables  $x_i$  represent the cluster assignment and the hidden variables  $R_j$  and  $T_j$  represent rotation and translation respectively, for each cluster in the model. These are the cluster characteristics. (We use blank nodes to represent hidden variables and gray nodes to represent observed variables.) The observations  $z_i$  are computed from the laser scan pair and are described in the section below.

The inputs to the algorithm are two scans, a reference scan  $g$  and second scan  $s$ . The objective is to transform parts of  $s$  according to the clustering assignments in such a way that the distance between the points in  $g$  and the transformed points in  $s$  is reduced. Therefore, for every point  $s_i$ , a hidden variable  $x_i$  is created indicating the clustering assignment. The error function that CRF Clustering minimizes during inference is then:

$$E = \sum_{j=1}^M \sum_{s_i \in C_j} \|g_i - (T_j + R_j s_i)\|^2 \quad (6.22)$$

where  $C_j$  is the set of points in  $s$  assigned to cluster  $j$  and  $\|\cdot\|$  denotes the  $L_2$ -distance.

CRF clustering can be trained as a normal CRF using pseudolikelihood. In our experiments, the model parameters were learned using this technique from of a training data set.

### 6.4.2 Local Feature

The current implementation of CRF clustering has one local feature. It is also possible to add more local features describing, for example, the geometry of sets of points or even appearance from vision data. However, to reduce the inference time we consider only the local feature described below:

**Distance between laser points:** This feature measures the distance between points in

the laser scan  $g$  and the associated points in the laser scan  $s$  after rotated and translated according to the cluster assignments. As different parts of  $s$  are assigned to different clusters this transformation does not preserve the original shape of  $s$ . Instead, it tries to break  $s$  into parts with the same motion pattern. The equation below describes how this feature is computed for every pair of points  $i$ :

$$\mathbf{f}_{\text{dist}}(R_j, T_j, x_i, s_i, g_i) = \frac{\|T_j + R_j s_i - g_i\|^2}{\sigma^2}, \quad (6.23)$$

where  $\sigma^2$  is the variance of the distances in the training data and  $x_i$  defines the cluster  $j$  the particular point  $i$  belongs to.  $R_j$  and  $T_j$  are computed from the points assigned to the cluster  $j$  by minimizing the sum of distances between points  $s$  and corresponding  $g$ :

$$R_j, T_j = \arg \min_{R, T} \sum_{s_j, g_j \in C_j} \|T + R s_j - g_j\|^2. \quad (6.24)$$

The parameters  $R$  and  $T$  are computed in the same way as described in the previous section. This is possible, since within one point cluster the environment can be considered static.

### 6.4.3 Pairwise Features

The pairwise ensures consistency of the clustering assignments by incorporating neighborhood information. For example, if a pair of points  $i$  is assigned to cluster  $j$  it is very likely that the neighbor pair  $i + 1$  will also be assigned to the same cluster. This assumption is true whenever pairs  $i$  and  $i + 1$  belong to the same rigid object. In a common laser scan this is a reasonable assumption since it is expected that clusters . We define three pairwise features with different properties as follows:

**Neighbor Feature:** This is a simple pairwise feature returning two possible values. These values are parameters of the model, estimated during learning. More precisely, the feature is defined as:

$$\mathbf{f}_{\text{neigh}}(x_i, x_{i+1}) = \begin{cases} \lambda_1, & \text{if } x_i = x_{i+1} \\ \lambda_2, & \text{if } x_i \neq x_{i+1} \end{cases} \quad (6.25)$$

where  $\lambda_1$  and  $\lambda_2$  are the parameters of the feature.

**Weighted Neighbor Feature:** This feature is similar to the *Neighbor Feature* except that the output is weighted by the Euclidian distance between the neighbor points. The idea is to capture the notion that neighbor points further away are less dependent than neighbor points nearby.

$$\mathbf{f}_{\text{Wneigh}}(x_i, x_{i+1}, s_i, s_{i+1}) = \begin{cases} \lambda_1/\Delta, & \text{if } x_i = x_{i+1} \\ \lambda_2/\Delta, & \text{if } x_i \neq x_{i+1} \end{cases} \quad (6.26)$$

where  $\Delta$  is the  $L_2$  distance between points  $s_i$  and  $s_{i+1}$ :  $L_2 = \|s_i - s_{i+1}\|^2$ .

**Stiffness Feature:** This feature tries to enforce stiffness for points that belong to the same cluster. The feature computes the difference of distances between neighbor points before and

after the transformation given by the cluster assignment. The idea is that if the points belong to the same cluster, their distances must be preserved after the transformation. This feature can be written as:

$$\mathbf{f}_{\text{stiff}} = \frac{\left\| \left\| s_i - s_{i+1} \right\|^2 - \left\| (T_j + R_j s_i) - (T_k + R_k s_{i+1}) \right\|^2 \right\|^2}{\sigma^2} \quad (6.27)$$

where  $R_j$ ,  $R_k$  and  $T_j$ ,  $T_k$  are the cluster parameters given by the hidden variables  $x_i$  and  $x_{i+1}$  respectively.

#### 6.4.4 Inference Procedure

Performing inference in this model is different from performing inference in a normal CRF. Since the values of the observations change with the hidden states (recall that the local features depend on the cluster assignments and cluster parameters) normal BP cannot be applied. Instead, we formulate a different message passing procedure where with an initial random cluster assignment, the local features are computed. Messages are then propagated back and forward in the chain model to estimate new values for the hidden variables  $\mathbf{x}$ . The new cluster assignments  $\mathbf{x}$  are used to compute new cluster parameters  $R$  and  $T$ . Features are recomputed with these new parameters and we iterate this procedure until convergence. Before each updating of  $R$  and  $T$  the merging procedure described in the next section is performed to reduce and collapse clusters until the right number is obtained.

**Figure 6.3** illustrates the inference process with the three stages: 1) Features are computed from initial values of  $R$  and  $T$ . 2) Messages are propagated forward and backward in the chain graphical model to obtain the MAP assignment for each data point. 3) The cluster assignments are used to recompute  $R$  and  $T$  and these processes are iterated until convergence. After step 3, a merging procedure can be applied to collapse clusters with similar characteristic in order to obtain the right number of clusters.

#### 6.4.5 Computing the Number of Clusters

We now describe a procedure to discover the correct number of clusters. One of the nicest properties of the proposed clustering procedure is to have a probabilistic evaluation of the clustering assignments. The algorithm outputs a likelihood for each laser beam indicating how certain it is regarding the cluster assignment. This can be used to formulate a simple merging procedure where beams that have the probability mass divided into two or more clusters indicate potential cluster candidates for merging.

This idea is the core of the proposed merging procedure. The algorithm is initialized with a large number of clusters (more than the number of moving objects) and as it proceeds, clusters are merged until the correct number is found. In the current implementation, the average cluster assignment likelihood is initially computed. When the average likelihood is below a threshold (usually 0.9), the corresponding cluster can be merged to another. The most probable cluster to merge into is the cluster having most of the remaining probability mass. The pseudocode for the merging algorithm is presented below:



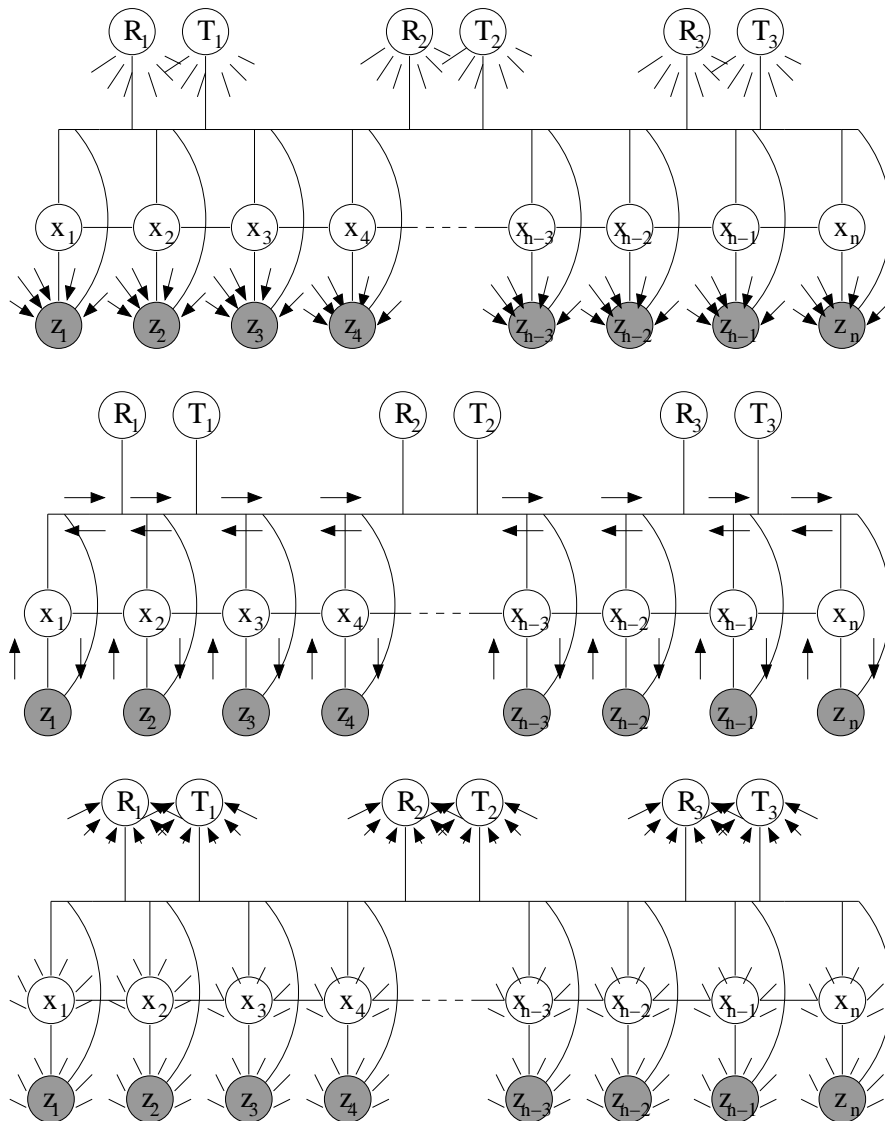


Figure 6.3: Inference procedure in the CRF clustering model. The parameters of the clusters  $R$  and  $T$  are used to compute the local features  $z$  (top). With the observations  $z$  computed, a conventional message passing procedure in a chain graphical model is used to compute the MAP assignments for the clusters  $x$  (middle). Finally, observations and clusters assignments  $x$  are used to recompute the cluster parameters  $R$  and  $T$  (bottom).

---

**Algorithm 8:** Merge Clusters

---

**Input:** Likelihood for Cluster Assignments  $p$  and Cluster Assignment  $C$ **Output:** New Cluster Assignment  $C'$ 

```

1 foreach cluster  $j$  do compute the average likelihood  $\mathcal{L}_j$ ;
2 foreach  $\mathcal{L}_j > 0.9$  do
3   | Find the most compatible cluster  $k$  from  $C$ ;
4   | Merge  $c_j$  and  $c_k$  in the new  $C'$ ;
5 end
6 Returns  $C'$ ;

```

---

## 6.5 Experimental Evaluation

In this section we analyze convergence properties of the algorithm and compare our algorithm to K-Means and to the Consistency-Based Detector (CBD) [Wang *et al.*, 2007] for the problem of motion clustering <sup>1</sup>.

Experiments were performed in an urban environment with a car and consist of 30 pairs of laser scans selected from a trajectory of about 2 km. The data reflect typical driving situations such as cars overtaking other cars, crossing by and moving on the opposite lane. For each pair, the scans were taken at 2m to 4m apart which corresponds to the vehicle motion during the data acquisition. Laser points for each pair were manually assigned to different clusters for ground truth purposes. To evaluate how the approach deals with imperfect data associations between laser scans, we ran our algorithm with both the true, manually generated associations between laser points (CRF-T), and the associations computed automatically via CRF-Matching (CRF-M) [Ramos, Fox, & Durrant-Whyte, 2007].

### 6.5.1 Convergence Properties

In most of the experiments the algorithm converged between 3 and 7 iterations. One particular case is illustrated in Figure 6.4. The algorithm is initialized with 10 clusters which are merged as algorithm iterates. The pictures show the likelihood of each laser beam be assigned to the clusters (dark red is high likelihood, dark blue is low likelihood). Based on the likelihoods, the merging procedure described before is able to combine different clusters to obtain the correct solution.

### 6.5.2 Cluster Evaluation Measure

Once we obtained a clustering solution, we compared it with the ground truth assignments. To evaluate the clustering performance, we used the V-measure [Rosenberg & Hirschberg, 2007], an external, entropy based, cluster evaluation measure. This measure,  $V$ , is the harmonic mean of homogeneity  $H$  and completeness  $C$  of the cluster assignments.

---

<sup>1</sup>CRF-Clustering also estimates the motion for detected objects which is not directly possible with the other methods without using an additional filtering technique

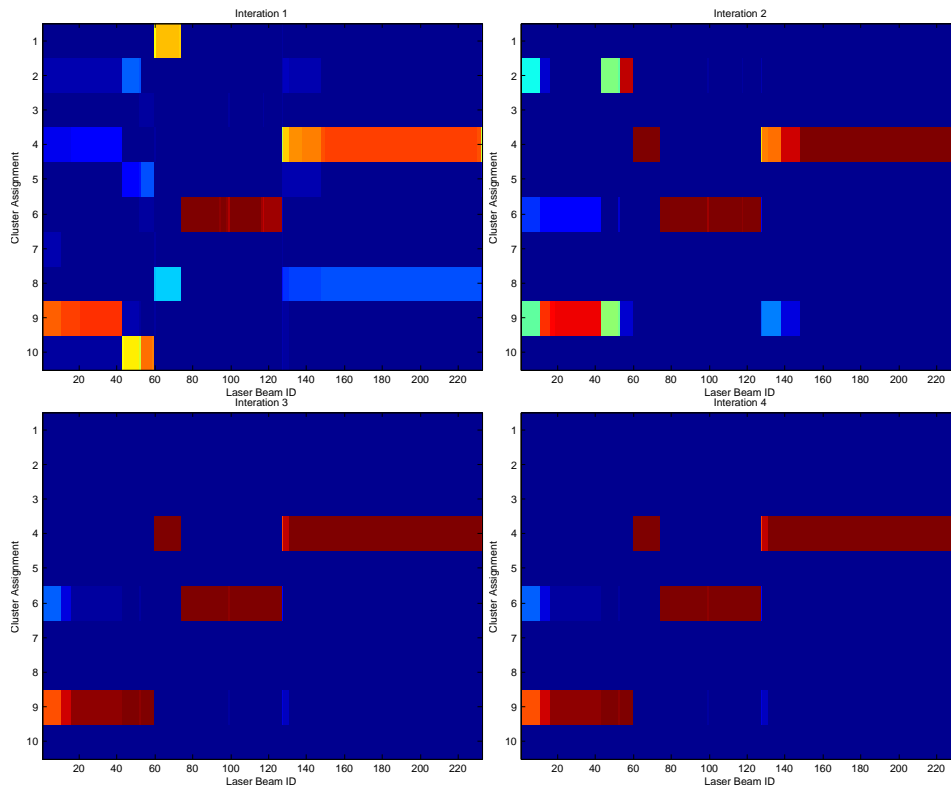


Figure 6.4: Typical convergence behavior for CRF clustering. The pictures show the likelihoods for the clustering assignment. Dark red represents higher likelihoods and dark blue lower likelihoods. After the first iteration, the probability mass is distributed among 8 clusters. In the second iteration, clusters 1 and 8 are merged into cluster 4. In the third iteration, cluster 2 is merged into cluster 9, which gives the final result with 3 clusters only. This sequence corresponds to the scan of [Figure 6.6](#).

Using this measure we avoid some problems that standard “precision” and “recall” have:

- clusters containing few points do not influence the measure;
- matching ground truth clusters and provided ones;
- dealing with homogeneity and completeness of clustering.

V-measure computes a measure of homogeneity and completeness of the clustering solution. Let  $C$  be the distribution of the classes (groundtruth) and  $K$  the distribution of the clusters. Homogeneity reflects the fact that points in one cluster should belong only to one class. This is measured by considering the normalized conditional entropy of the class distribution, given the clusters

$$h = \begin{cases} 1 & \text{if } H(C, K) = 0 \\ 1 - \frac{H(C|K)}{H(C)} & \text{else} \end{cases} \quad (6.28)$$

Symmetrically, completeness reflects the fact that points in one class should be associated only to one clusters. This is measured by considering the normalized conditional entropy of the cluster distribution, given the classes

$$c = \begin{cases} 1 & \text{if } H(K, C) = 0 \\ 1 - \frac{H(K|C)}{H(K)} & \text{else} \end{cases} \quad (6.29)$$

The V-measure is then computed as the harmonic mean of homogeneity and completeness

$$V_\beta = \frac{(1 + \beta)hc}{(\beta h) + c} \quad (6.30)$$

where  $\beta$  is a blending factor. For  $\beta > 1$  completeness is weighted more strongly and for  $\beta < 1$  homogeneity is weighted more strongly. In our experiment we were interested in both, so we set the blending factor to 1.

### 6.5.3 Comparison with Consistency-based Detection

In this section we compare CRF-Clustering with the consistency-based detector (CBD) introduced in [Wang *et al.*, 2007]. The CBD algorithm is a heuristic-based algorithm for detecting moving objects in range data. The main concept behind the algorithm is that static objects are consistent about the *free space* while dynamic objects are not. The major drawback of this algorithm is that it is based on two main assumptions: a good estimate of the robot displacement is available; the object movements are orthogonal to the observed shape. In more detail, let  $g$  be a reference scan and  $s$  the current one. Suppose an estimated pose of the robot is known so that  $s$  and  $g$  are aligned w.r.t. the static objects. On a first stage, the algorithm classifies the points in  $s$  that fall within the free space<sup>2</sup> of  $g$  as dynamic. Points in  $s$  are then segmented in connected regions and a segment is classified as dynamic if more than half of its points are classified as dynamic. While the first assumption often holds in real situations (use of inertial units, GPS, scan matching), the second is more subtle and can create problems especially in outdoor environments.

<sup>2</sup>the free space is defined as the space travelled by the laser beams before encountering an obstacle

	CBD		K-Means		CRF-T		CRF-M	
	mean	std	mean	std	mean	std	mean	std
<b>H</b>	0.8207	0.2790	0.4154	0.2719	0.9832	0.0493	0.8624	0.0521
<b>C</b>	0.8926	0.2976	0.9503	0.0652	0.9899	0.0288	0.9031	0.0312
<b>V</b>	0.8503	0.2901	0.5372	0.2470	0.9864	0.0395	0.8859	0.0407

Table 6.1: Comparison between CBD, K-means and CRF clustering

Figure 6.5 shows a typical example in which the second assumption is violated. The robot is approaching an intersection while another car is moving in front of it. We see (top) that CBD is not able to detect the car. That is because most of the car measurements are not classified as dynamic due to the big overlap. On the other hand, CRF-Clustering (bottom) is able to correctly detect the moving car by clustering laser points according to their motion pattern.

Table 6.1 shows a numerical comparison between the two techniques. Mean and standard deviation for homogeneity, completeness and V-measure are presented for the different approaches. CRF-Clustering obtains better results in both cases, with true data association (CRF-T), and data association using CRF-Matching (CRF-M).

#### 6.5.4 Comparison with Modified K-Means

In this section we compare CRF-Clustering with a modified version of the K-Means algorithm. K-Means is a well known and standard algorithm for clustering data points into  $k$  partitions, minimizing the squared error function

$$E = \sum_{i=1}^k \sum_{x_j \in S_i} \|x_j - \mu_j\|^2 \quad (6.31)$$

where there are  $k$  clusters  $S_i$  and  $\mu_i$  is the mean of all the points  $x_j \in S_i$ . However, K-Means cannot be directly applied to our motion clustering scenario. The problem is that we are clustering points (which are a pair of 2D objects) according to their motion (which is a 3D quantity). For this reason, we modified the original formulation, minimizing Equation 6.22 instead of Equation 6.31. The first modification lies in the way the cluster centroids are computed. In our case, the centroids represent the rigid body transformation underlying the object movement (rotation and translation), which is computed according to Equation 6.24. Once the centroids are obtained, we associate point  $i$  to the cluster which minimize

$$\operatorname{argmin}_j \|g_i - (T_j + R_j s_i)\|^2 \quad (6.32)$$

where  $(g_i, s_i)$  is the point pair,  $T_j$  and  $R_j$  are translation and rotation of the  $j$ -th motion cluster.

The main problem of K-Means and similar algorithms is that they do not consider relations between points in the data. More specifically, they assume that points are independent. When dealing with moving objects, neighbor observations are spatially dependent as they

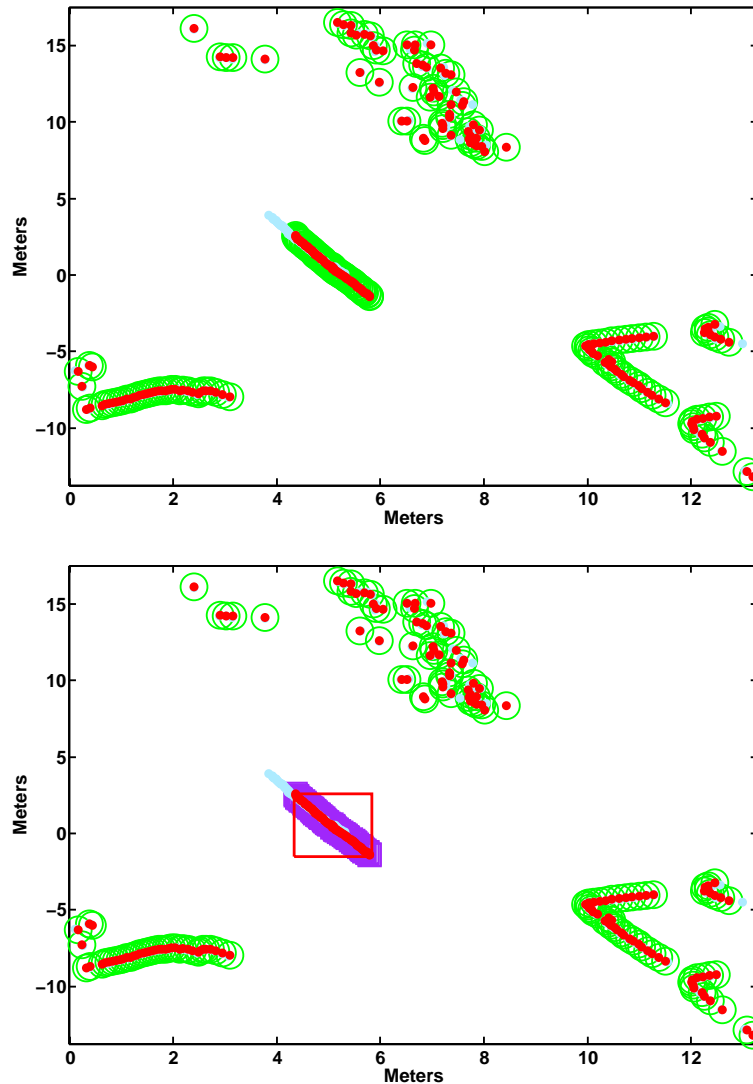


Figure 6.5: Example of consistency-based (CBD) results (top) and CRF-Clustering results (bottom). Red dots indicate the current scan, blue dots the previous one (mostly hidden). The colored markers around dots indicate the cluster assignments. It can be noticed that CBD fails to detect the moving object (all points in the same green cluster). The same problem does not exist with CRF-Clustering which explicitly models the object motion. The static points are in the green cluster, moving object points in the purple cluster. The bounding box on the right image indicates the moving object being correctly detected.

represent measurements from the same entity. Discarding this information can lead to very noisy and inhomogeneous clustering results. This is clearly depicted in [Figure 6.6](#), where we compare the result of K-Means and CRF-Clustering on a typical case. As can be seen, K-Means (top) produces a very noisy result, while the result provided by CRF-Clustering (bottom) is more accurate and homogeneous. Both algorithms were initialized with the maximum number of dynamic objects (three in our case) and we can see that CRF-clustering is able to detect the correct number of clusters.

Finally, [Table 6.1](#) shows a numerical comparison between the two techniques on the data collected by our vehicle. We show the mean and standard deviation for homogeneity, completeness and V-measure for the different approaches. Note that K-Means results are based on the manually generated ground truth associations.

## 6.6 Connection With Previous Works

The first approach to scan matching with ICP is the work of Lu and Milios [[Lu & Milios, 1994](#)]. They combine the normal Nearest Neighbor search of ICP with angular constraints in the Iterative Dual Correspondence (IDC). Pfister *et al.* [[Pfister \*et al.\*, 2002](#)] propose a weighted algorithm, where the influence of each point in the error formulation is weighted according to uncertainty like measurement noise, sensor incidence angle and correspondence error. This error is then minimized using numerical optimization methods. An improved metric is introduced by Minguez *et al.* [[Minguez, Montesano, & Lamiroux, 2006](#)]. The correspondences between scans are established with this measure and the minimization of the error is also carried out in terms of this distance. Censi [[Censi, 2008](#)] introduced a point to line metric and an exact closed-form for minimizing it.

Some researches focused on estimating the uncertainty of the scan matching algorithm. The most rigorous study of the covariance estimation problem has been developed by Bengtsson *et al.* [[Bengtsson & BaerVELdt, 2001](#)]: nevertheless, the two methods proposed there (the Hessian method and the Offline method) have some drawbacks. The closed-form Hessian method over-estimates the covariance in some cases. The Offline method gives reasonable results but cannot be used online, as it is based on a computationally expensive procedure. Censi [[Censi, 2007a](#)] presented a method for estimating the covariance of the ICP algorithm, based on the analysis of the error function being minimized.

As for the dynamic environments, detection and tracking of moving objects using laser range-finders has been extensively studied [[Bar-Shalom & Li, 1995](#)]. A first class of algorithms addresses the detection problem only in terms of separating the data into two main clusters: static and dynamic. The dynamic points are then filtered out to obtain a better motion estimation for the moving platform. Hähnel *et al.* [[Hähnel \*et al.\*, 2003b](#)] presented an offline, EM based approach for filtering moving points in range data. The approach of Wolf and Sukhatme [[Wolf & Sukhatme, 2005](#)] maintains two separate maps for the static and the dynamic parts of an environment. The maps are updated using a modified version of the occupancy grid framework which also infers the nature of the points (static or dynamic).

Another class of algorithms focuses on the object segmentation and tracking. Anguelov *et al.* [[Anguelov \*et al.\*, 2002](#)] use simple differencing for detecting the moving points and then apply a modified EM algorithm for clustering the different objects. However, the algorithm needs the number of objects as input and does not consider interactions between neighbor points. In [[Schulz \*et al.\*, 2001](#)], a feature based approach is used to detect the mov-

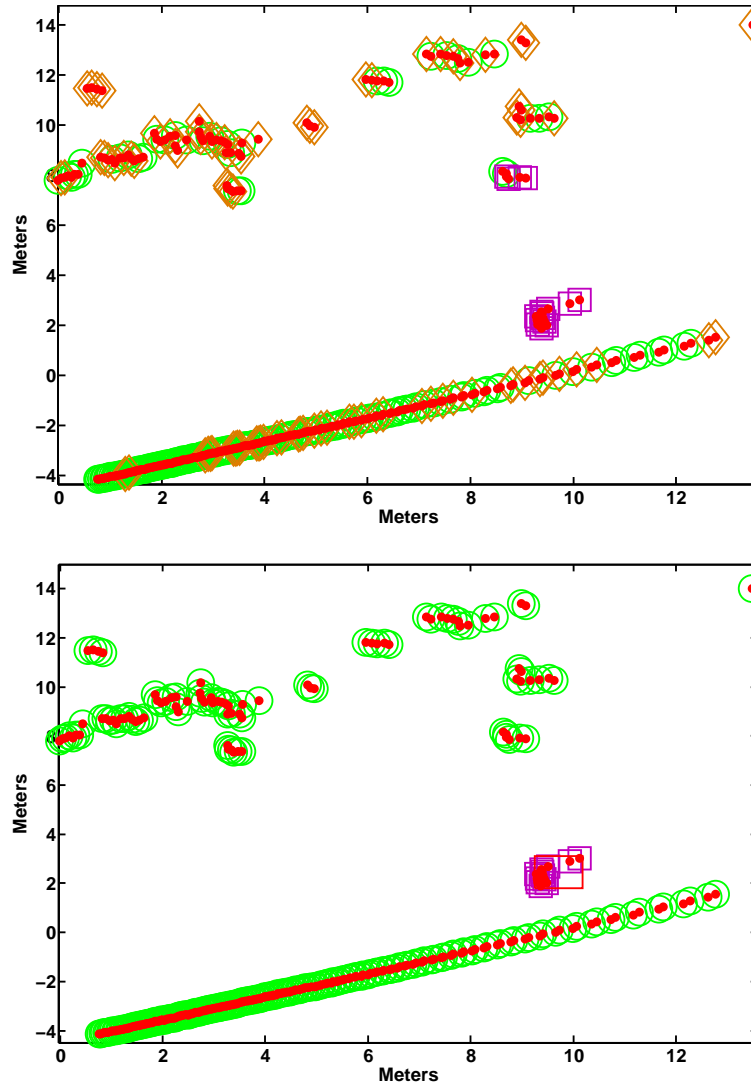


Figure 6.6: Modified K-means results (top) and CRF-Clustering results (bottom). The different symbols indicate the cluster assignment and the red dots indicate the current scan. It can be noticed that the clustering solution found by K-Means is significantly noisy (various different clusters). This is avoided with CRF-Clustering with the use of the neighborhood dependencies. The bounding box on the bottom image indicate the moving object being correctly detected.



ing objects. These objects are then tracked using a joint probabilistic data association filter (JPDAF). The features used are the local minima of the laser data. While such an approach works well in the presence of people, it is not applicable to larger moving objects such as cars, buses and so on. Wang *et al.* [Wang *et al.*, 2007] defined an integrated solution for the mapping and tracking problem: static points are used for mapping while dynamic ones for tracking. The detection and segmentation of dynamic points is based on data differencing and consistency-based motion detection. Points are classified in static and dynamic and clustered in segments. When a segment contains enough dynamic points is considered dynamic. Montesano *et al.* [Montesano, Minguez, & Montano, 2005] improved the classification procedure described in [Wang *et al.*, 2007] by jointly solve it in a Bayesian framework. Moreover, they integrated the mapping and tracking within a path planner for indoor navigation.

Although most of these approaches focus on how to track different objects under different hypotheses, the detection part is mainly based on heuristics. The main technique used is based on map differencing, where points are considered dynamic if there is some inconsistency between two consecutive scans. Moreover, the detection routine is only able to observe the actual position of the object (given a stable reference point) and the velocities are computed by the tracking algorithm. In this paper we address the problem in a more formal way: points are clustered according to their inherent motion while computing the motion parameters.

## 6.7 Conclusions

In this chapter, we have explained how to obtain an incremental estimate of the robot motion, both in static and dynamic environments. When the only moving object in the environment is the robot itself, the problem has been deeply studied, and we described state of the art solutions for both the mean and covariance estimate.

Regarding dynamic environments, we have introduced CRF-Clustering, a novel technique for clustering dependent data into homogeneous partitions. Although this is a general clustering algorithm, in this chapter we show its capability to detect and predict the motion of moving objects from range data. Existing approaches for moving object detection such as CBD are mainly based on scan consistency, classifying points as dynamic if they violate the free space of the map. In contrast, our technique explicitly reasons about the underlying motion of the objects, thus being more effective for the problem. By employing Conditional Random Fields, our approach is able to consider relations among different points in the scans and different properties of the moving objects. Moreover, our algorithm is able to estimate the underlying motion of different objects, which can be used as input to a filtering technique.

Our experiments show that CRF-Clustering performs better than CBD techniques, especially in situations where the motion of the object is not orthogonal to the observed shape. We also showed that this problem is not trivial from a clustering perspective. Classical algorithms, such as K-Means, fail to provide homogeneous clusters, as they assume that data points are independent.



## Chapter 7

# Loop Closing as Lazy Localization

### 7.1 Introduction

In the previous chapter, we discussed how to compute an incremental estimate of the robot motion, both in static and dynamic environments. While this process alleviates the error produced from the odometry and robot motion, it does not eliminate it. The estimation error still accumulates over time and, especially in big and featureless environment, leads to inconsistent global estimates (see [Figure 7.1](#)).

To overcome this problem, global constraints have to be added to the system. In a filtering framework, those constraints are implicitly created within the measurement model. Features are associated to observations regardless of when or where they have been observed. However, data association becomes more difficult when features come from different parts of the map. As also stated in [Chapter 5](#), the current observation is associated with diverse landmarks cliques. These cliques are usually uncorrelated among each other. Here, data association is harder as each clique can push the system towards different parts of the solution space. This fact, convinced various researchers to address the loop closing as a standalone problem and different solutions have been proposed [[Neira, Tardos, & Castellanos, 2003](#); [Fox \*et al.\*, 2003](#); [Levin & Szeliski, 2004](#); [Newman, Cole, & Ho, 2006b](#)]

At this point, it seems that the loop closing problem is the same as global localization one. However, despite some similarity, the two problems have different requirements. In localization, the map is known and complete, meaning that we know a priori that the robot is operating within the environment and cannot be anywhere else. Moreover, localization systems aim to locate the robot precisely and at every time step. Loop closing is, informally, a “lazy localization” problem: the answer can be given lazily, because the higher-level global optimizer does not need this information with stringent time constraints. There is no need to provide the robot position at every time step, the main point is that it has to be reliable and correct.

Moreover, in order to use the loop closing constraints within either a filtering framework or an optimization one, some information about the stiffness of the constraints should be provided. This can be done by estimating a probability distribution of the closure point.

In this Chapter, we address the loop closing problem. We derive a novel formulation of

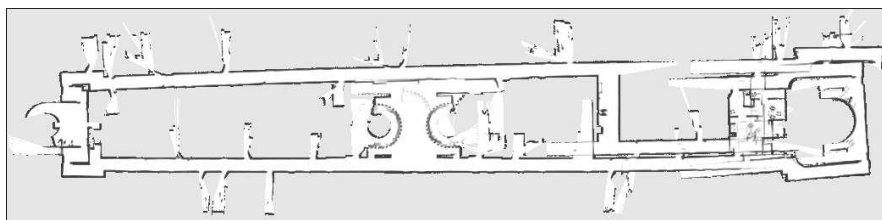


Figure 7.1: An incremental estimate of robot position in a huge, ambiguous environment. As can be seen, the approach was able to correct the local errors, but not the global ones. This results in a map which is still inconsistent.

it in terms of smoothing and propose a technique for computing both the global loop closing constraints and its uncertainty.

## 7.2 Global and Lazy Localization

SLAM methods that handle loop closing explicitly need a constraint between the current pose of the robot and the old part of the map to ensure consistency of the overall map. When the robot reenters a known area, a precise estimation of its position in the map is needed. To provide a reliable estimate, the robot has to reobserve the environment and use these observations to relocalize it self. Several authors casted this problem as global localization on the map built so far [Gutmann & Konolige, 1999],[Fox *et al.*, 2003], that is computing the distribution

$$p(x_n | y_{0:n}, m) \quad (7.1)$$

where 0 is assumed to be the time at which the localization procedure is activated, and  $m$  is the map built before that time. When the pose is disambiguated, a new constraint can be added to the global map graph. Note that Equation 7.1 is a *filtering* distribution: it is the estimate of the last pose based on the past. We argue that in this context knowledge of the *smoothing* distribution

$$p(x_0 | y_{0:n}, m) \quad (7.2)$$

would be more useful. The reason is readily explained: in most cases, a single-shot relocalization is not reliable, and several data are needed to disambiguate the robot position. Given that one should consider an interval of time, it is better to have an estimate of the robot pose at the beginning of the interval, rather than at the end. To see why, consider the canonical example of a loop closure in Figure 7.2. A global localization formulation would give a constraint between  $x_B$  and  $x_F$ , resulting in an inconsistent map. The lazy localization formulation, instead, returns a constraint between  $x_A$  and  $x_E$ , giving a better map estimation.

The two solutions, for time 0 and  $k$ , seems to be equivalent at first sight. However, we argue that the solution should be the closure position and not the actual pose. Assume that the robot re-enters a previously mapped area at time 0. A method that gives an estimate of the pose at time 0 is more useful than a method that gives an estimate at time  $k$ . In fact, all the information needed for loop closure is in the estimate of  $x_0$ , and imposing a loop

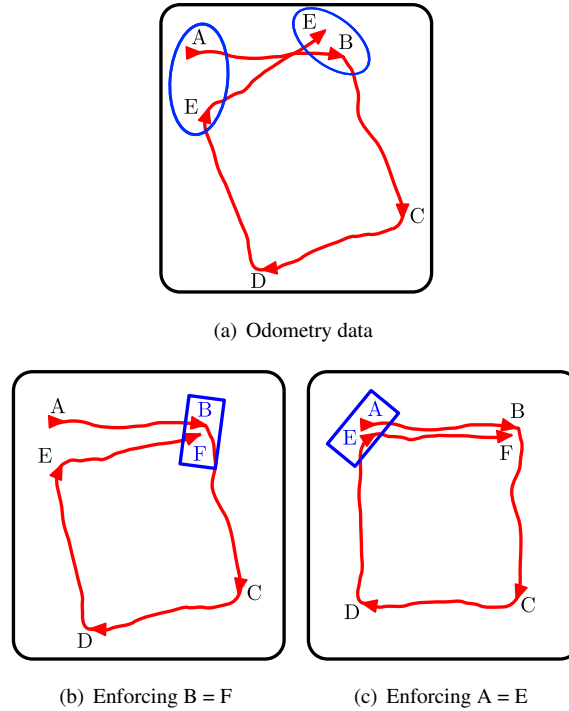


Figure 7.2: This picture illustrates the difference between modelling loop closing as smoothing or as filtering. **Figure 7.2(a)** shows the the odometry data of a robot moving in a square environment. The two paths A–B and E–F correspond to the same corridor; assume that the robot can relocalize itself while traversing this corridor. If one models loop closing as filtering, one would get the constraint  $B = F$  which corresponds to the resulting map in (b). If one models loop closing as smoothing, the result is the constraint  $A = E$ , which results in the more consistent map in (c).

closure constraint at time  $k$  would probably create an inconsistent map. The information lost in closing at  $k$  is represented by the missed association during the path from 0 to  $k$ .

Not only the smoothing distribution is more useful, we will show it can also be computed efficiently if a reasonably precise estimate of the incremental robot motion is available. An example of this is the estimate given by a scan-matcher, which is already available at zero-cost if localization is done as a subprocess of SLAM.

More formally, we are set to compute the distribution

$$p(\mathbf{x}_0 | \mathbf{y}_{0:n}, \mathbf{s}_{0:n}, \mathbf{m}) \quad (7.3)$$

where  $\mathbf{s}$  is an estimate of the incremental motion of the robot:  $\mathbf{s}_{i:j} \triangleq \mathbf{x}_j \ominus \mathbf{x}_i$ . We will present three algorithms that compute **Equation 7.3**: the first two are slight modifications of particle filters already studied in the literature, while the Frozen-Time Smoother is the novel contribution of this thesis.

From now on, we omit the map  $m$  from the equations, and use the abbreviations  $\mathbf{y}_: = \mathbf{y}_{0:n}$ ,  $\mathbf{s}_: = \mathbf{s}_{0:n}$ .

### 7.3 Particle Filter Approaches

In this section we develop two approximated solutions to the smoothing problem, as the cost of an exact implementation using Monte Carlo techniques (particle smoother) is quadratic in the number of particles.

The density in Equation 7.2 can be factored as

$$\begin{aligned}
 & p(\mathbf{x}_0 | \mathbf{y}_{0:n}) \\
 &= \int p(\mathbf{x}_0, \mathbf{x}_1 | \mathbf{y}_{0:n}) d\mathbf{x}_1 \\
 &= \int p(\mathbf{x}_1 | \mathbf{y}_{0:n}) p(\mathbf{x}_0 | \mathbf{x}_1, \mathbf{y}_0) d\mathbf{x}_1 \\
 &= p(\mathbf{x}_0 | \mathbf{y}_0) \int \frac{p(\mathbf{x}_1 | \mathbf{y}_{0:n}) p(\mathbf{x}_1 | \mathbf{x}_0) d\mathbf{x}_1}{\int p(\mathbf{x}_1 | \mathbf{x}_0) p(\mathbf{x}_0 | \mathbf{y}_0) d\mathbf{x}_0}
 \end{aligned} \tag{7.4}$$

where the map,  $m$ , is omitted for clarity purposes. We have decomposed the problem into a forward recursion up to  $p(\mathbf{x}_n | \mathbf{y}_n)$ , which is solved by forward filtering, and a backward recursion,  $p(\mathbf{x}_{t+1} | \mathbf{y}_{t:n})$ , which is solved by propagating the smoothed density at time  $t + 1$  back to time  $t$  by using the motion model  $p(\mathbf{x}_{t+1} | \mathbf{x}_t)$ . This factorization is the basis of the Forward-Backward Smoothing algorithm. For some distribution family, this can be solved by applying the Expectation-Maximization algorithm. However, there exist no optimal solution to those integrals when using grid maps.

We decide to approximate the density with a sum of weighted samples

$$\hat{p}(d\mathbf{x}_0 | \mathbf{y}_{0:n}) = \sum_{i=1}^N w_{0|n}^{(i)} \delta_{\mathbf{x}_0}^{(i)}(d\mathbf{x}_0) \tag{7.5}$$

where the importance weight are recursively defined as

$$w_{t|n}^{(i)} = w_t^{(i)} \left[ \sum_{j=1}^N w_{t+1|n}^{(j)} \frac{p(\mathbf{x}_{t+1}^{(j)} | \mathbf{x}_t^{(i)})}{\sum_{k=1}^N p(\mathbf{x}_{t+1}^{(j)} | \mathbf{x}_t^{(k)})} \right] \tag{7.6}$$

with  $w_{n|n}^{(i)} = w_n^{(i)}$ . A direct implementation of this filter incurs in a  $O(N^3)$  cost, which can be reduced to a  $O(N^2)$  considering that the denominator of the right equation does not depend on  $x_t^{(i)}$ . Even with this simplification, the cost is still too expensive for using the algorithm in practical application. A typical amount of particle for localization is in the order of 10000, preventing a real time usage of the particle smoother.

#### 7.3.1 Approximate Particle Smoothing

As a first approximation, we decide to use a slightly modified Monte Carlo localization algorithm. The idea is that one can use a vanilla particle filter and remember the first pose of

the particles. Then the distribution of the first poses, weighted according to the distribution of the last poses, can be assumed as a crude approximation to [Equation 7.3](#). In the following, this is referred to as Monte Carlo Smoothing v1 (MCS-1).

The algorithm has a cost complexity  $O(N)$ , thus the same complexity as Monte Carlo localization. In fact, the algorithm is a variation of Monte Carlo localization, with the difference that we store, for each particle, also the initial pose. After filtering, we return the initial pose of every particle, but with the weight of the particles on the last step. However, the state space is now the space of the *whole* trajectory and using a filtering approximation results in a poor approximation of the true distribution. The dimension of the state space is exponential in the length of this trajectory, so more particles than global localization are needed. Secondly, the particles representing the initial pose do not “move” towards the most probable regions, resulting in an overconfident estimate.

---

**Algorithm 9:** Monte Carlo Smoothing v1
 

---

**Input:** observation sequence  $\mathbf{y}_{0:n}$  and map  $\mathbf{m}$

**Output:** weighted samples  $\left\{ \left\langle \mathbf{x}_{0|n}^{(i)}, w_{0|n}^{(i)} \right\rangle \right\}_{i=1}^N$

*// Forward Filtering*

```

1 for  $t \leftarrow 0$  to  $n$  do
2   for  $i \leftarrow 1$  to  $N$  do
3     sample  $\mathbf{x}_t^{(i)}$  from  $p(\mathbf{x}_t | \mathbf{x}_{t-1}^{(i)})$ ;
4     augment the trajectory  $\mathbf{x}_{0:t}^{(i)}$ ;
5      $w_t^{(i)} \leftarrow p(\mathbf{y}_t | \mathbf{x}_t^{(i)}, \mathbf{m})$ ;
6   end
7   resample the particles according to  $w_t$ ;
8 end

```

*// Recovering the first poses*

```

9 result =  $\emptyset$ ;
10 for  $i \leftarrow 1$  to  $N$  do
11   result  $\leftarrow$  result  $\cup \left\langle \mathbf{x}_0^{(i)}, w_n^{(i)} \right\rangle$ ;
12 end
13 return result;

```

---

The following is the formal justification of the approximation. Consider the target distribution in [Equation 7.2](#), according to the Chapman-Kolmogorov equation with respect to  $\mathbf{x}_0$ , we have

$$p(\mathbf{x}_0 | \mathbf{y}_{0:k}) = \int p(\mathbf{x}_0 | \mathbf{x}_k, \mathbf{y}_{0:k}) p(\mathbf{x}_k | \mathbf{y}_{0:k}) d\mathbf{x}_k \quad (7.7)$$

Now note that the distribution  $p(\mathbf{x}_k | \mathbf{y}_{0:k})$  is the filtering distribution, for which we have the

particle representation

$$p(\mathbf{x}_k = x | \mathbf{y}_{0:k}) = \sum_i w^{(i)} \delta(x - \mathbf{x}_k^{(i)}) \quad (7.8)$$

Here, we have written the free variable  $x$  explicitly, as we need it later. The other distribution in [Equation 7.7](#) is  $p(\mathbf{x}_0 | \mathbf{x}_k, \mathbf{y}_{0:k})$ , for which we implicitly have the following particle approximation:

$$p(\mathbf{x}_0 = a | \mathbf{x}_k = b, \mathbf{y}_{0:k}) = \sum_j \delta((b \ominus a) - (\mathbf{x}_k^{(j)} \ominus \mathbf{x}_0^{(j)})) \quad (7.9)$$

which represent the displacement between the pose of particle  $j$  at time  $k$  and its stored first pose at time 0. By putting [Equation 7.8](#) and [Equation 7.9](#) in [Equation 7.7](#), one obtains:

$$p(\mathbf{x}_0 = a | \mathbf{y}_{0:k}) = \int p(\mathbf{x}_k = x | \mathbf{y}_{0:k}) p(\mathbf{x}_0 = a | \mathbf{x}_k = b, \mathbf{y}_{0:k}) dx \quad (7.10)$$

$$= \int \sum_i w^{(i)} \delta(x - \mathbf{x}_k^{(i)}) \sum_j \delta((b \ominus a) - (\mathbf{x}_k^{(j)} \ominus \mathbf{x}_0^{(j)})) dx \quad (7.11)$$

$$= \sum_i w^{(i)} \delta(a - \mathbf{x}_0^{(i)}) \quad (7.12)$$

As can be seen, that reflects the algorithm. The Dirac's delta are centered on the first pose of the particle, and the weights are the same of the filtering step. A concise description of the algorithm is given in [Algorithm 9](#)

### 7.3.2 Improving Approximation by using a Scan Matcher

A less crude approach uses the assumption that the incremental estimate of the pose is precise. The smoothing distribution can be factorized as

$$p(\mathbf{x}_0 | \mathbf{y}_:, \mathbf{s}_:) = \int p(\mathbf{x}_0 | \mathbf{x}_n, \mathbf{y}_:, \mathbf{s}_:) p(\mathbf{x}_n | \mathbf{y}_:, \mathbf{s}_:) d\mathbf{x}_n \quad (7.13)$$

The factorization split the target distribution into the filtering distribution  $p(\mathbf{x}_n | \mathbf{y}_:, \mathbf{s}_:)$  and the inverse informed motion model  $p(\mathbf{x}_0 | \mathbf{x}_n, \mathbf{y}_:, \mathbf{s}_:)$ . Intuitively, the question “Where was I at time 0?”, is split into “Where am I now?” and “What was my incremental motion?”.

In SLAM, an incremental estimate is readily available as the scan-matcher result. If we make the assumption that the error of the scan matcher is very small, we can approximate  $p(\mathbf{x}_0 | \mathbf{x}_n, \mathbf{y}_:, \mathbf{s}_:)$  as a Dirac distribution. Therefore, the integral in [Equation 7.13](#) is greatly simplified, and the target distribution is approximated by translating the estimate at time  $n$  back in time according to the incremental estimate  $\mathbf{s}_{0:n}$ .

$$p(\mathbf{x}_0 = x | \mathbf{y}_{0:k}) = \int p(\mathbf{x}_0 = x | \mathbf{x}_k, \mathbf{y}_{0:k}) p(\mathbf{x}_k | \mathbf{y}_{0:k}) d\mathbf{x}_k \quad (7.14)$$

$$= \int \sum_i w^i \delta(x - \mathbf{x}_k^i) \delta(\mathbf{x}_k \ominus \hat{\mathbf{s}}_{0:k}) d\mathbf{x}_k \quad (7.15)$$

$$= \sum_i w^i \delta(x - (\mathbf{x}_k^i \ominus \hat{\mathbf{s}}_{0:k})) \quad (7.16)$$



This approach, which we call Monte Carlo Smoothing v2(MCS-2), is sketched as [Algorithm 10](#). The modifications with respect to a Monte Carlo Localization is that the incremental estimate is used, both for evolving the particles, and to translate them back to time 0. In MCS-1 the particles do not “move” within the state space, eventually resulting in few of them having non-zero importance weight. In MCS-2, thanks to the backward translation by the scan-matcher result, the particles *do* “move” within the state space, resulting in more particle diversity. This effect is depicted in [Figure 7.4](#).

---

**Algorithm 10:** Monte Carlo Smoothing v2
 

---

**Input:** observation sequence  $\mathbf{y}_{0:n}$ , incremental estimate  $\mathbf{s}_{0:n}$  and map  $\mathbf{m}$

**Output:** weighted samples  $\left\{ \left\langle \mathbf{x}_{0|n}^{(i)}, w_{0|n}^{(i)} \right\rangle \right\}_{i=1}^N$

*// Forward Filtering*

```

1 for  $t \leftarrow 0$  to  $n$  do
2   for  $i \leftarrow 1$  to  $N$  do
3     sample  $\mathbf{x}_t^{(i)}$  from  $p(\mathbf{x}_t | \mathbf{x}_{t-1}^{(i)})$ ;
4      $w_n^{(i)} \leftarrow p(\mathbf{y}_n | \mathbf{x}_n^{(i)}, \mathbf{m})$ ;
5   end
6   resample the particles according to  $w_n$ ;
7 end

```

*// Backward Projection*

```

8 result =  $\emptyset$ ;
9 for  $i \leftarrow 1$  to  $N$  do
10   // Translate back in time
11    $\mathbf{x}_{0|n}^{(i)} \leftarrow \mathbf{x}_n^{(i)} \ominus \mathbf{s}_{0:n}$ ;
12    $w_{0|n}^{(i)} \leftarrow w_n^{(i)}$ ;
13   result  $\leftarrow$  result  $\cup \left\langle \mathbf{x}_{0|n}^{(i)}, w_{0|n}^{(i)} \right\rangle$ ;
14 end
15 return result;

```

---

## 7.4 The Frozen-Time Smoother

In this section, we introduce the “Frozen Time Smoother” (FTS), an approximate algorithm for the lazy localization problem.

For this algorithm to work, we need two assumptions.

- an estimate of the *incremental* pose of the robot is available, roughly precise during the time it takes to localize (we will discuss the precision requirements later).

- a fast way to compute a “translated” likelihood is available.

The first assumption is satisfied by using a scan matcher algorithm, or a visual odometry one. As for the second, in this thesis, we use a Generic Hough Transform (GHT)-like [Ballard, 1981] algorithm, whose input is raw data. If one wants to use features, a completely equivalent algorithm that could be plugged here is the one described by Paz *et al.* [Paz *et al.*, 2005].

If the two assumptions hold, we can find a particularly simple factorization of the target distribution. The sensor data  $\mathbf{y}_k$  can be considered independent when conditioning on both the initial state  $\mathbf{x}_0$  and the incremental motion  $\mathbf{s}_{k:0}$ :

$$p(\mathbf{x}_0|\mathbf{y}_:, \mathbf{s}_:) \propto p(\mathbf{x}_0) \prod_k p(\mathbf{y}_k|\mathbf{x}_0, \mathbf{s}_{0:k}) \quad (7.17)$$

Again, using the Chapman-Kolmogorov equation to  $p(\mathbf{y}_k|\mathbf{x}_0, \mathbf{s}_{0:k})$  with respect to  $\mathbf{x}_k$ , we obtain

$$p(\mathbf{x}_0|\mathbf{y}_:, \mathbf{s}_:) \propto p(\mathbf{x}_0) \prod_k \int p(\mathbf{y}_k|\mathbf{x}_k) p(\mathbf{x}_k|\mathbf{x}_0, \mathbf{s}_{0:k}) d\mathbf{x}_k \quad (7.18)$$

Having a precise estimate of the robot motion, one can approximate the multi-step inverse motion model  $p(\mathbf{x}_k|\mathbf{x}_0, \mathbf{s}_{k:0})$  using a Dirac distribution

$$p(\mathbf{x}_k = a|\mathbf{x}_0 = b) \simeq \delta(b \ominus a - \mathbf{s}_{k:0}) \quad (7.19)$$

With this approximation, the integral in Equation 7.18 can be easily solve, obtaining

$$p(\mathbf{x}_0 = a|\mathbf{y}_:, \mathbf{s}_:) \propto p(\mathbf{x}_0 = a) \prod_k p(\mathbf{y}_k|\mathbf{x}_k = a \oplus \mathbf{s}_{0:k}) \quad (7.20)$$

The right hand side is a product of “translated likelihoods”: each likelihood at time  $k$  is translated by the motion  $\mathbf{s}_{0:k}$ . In short, because the incremental estimate is available and is precise, we can compute the likelihood at time 0 very efficiently.

FTS uses a three-dimensional  $(x, y, \theta)$  grid as the representation of the belief, at the *freezing time* 0. The grid is used essentially as a voting space, therefore its resolution is relatively unimportant (in the experiments, we set the resolution to 1m, 30deg). The grid will represent in turn  $p(\mathbf{x}_0)$  (prior),  $p(\mathbf{x}_0|\mathbf{y}_0)$ ,  $p(\mathbf{x}_0|\mathbf{y}_{0:1})$ , etc. Note that the scans can be integrated in arbitrary order, and some can be postponed or skipped altogether.

### 7.4.1 Translated Likelihood Computation by GHT

The map representation is contingent on using the GHT for computing the likelihood. Points are sampled from the surfaces in the environment, and for these points the surface orientation is estimated. The result is a “normal map”: a series of tuples  $\langle \mathbf{p}_j, \alpha_j \rangle$  where  $\mathbf{p} \in \mathbb{R}^2$  is a point on the environment surfaces and  $\alpha_j$  is the direction of its normal. An example of such a map is shown in Figure 7.3. We remark that this is an extremely compact representation. Every scan is converted in the same way to a local normal map. The local and the global normal map are passed to the GHT algorithm.

The GHT creates hypotheses by considering all possible correspondences between the local and normal map points. These hypotheses (can also be thought as ‘votes’) are accumulated in a grid. At the end of the computation, one can assume that this is an approximation to  $p(\mathbf{x}_k|\mathbf{y}_k)$ .

More in detail, if the  $i$ -th point of the local map corresponds to the  $j$ -th point on the global map, then the pose of the robot must be

$$\hat{\mathbf{x}}_k = (\hat{\mathbf{t}}_k, \hat{\theta}_k) \quad (7.21)$$

where  $\hat{\mathbf{t}}_k$  and  $\hat{\theta}_k$  are given by:

$$\hat{\theta}_k \leftarrow \alpha_j - \alpha_i \quad (7.22)$$

$$\hat{\mathbf{t}}_k \leftarrow \mathbf{p}_j - \mathbf{R}(\hat{\theta}_k) \mathbf{p}_i \quad (7.23)$$

And now to the trick that makes everything work: because we know the motion the robot did from  $\mathbf{x}_0$  to  $\mathbf{x}_k$ , we can compute directly  $p(\mathbf{x}_0|\mathbf{y}_k)$ . In the GHT loop, we simply translate  $\hat{\mathbf{x}}_k$  back by  $s_{0:k}$ , to obtain the hypothesis for  $\hat{\mathbf{x}}_0$ :

$$\hat{\mathbf{x}}_0 \leftarrow \hat{\mathbf{x}}_k \ominus \hat{s}_{0:k} \quad (7.24)$$

After the distribution  $p(\mathbf{x}_k|\mathbf{y}_0)$  has been computed, this new information is integrated in the belief by a simple multiplication, according to [Equation 7.20](#). [Algorithm 11](#) shows a pseudo-code for FTS. The algorithm is not complex and can be described in a few lines of text.

---

**Algorithm 11:** Frozen-Time Smoother
 

---

**Input:**

- a “freezing time” (0)
- a set of sensor scans  $\mathbf{y}_0, \mathbf{y}_1, \dots$
- an incremental estimate of the pose  $s_{0:k}$
- (optional) a prior for  $\mathbf{x}_0$

**Output:** a grid estimate of  $p(\mathbf{x}_0)$ 

- 1 Create the reference normal-map refMap
  - 2 Initialize the belief  $bel$  to the prior  $p(\mathbf{x}_0)$
  - 3 **for** some  $\mathbf{y}_k$ , in arbitrary order **do**
  - 4     Create a local normal-map localMap from scan  $\mathbf{y}_k$
  - 5      $p(\mathbf{x}_0|\mathbf{y}_k) \leftarrow \text{ght}(\text{refMap}, \text{localMap}, \hat{s}_{0:k})$
  - 6      $bel \ *= \ p(\mathbf{x}_0|\mathbf{y}_k)/p(\mathbf{x}_0)$
  - 7 **end**
- 

The computational complexity of FTS can be easily computed. At every iteration (one for observation) the algorithm computes the GHT. GHT is essentially a double loop. The outer loop iterates on the number of the global maps points, which is fixed and depends on the dimension of the readings. The inner loop, instead, depends on the number of points in

the map and grows linearly with it. As the map is built while the robot is moving, we can say that the number of points in the map is proportional to the length of the robot trajectory. The computational complexity of a naive implementation of FTS is then  $O(K)$ , where  $K$  is the number of readings used to built the map. This complexity, however, can be reduced if a robot prior is used as a search area.

---

**Function** ght (refMap, localMap,  $\hat{s}$ )

---

```

1 for  $\langle p_i, \alpha_i \rangle \in \text{localMap}$  do
2   for  $\langle p_j, \alpha_j \rangle \in \text{refMap}$  do
3     // Compute an estimate of the pose at time k
4      $\hat{\theta}_k \leftarrow \alpha_j - \alpha_i$ 
5      $\hat{t}_k \leftarrow p_j - R(\hat{\theta}_k) \cdot p_i$ 
6      $\hat{x}_k \leftarrow \langle \hat{t}_k, \hat{\theta}_k \rangle$ 
7     // Use the displacement  $\hat{s}$  and the estimate  $\hat{x}_k$ 
8     // to compute the pose at time 0
9      $\hat{x}_0 \leftarrow \hat{x}_k \ominus \hat{s}$ 
10    // Add a vote for  $\hat{x}_0$ 
11    buffer [ $\hat{x}_0$ ] ++
12  end
13 end
14 return buffer
```

---

## 7.5 Discussion

Quantitative results are discussed in the next section. Here, we would like to point out some interesting properties of FTF.

### 7.5.1 On the Use of a Grid

FTS uses a three-dimensional grid, but no expensive operation is performed on the grid. Compare with using plain Markov localization, in which one has to perform a convolution on the grid (prediction step), and compute the likelihood for every cell (update step). Also note that Markov Localization would need small cells, or else it would make little sense to compute the likelihood for a  $1\text{m} \times 1\text{m}$  cell.

Instead, FTS's grid can be as coarse as the filter designer wants. The cell size does not impact the speed of the GHT step<sup>1</sup>. The cost of weighting the grid by the likelihood *does* depend on the grid resolution, but this cost is, in practice, negligible with respect to the GHT step.

---

<sup>1</sup> If one assume the software is running on an ideal Von Neumann machine with  $O(1)$  memory access.

Moreover, the grid representation is easier to handle than particle distributions. The problem with particles is that they tend to concentrate on small parts of the state space after few observations. If one uses too few particles to bootstrap the filter, it is likely that no particle will be near the true solution: in few steps the distribution will suffer from particle depletion. In actual implementations, this problem must be mitigated by either using a particularly relaxed likelihood, or by injecting new particles in the distribution.

### 7.5.2 On the Assumption of a Precise Incremental Estimate

The MCS-2 and FTS methods assume the availability of a ‘precise’ incremental estimate. We would like to make some remarks on the effect of the (inevitable) error on  $s$ .

Consider the approximation step in the FTS derivation: from Equation 7.18 to Equation 7.20, this term:

$$\int p(\mathbf{y}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{x}_0)d\mathbf{x}_k$$

is approximated by this term:

$$p(\mathbf{y}_k|\mathbf{x}_k \ominus \mathbf{s}_{0:k})$$

because  $p(\mathbf{x}_k|\mathbf{x}_0)$  is assumed to be a Dirac distribution according to the increment  $\hat{\mathbf{s}}_{0:k}$ .

If  $\hat{\mathbf{s}}_{0:k}$  is uncertain, then the translation operation should be followed by a ‘smoothing’ operation. Let  $\hat{\mathbf{s}}$  be distributed as a Gaussian with covariance  $\Sigma$ . In our informal notation, this would be

$$\int p(\mathbf{y}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{x}_0)d\mathbf{x}_k = p(\mathbf{y}_k|\mathbf{x}_k \ominus \hat{\mathbf{s}}_{0:k}) * \mathcal{N}(0, \Sigma)$$

We do not do this smoothing operation yet. There are multiple ways this could be integrated in the algorithm. The straightforward way would be, in the GHT step, to distribute votes according to the distribution – but this would be expensive.

A different way would be to adjust the size of the grid. In the first step, we are integrating  $\mathbf{y}_0$  with  $\hat{\mathbf{s}}_{0:0} = \mathbf{0}$ . Therefore, one could use a very fine-grained grid, as small as the sensor is precise. In later iterations, the grid for  $p(\mathbf{x}_0|\mathbf{y}_k)$  should be made coarser. For example, if  $\hat{\mathbf{s}}_{0:k}$  has a precision of 20cm, a grid cell should be around 20–40cm. Using a coarser grid will take uncertainty of  $\hat{\mathbf{s}}$  into account while not requiring more CPU.

### 7.5.3 On the Laziness of FTS

Both the PFs algorithm rely on a filtering stage which implies frequent integration of the observations. On the contrary, FTS can integrate observation “distant” in time, as long the scan-matcher is sufficiently precise.

Moreover, in FTS the scans can be processed in arbitrary order: the factorization in Equation 7.20 is, of course, commutative. While the typical case would be to integrate scans as they are available, there is much freedom here. One can skip scans, or procrastinate and postpone some, if there are not enough computational resources at the moment. For the experiments, we integrated one scan every 5m and simply discarded the others.

The integration order being arbitrary hints to the fact that FTS has the same “consideration” for every scan. This in contrast with any PF, where the initial observations are more important than the last, as the first scans essentially choose which part of the state space will be explored.

### 7.5.4 On the Probability Distribution

FTS, and in similar way MCS-1 and MCS-2, not only provide a loop closing constraint, but they compute a probability distribution of it. This distribution can be used in various way. A first way, which is more specific to the work of this thesis, is to have a stochastic relation between two poses. This kind of constraint is in the form

$$\delta_{ij} = x_j \ominus x_i + \omega \quad (7.25)$$

where  $\omega$  is a zero mean Gaussian noise with covariance matrix  $R$ . The parameter of this relation are approximated with the first two moments of the grid (or particle) distribution. It is worth to notice, that this works under the assumption that the posterior on the closure point is unimodal. In the case of multimodal posteriors, a clustering algorithm needs to be performed, in order to reduce the problem into several unimodal ones.

Another way of using this distribution is in a topological mapping problem. As pointed out in [Ranganathan, 2008], it is possible to use a Rao Blackwellized particle filter to compute a posterior over the possible map topologies. In such a framework, this distribution can be used as an informed proposal distribution for the particle filter.

## 7.6 Experiments

We chose two logs from the Radish [Howard & Roy, 2003] repository. The first was collected in an Intel building in Seattle. This is a building, of size 30m×30m, with office rooms, many of which are very similar. Some are cluttered with object/people, and there are also curved surfaces. In such environment, the typical situation is that there are many initial hypotheses that gets quickly disambiguated.

The second log was collected in the Aces building at the University of Texas in Austin. This is the typical ambiguous situation: in this 60m×60m environment, there is a symmetry – a central room, from which four similar corridors depart, with other feature-less corridors around. In this case, the typical situation is that a filter must keep a relatively small number of hypotheses (2 or 4) for a long time, until there is enough data to completely disambiguate the pose.

We used GMapping<sup>2</sup> to process the logs, and we obtained two sets of data: the final SLAM result, and an incremental scan-matching estimate. The SLAM result was used as the map input  $m$  to the algorithms, from which the PFs would create an occupancy grid, and FTS creates the normal map. The scan-matching result was split into multiple chunks, with each being an independent experiment. In each chunk, the robot traveled for about 30m. In total, we obtained 20 chunks for Aces and 40 for Intel. Each method was given the global map and one scan-matching chunk, and had to guess where the robot started.

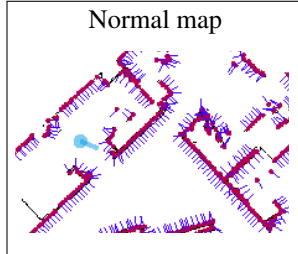
FTS's parameters were chosen as follows. The normal map has a resolution of one point every 0.2m – this results in about 4000 points in the aces environment. The belief grid has a resolution of 1m x 1m x 30deg. We integrate a scan every 5m, and discard the others.

As for the PFs, the occupancy grid has a resolution of 0.05m, the number of particles is 10000 (sufficient but not excessive for the environments considered. The scans are integrated every 0.5m or 0.5rad, whichever comes first.

---

<sup>2</sup><http://openslam.org/>

## EXAMPLE OF FROZEN TIME SMOOTHER BEHAVIOR



FTS uses a normal map as the environment representation (above).

FTS's state (A, C, E, F) is the belief on the first pose of the robot, at the 'freezing time' 0.

Thanks to the knowledge of the scan-matcher distribution, scans can be integrated very infrequently (10m apart in this example).

(A) – It is not uncommon for FTS to guess in one-shot: the true robot pose is indicated by a

marker, and the red square is the peak of the distribution.

(B) – The vanilla GHT algorithm computes  $p(\mathbf{x}_k|\mathbf{y}_k)$ . Our modified version computes  $p(\mathbf{x}_0|\mathbf{y}_k)$ , based on the scan-matcher result.

(C) – After integrating two scans, the pose is pretty much disambiguated. However, it takes another two integrations of scans (E, F) for the belief to go to zero in the other parts of the state space.

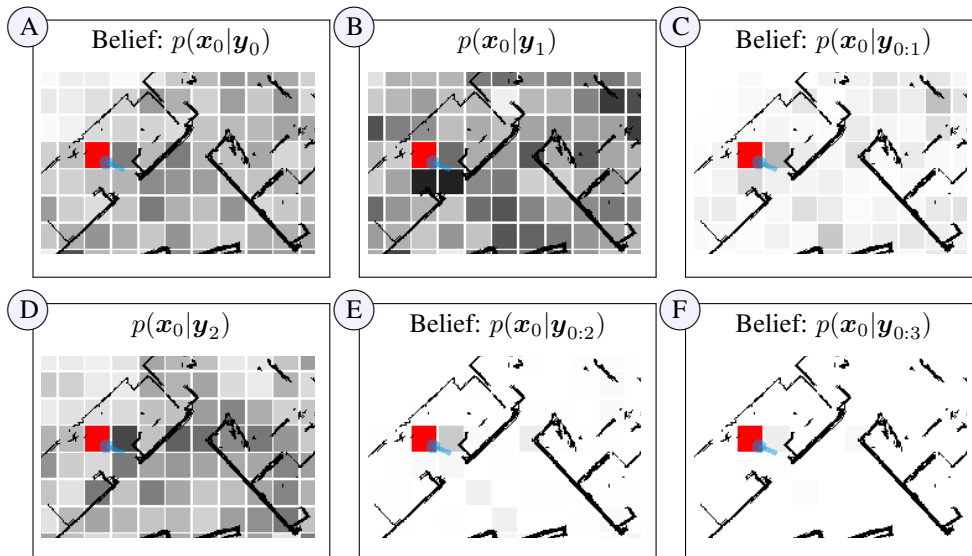
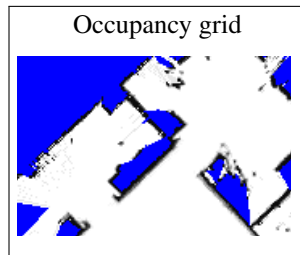


Figure 7.3: Example of Frozen-Time Smoother behavior.

## EXAMPLE OF APPROXIMATED MONTE CARLO SMOOTHING BEHAVIOR



Some execution steps of Monte Carlo Smoothing v1 and v2 are depicted. The map used is a conventional occupancy grid, where blue (dark gray) identifies the unknown region. Figures A-C depict MCS-1, while figures D-F depict MCS-2.

Both algorithms converged to the same solution, which is the exact one. However, despite the same initialization of the particles (figures A and D), MCS-1 is more confident than MCS-2, converging to a single particle.

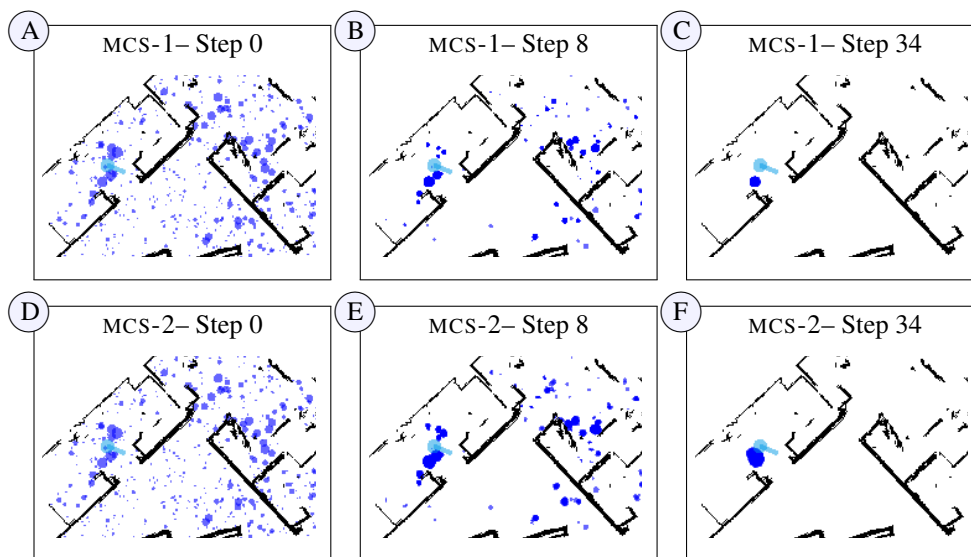


Figure 7.4: Example of approximated Monte Carlo smoothing behavior



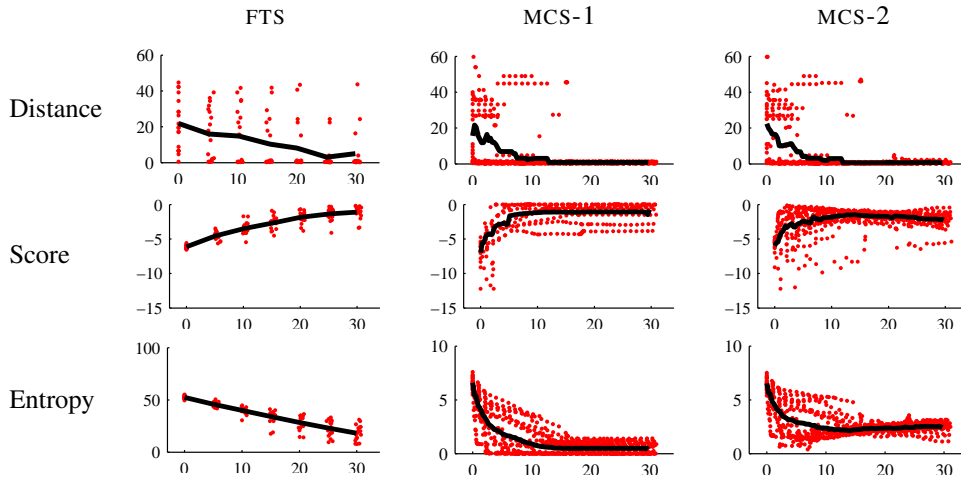


Figure 7.5: Results for the Aces environment. In all the plots, the  $x$  axis measures the linear distance along the chunk in meters; each chunk is about 30 meters long. The red dots are individual samples; the black bold line is the samples average.

The most natural performance measure is the **distance** between the maximum-likelihood pose estimated by the method, and the true pose. This is a quick measure that indicates whether the method has converged near the true solution. Note, however, that this distance measure could be misleading in ambiguous situations. For example, as can be seen in [Figure 7.5](#), there are three cases in which FTS cannot disambiguate the pose by considering only one scan every 5m, simply because there is not enough information. In those cases, the ‘true’ pose is actually the second highest peak of the distribution, but this cannot be seen by considering only the distance measure. Therefore other statistics are needed.

To compute the following two measures in a consistent way, we converted the PF distribution to a grid distribution with the same resolution as the grid used for FTS.

Another measure – less intuitive than the distance, but more correct – is the estimated likelihood for the true pose. That is, if each method estimates  $p(\mathbf{x}_0 = x | \mathbf{y}_., s.) = f(x)$ , we consider the **score**  $s = \log f(\bar{\mathbf{x}}_0)$ , where  $\bar{\mathbf{x}}_0$  is the true pose. It can be shown that the score is an approximation to the KLD distance between the belief and the true distribution. In [Figure 7.5](#), we see that the score is high, even in the cases for which the first peak is not the true pose. In [Figure 7.6](#), there is one case in which FTS’s distance is high and the score is low: this is a genuine failure, caused by a corresponding failure of the scan-matcher during the chunk (GMapping, employing a RBPF, can afford to have a non particularly robust scan matcher).

It is interesting to note that the score values are similar among the three methods: this reinforces our belief that they are computing the same distribution, albeit in completely different ways.

A measure that describes the character of the method is the **entropy** of the estimated distribution. The entropy for the PFs quickly goes to zero, as particles tend to concentrate in small areas of the state space. MCS-1 has a lower entropy than MCS-2, as the particles do not “move” within the state space. Instead, for FTS, the distribution is very smooth and

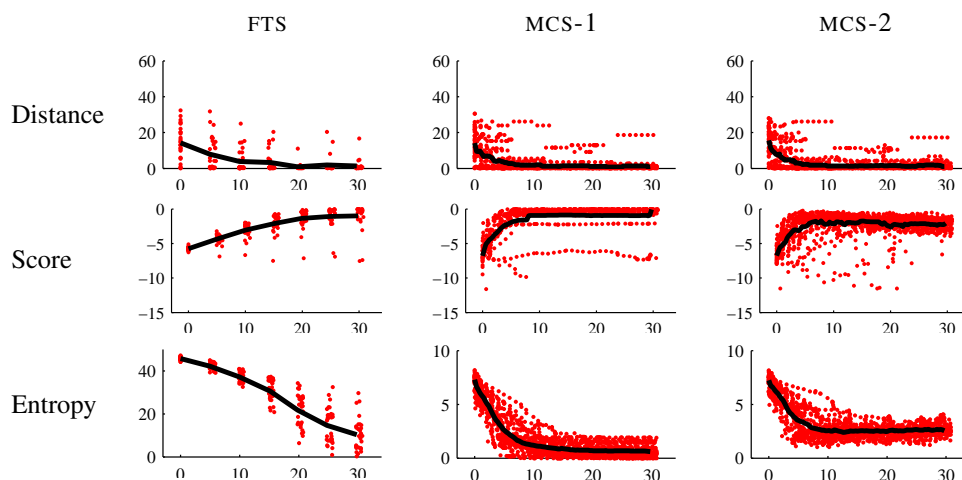


Figure 7.6: Results in the Intel environment. In all the plots, the  $x$  axis measures the linear distance along the chunk in meters; each chunk is about 30 meters long. The red dots are individual samples; the black bold line is the samples average.

non-zero practically everywhere. This gives a very high value of entropy. Note, however, that large values of entropy do not imply bad precision. In fact, as can be seen by comparing the distance and entropy graphs, for the first step, the entropy is high even when FTS did a one-shot localization.

Finally, we do some consideration on the efficiency. On an Intel Core 2 Duo, with 2.0GHz and 4Mb of cache, an iteration of FTS needed about 1.2s, while the PFs took about 2.1s. In these experiments, FTS was fed about one fifth of the data fed to the PFs. Therefore, we observed about an order of magnitude gain in efficiency. Note, however, that this kind of benchmark is highly dependent on the implementation and the parameters used. For the PFs, we used some old, well-honed source code with all the tricks we know for a particularly efficient and robust implementation. For FTS, we did the straightforward naive implementation of [Algorithm 11](#). We are particularly happy about these numbers, as FTS is already quite efficient, and there is a lot of space for improvement.

### 7.6.1 Connection with Previous Works

In Gutmann *et al.* [[Gutmann & Konolige, 1999](#)] loop closure is reduced to global Markov localization in the partially built map. This is, of course, extremely expensive, therefore the likelihood computation is approximated by correlation of the local map over the occupancy grid for the global map, and such correlation is efficiently implemented using special-purpose vectorization operations (MMX). After the robot is localized, the new constraint is added to the global constraints of the map.

In Fox *et al.* [[Fox \*et al.\*, 2003](#)], an important problem of loop closure is pointed out: the robot might be *outside* the map built so far. They extended Bayesian localization by using a different way of computing the likelihood, when the robot is in an unknown area. In this way, they reduced the number of false positive in the loop detection, which lead to enforcing

wrong loop closing constraints.

In Neira *et al.* [Neira, Tardos, & Castellanos, 2003], the map is represented using features (segments and corners). They handle loop closure explicitly by trying to match sets of features in the local map to sets of features in the global map. For this, they use random samples consensus, followed by a joint compatibility test. Clearly, if one has a feature map, the loop closure problem can be solved easily and efficiently.

Visual loop closure was attempted in Levin and Szeliski [Levin & Szeliski, 2004], by looking for correspondences between omnidirectional images. They used a distance matrix to express similarity between all images, the quality of which is improved by imposing epipolar constraints. They found out that off-diagonal elements are found when the same path is traversed again.

In grid-based Rao-Blackwellized particle filters (RBPF) [Hähnel *et al.*, 2003a; Grisetti, Stachniss, & Burgard, 2005], in theory one does not need to explicitly handle loop closure. Each particle represents an hypothesis on the trajectory: when the robot revisits parts of the old map, only the particles with a good trajectory survive. However, in practice, there are some complications. In RBPF the number of particles is limited by the available memory and CPU. This means that there are typically as few as 50-100 particles representing all the hypotheses on the trajectory. When a loop is closed, only very few of these particles will have a non-zero likelihood, and therefore there will be a loss of diversity (particle depletion). Stachniss *et al.* [Stachniss *et al.*, 2005] addressed the problem by actively detecting loops and then using techniques to restore particle diversity after closing the loop.

One of the problems to consider is that loop closure cannot be an instantaneous decision, because further data could disprove the identified closing point. Therefore, one should keep different hypotheses on the map topology. Haehnel [Haehnel *et al.*, 2003] extended FastSLAM with lazy data association;

In Ho and Newman [Newman, Cole, & Ho, 2006b] loop closure is completely decoupled from the incremental map estimation. They considered the environment as a collection of discrete scenes, for which a distance measure can be defined. The resulting scene-to-scene distance matrix is used to detect consistent *sequences* of scenes that indicate loop closure.

## 7.7 Conclusions

We presented an algorithm, called Frozen-Time Smoother, which can efficiently solve the global localization problem when it is formulated as a smoothing problem, and a precise incremental estimate of the robot motion is available. These assumptions hold when when global localization is used for loop closing in SLAM,

Using this estimate, the GHT can be slightly modified to compute the distribution of the pose at a certain “freezing time”, given a sensor reading collected much later in time.

We compared the FTS with the closest technique available in the literature. Under the same assumptions of the FTS, we modified the Monte Carlo filter to obtain an approximated solution to the smoothing problem. The experiments suggest that a naive implementation of FTS is more efficient than an extremely optimized state-of-the-art Monte Carlo filter working under the same assumptions.

Moreover, it has several other nice properties. The grid representation is efficient and does not have the common problems of particle methods, like overconfidence and the need

of a frequent update. Moreover, FTS has an intrinsic laziness, as it does not need frequent updates and it can process data in arbitrary order.

## Chapter 8

# Probabilistic Map Optimization

### 8.1 Introduction

Map optimization is the process of combining local constraints (provided by the incremental mapper) and global constraints (provided by the loop-closure algorithm) to obtain an overall consistent map. This problem is usually solved using the so-called maximum likelihood (ML) approaches. Instead of maintaining a posterior, the goal of these approaches is to calculate the maximum likelihood map based on the observations of the robot and its motions. In ML algorithms, the problem instances are typically described by a graph, whose nodes represent either robot poses or landmark locations. An edge between two nodes represents a relative measurement of them. Finding a maximum likelihood solution to this problem means to determine the assignment of poses to the nodes of the graph which provides the best explanation of the measurements. Traditional ML approaches assume the data associations as given and focus mainly on estimating the position of the nodes, not their uncertainty. However, finding potential data associations requires to estimate the marginal probability distribution over the nodes locations. Still, once the ML configuration of the nodes is known, the marginals can be computed by inverting the (sparse) information matrix of the system. Unfortunately, real world problem instances are often described by graphs having thousands of nodes. Inverting matrices of this size is computationally too expensive and other paths need to be followed.

In this chapter we model the map optimization problem in probabilistic terms. We will show that the SLAM graph defines a Gaussian Markov random field (GMRF) and optimization is just an inference problem over this field. This idea has originally been proposed by Ranganathan *et al.* [Ranganathan, Kaess, & Dellaert, 2007], who model the full SLAM problem and utilized belief propagation as inference algorithm. In our work, instead, we focus on the delayed state parametrization of SLAM, developing a unified parametrization for it. Moreover, we describe a novel algorithm for computing the marginal covariances. Our approach outperforms the technique introduced by Ranganathan *et al.* [Ranganathan, Kaess, & Dellaert, 2007] while keeping the same time complexity. Additionally, our algorithm is able to obtain estimates which are closer to the exact ones and generally more conservative. This is important when solving data association problems since over-confident covariances may result in losing valid associations.

## 8.2 Gaussian Markov Random Field

A general Markov random field (MRF) is a model for describing the joint probability distribution over set of random variables. More formally, a MRF represents the conditional independence between variables using an undirected graph  $G = (V, E)$ . Each vertex  $v \in V$  in the graph represents a random variable. An edge  $\{u, v\} \in E$  between two nodes represents a dependency between the random variables  $u$  and  $v$ . The edges and the vertexes of the graph are labeled with a set of potential functions  $\phi$  defined over a subset of  $V$ . Conditional independence is associated with *graph separation*. The Hammersley Clifford Theorem [Lauritzen, 1996] stated that the graph separation relation in the graph is exactly the conditional independence relation over  $V$ . In other words, suppose that  $x_a, x_b, x_c$  are a set of random variables and  $a, b, c \subset V$  the corresponding nodes in the graph. We have that  $x_a$  and  $x_b$  are independent conditioned on  $x_c$ , iff every path from a node in  $a$  to a node in  $b$  intercept a node in  $c$ .

GMRFs are a particular case of MRF suitable for describing multivariate Gaussian distributions. In this case, we just have singleton potential functions (describing a prior belief) and pairwise potential functions (describing the relationship between two different variables). The full joint distribution can be written as

$$p(x) = \frac{1}{Z} \prod_{i=1}^n \phi_i(x_i) \prod_{j=i+1}^n \phi_{i,j}(x_i, x_j). \quad (8.1)$$

Here  $\phi_i(x_i)$  represents the prior belief about the variable  $x_i$  and  $\phi_{i,j}(x_i, x_j)$  represents the stochastic constraint between the variables  $x_i$  and  $x_j$ .

If we consider the canonical parametrization of the Gaussian, we can express each pairwise potential as

$$\phi_{i,j}(\mathbf{x}_i, \mathbf{x}_j) = \exp \left\{ \mathbf{c} + \boldsymbol{\eta}_{ij}^T \mathbf{x}_{ij} - \frac{1}{2} \mathbf{x}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{x}_{ij} \right\}, \quad (8.2)$$

where

$$\mathbf{x}_{ij} \triangleq \begin{bmatrix} \mathbf{x}_i \\ \mathbf{x}_j \end{bmatrix} \quad (8.3)$$

$$\boldsymbol{\eta}_{ij} \triangleq \begin{bmatrix} \boldsymbol{\eta}_{ij}^i \\ \boldsymbol{\eta}_{ij}^j \end{bmatrix} \quad (8.4)$$

$$\boldsymbol{\Omega}_{ij} \triangleq \begin{bmatrix} \boldsymbol{\Omega}_{ij}^{[ii]} & \boldsymbol{\Omega}_{ij}^{[ij]} \\ \boldsymbol{\Omega}_{ij}^{[ji]} & \boldsymbol{\Omega}_{ij}^{[jj]} \end{bmatrix} \quad (8.5)$$

Here,  $\boldsymbol{\Omega}$  and  $\boldsymbol{\eta}$  are respectively the information matrix and the information vector of a measurement between two nodes. The singleton potentials represents the prior information about a node and are expressed as

$$\phi_i(\mathbf{x}_i) = \exp \left\{ \mathbf{c} + \boldsymbol{\eta}_i^T \mathbf{x}_i - \frac{1}{2} \mathbf{x}_i^T \boldsymbol{\Omega}_i \mathbf{x}_i \right\}, \quad (8.6)$$

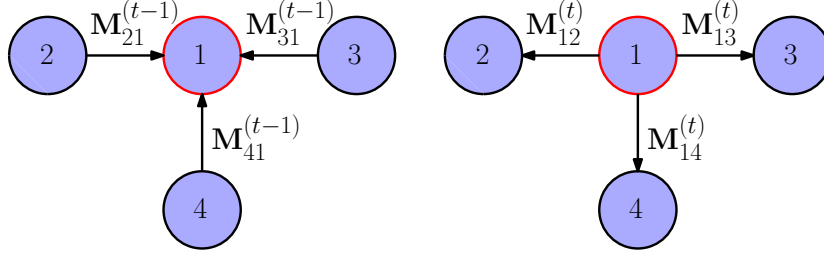


Figure 8.1: Message flow of BP on a simple graph at time  $t$  for node 1. In the left image node 1 computes its marginal from the messages sent by its neighbors. In the right image, node 1 calculates the new messages and send them . The messages are computed according to Equation 8.9 and Equation 8.10.

### 8.2.1 Inference

An important inference problem for a graphical model is computing the marginals  $p_i(x_i)$ , obtained by integrating  $p(x)$  over all variables except  $x_i$ , for each node  $i$ . This problem can be solved very efficiently in graphs that are trees by a form of variable elimination, known as belief propagation, which also provides an approximate method for general graphs.

The goal of belief propagation is to compute the marginal distribution over a graphical model by means of local message passing. This algorithm has been introduced by Pearl [Pearl & Russel, 2000] for inference on Bayesian Network. If the graphical model does not contain loops, this local passing scheme is guaranteed to give the exact solution for the marginals. Loopy belief propagation is an approximated algorithm, which uses the same equation of belief Propagation, but in a graph with cycles. As discussed by Weiss and Freeman [Weiss & Freeman, 2001] and Malioutov *et al.* [Malioutov, Johnson, & Willsky, 2006], loopy belief propagation on general graphs computes correct marginal means and generally incorrect covariances.

Belief propagation works by iteratively computing local messages and beliefs for every node in the graph, starting with constant messages. The propagation of the messages is repeated until a fixed point is reached. It can be shown that in the context of trees only two iterations are needed: from leaves to root and vice versa. Using the superscript  $(t)$  for denoting the current iteration, the belief parameters (denoted by  $\mathbf{m}^{(t)}$  and  $\mathbf{M}^{(t)}$ ) are given by

$$\mathbf{m}_i^{(t)} = \boldsymbol{\eta}_i + \sum_{j \in \mathcal{N}_i} \mathbf{m}_{ji}^{(t-1)} \quad (8.7)$$

$$\mathbf{M}_i^{(t)} = \boldsymbol{\Omega}_i + \sum_{j \in \mathcal{N}_i} \mathbf{M}_{ji}^{(t-1)}, \quad (8.8)$$

where  $\boldsymbol{\eta}_i$  and  $\boldsymbol{\Omega}_i$  are the parameters of the prior belief (the singleton potential functions),  $\mathcal{N}_i$  is the neighboring set of node  $i$ , and the messages from node  $i$  to node  $j$  are defined as

$$\mathbf{m}_{ij}^{(t)} = \boldsymbol{\eta}_{ij}^j - \boldsymbol{\Omega}_{ij}^{[jj]} \left( \boldsymbol{\Omega}_{ij}^{[ii]} + \mathbf{M}_i^{(t)} - \mathbf{M}_{ji}^{(t-1)} \right)^{-1} \left( \boldsymbol{\eta}_{ij}^i + \mathbf{m}_i^{(t)} - \mathbf{m}_{ji}^{(t-1)} \right) \quad (8.9)$$

$$\mathbf{M}_{ij}^{(t)} = \boldsymbol{\Omega}_{ij}^{[jj]} - \boldsymbol{\Omega}_{ij}^{[ji]} \left( \boldsymbol{\Omega}_{ij}^{[ii]} + \mathbf{M}_i^{(t)} - \mathbf{M}_{ji}^{(t-1)} \right)^{-1} \boldsymbol{\Omega}_{ij}^{[ij]} \quad (8.10)$$

using the definition in [Equation 8.4](#) and [Equation 8.5](#). See [Figure 8.1](#) for an example of message flow.

### Loopy Belief Propagation

When loopy belief propagation is applied, the marginal covariances can be either overconfident or conservative. However, Weiss and Freeman [[Weiss & Freeman, 2001](#)] showed that they are always overconfident for GMRFs with pairwise cliques, which is the case of SLAM. These estimates often result in a poor approximation of the true marginals, which cannot be used for data association, since this results in valid associations being rejected.

A more information theoretic analysis can be derived by considering how the marginal beliefs of every node are computed. With respect to [Equation 8.8](#), the marginal belief are computed by summing up all the incoming messages from the neighboring nodes. If we consider every message as an observation of the node from its neighbor, we end up with the measurement update of the Information Filter. However, this integration is correct only if the two observations are independent, which is not the case of graphs with loops.

### Belief Propagation over a Spanning Tree

A different approach is to approximate the full graph model by its spanning tree. Since the tree is obtained by eliminating edges (and therefore constraints) from the GMRF and inference on the tree is exact, it is possible to obtain conservative estimate of the true marginal covariances. However, the result of this approximation strongly depends on the property of the tree used. The best results are obtained when using a minimal spanning tree. Moreover, inference on the spanning tree do not consider the loopy structure at all, resulting in too conservative estimates.

## 8.3 SLAM as a Gaussian Markov Random Field

In this section we describe how the SLAM problem can be expressed in the Gaussian random Markov field (GMRF) framework. This formulation has been first introduced by Ranganathan *et al.* [[Ranganathan, Kaess, & Dellaert, 2007](#)], where they considered the full SLAM problem (also known as Smoothing and Mapping).

### 8.3.1 FullSLAM Graphical Model

In the formulation of [[Ranganathan, Kaess, & Dellaert, 2007](#)], they model the distribution over the whole robot trajectory,  $X \triangleq \{x_i : i = 0, \dots, M\}$ , and the landmark locations,  $L \triangleq \{l_j : j = 1, \dots, N\}$ , given the landmark measurements,  $Z \triangleq \{z_k : k = 1, \dots, K\}$ , and the odometry  $U \triangleq \{u_i : i = 1, \dots, M\}$ . The posterior can be factorized as

$$P(X, L|Z, U) \propto P(Z, U|X, L)P(X, L) \quad (8.11)$$

$$\propto P(x_0) \prod_{i=1}^M P(x_i|x_{i-1}, u_i) \prod_{k=1}^K P(z_k|x_{i_k}, l_{j_k}) \quad (8.12)$$



where  $P(x_i|x_{i-1}, u_i)$  represents the motion model and  $P(z_k|x_{i_k}, l_{j_k})$  is the measurement model, assuming known correspondences  $(i_k, j_k)$ . The motion model is given as  $x_i = x_{i-1} \oplus u_i + \omega_i$ , where  $\omega_i$  is a zero-mean Gaussian noise with covariance matrix  $Q_i$ ,  $\omega_i \sim \mathcal{N}(0, Q_i)$ . Similarly, the measurement model is  $z_k = l_{j_k} \ominus x_{i_k} + v_k$ , where  $v_k$  is a normally distributed zero-mean noise with covariance matrix  $R_k$ ,  $v_k \sim \mathcal{N}(0, R_k)$ .

Considering the canonical parametrization of the Gaussian, we can express each pairwise potential of the GMRF as either a motion or a measurement model. In the case of an odometry link, we have

$$\phi_{i,j}(\mathbf{x}_i, \mathbf{x}_j) = \exp \left\{ \mathbf{c} + \boldsymbol{\eta}_{ij}^T \mathbf{x}_{ij} - \frac{1}{2} \mathbf{x}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{x}_{ij} \right\} \quad (8.13)$$

$$\mathbf{x}_{ij} \triangleq \begin{bmatrix} \mathbf{x}_i \\ \mathbf{x}_j \end{bmatrix} \quad (8.14)$$

$$\boldsymbol{\eta}_{ij} \triangleq \begin{bmatrix} \boldsymbol{\eta}_{ij}^i \\ \boldsymbol{\eta}_{ij}^j \end{bmatrix} = \begin{bmatrix} \mathbf{F}_{ij} \\ -\mathbf{I} \end{bmatrix} \mathbf{Q}^{-1} \mathbf{a}_{ij} \quad (8.15)$$

$$\boldsymbol{\Omega}_{ij} \triangleq \begin{bmatrix} \boldsymbol{\Omega}_{ij}^{[ii]} & \boldsymbol{\Omega}_{ij}^{[ij]} \\ \boldsymbol{\Omega}_{ij}^{[ji]} & \boldsymbol{\Omega}_{ij}^{[jj]} \end{bmatrix} = \begin{bmatrix} \mathbf{F}_{ij} \\ -\mathbf{I} \end{bmatrix} \mathbf{Q}^{-1} \begin{bmatrix} \mathbf{F}_{ij} \\ -\mathbf{I} \end{bmatrix}^T. \quad (8.16)$$

where  $F_{ij}$  is the Jacobian of the function  $\oplus$  w.r.t.  $x_i$  and  $\mathbf{a}_{ij} = x_j - x_i \oplus u_j$ .

As for the measurement link they are

$$\phi_{i,j}(\mathbf{x}_i, \mathbf{l}_j) = \exp \left\{ \mathbf{c} + \boldsymbol{\eta}_{ij}^T \mathbf{x}_{ij} - \frac{1}{2} \mathbf{x}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{x}_{ij} \right\} \quad (8.17)$$

$$\mathbf{x}_{ij} \triangleq \begin{bmatrix} \mathbf{x}_i \\ \mathbf{l}_j \end{bmatrix} \quad (8.18)$$

$$\boldsymbol{\eta}_{ij} \triangleq \begin{bmatrix} \boldsymbol{\eta}_{ij}^i \\ \boldsymbol{\eta}_{ij}^j \end{bmatrix} = \begin{bmatrix} \mathbf{H}_{ij} \\ \mathbf{J}_{ij} \end{bmatrix} \mathbf{R}^{-1} \mathbf{c}_{ij} \quad (8.19)$$

$$\boldsymbol{\Omega}_{ij} \triangleq \begin{bmatrix} \boldsymbol{\Omega}_{ij}^{[ii]} & \boldsymbol{\Omega}_{ij}^{[ij]} \\ \boldsymbol{\Omega}_{ij}^{[ji]} & \boldsymbol{\Omega}_{ij}^{[jj]} \end{bmatrix} = \begin{bmatrix} \mathbf{H}_{ij} \\ \mathbf{J}_{ij} \end{bmatrix} \mathbf{R}^{-1} \begin{bmatrix} \mathbf{H}_{ij} \\ \mathbf{J}_{ij} \end{bmatrix}^T. \quad (8.20)$$

again,  $H_{ij}$  and  $J_{ij}$  are respectively the Jacobians of the function  $\ominus$  w.r.t.  $x_i$  and  $l_j$ , and  $\mathbf{c}_{ij} = z_k - l_{j_k} \ominus x_{i_k}$ .

### 8.3.2 Delayed State SLAM Graphical Model

An alternative formulation of the SLAM problem is to use a delayed-state representation rather than a feature based one [Eustice, Singh, & Leonard, 2005]. Delayed-state representations do not explicitly model features in the environment. Instead, the state vector is composed only by a sequence of poses. In this representation, raw data are registered to provide virtual observations of pose displacement. These virtual observations arise, for instance, by matching pairwise laser range data or camera images.

When using this representation, the information matrix of the corresponding multivariate Gaussian is exactly sparse, as it has been pointed out by Eustice *et al.* [Eustice, Singh, & Leonard, 2005]. This is the natural representation to express constraints between robot poses, being them either local (from a scan matcher) or global (from loop closing). Considering

a pair of poses,  $i$  and  $j$ , their displacement can be expressed by the following non linear stochastic function

$$\delta_{ij} = \mathbf{x}_j \ominus \mathbf{x}_i + \boldsymbol{\omega}. \quad (8.21)$$

Here  $\delta_{ij}$  is the virtual observation made from the pose  $i$  about the pose  $j$ ,  $\ominus$  is the standard motion composition operator and  $\boldsymbol{\omega}$  is a zero-mean Gaussian variable with covariance matrix  $\mathbf{R}$ , representing the uncertainty of the measurement. In the following, we consider the linearized version, being  $\mathbf{H}_{ij}$  and  $\mathbf{J}_{ij}$  respectively the Jacobians of the function  $\ominus$  w.r.t.  $x_i$  and  $x_j$ .

Translating a SLAM problem formulated according to the delayed-state framework into a GMRF is quite straightforward. Since the structure of the graph between the two models is preserved, all we need is to define the nature of the potential functions, such that the resulting joint probability distribution is unchanged.

If we consider the canonical parametrization of the Gaussian, we can express each pairwise potential as

$$\phi_{i,j}(\mathbf{x}_i, \mathbf{x}_j) = \exp \left\{ \mathbf{c} + \boldsymbol{\eta}_{ij}^T \mathbf{x}_{ij} - \frac{1}{2} \mathbf{x}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{x}_{ij} \right\}, \quad (8.22)$$

where

$$\mathbf{x}_{ij} \triangleq \begin{bmatrix} \mathbf{x}_i \\ \mathbf{x}_j \end{bmatrix} \quad (8.23)$$

$$\boldsymbol{\eta}_{ij} \triangleq \begin{bmatrix} \boldsymbol{\eta}_{ij}^i \\ \boldsymbol{\eta}_{ij}^j \end{bmatrix} = \begin{bmatrix} \mathbf{H}_{ij}^T \\ \mathbf{J}_{ij}^T \end{bmatrix} \mathbf{R}^{-1} \mathbf{e}_{ij} \quad (8.24)$$

$$\boldsymbol{\Omega}_{ij} \triangleq \begin{bmatrix} \boldsymbol{\Omega}_{ij}^{[ii]} & \boldsymbol{\Omega}_{ij}^{[ij]} \\ \boldsymbol{\Omega}_{ij}^{[ji]} & \boldsymbol{\Omega}_{ij}^{[jj]} \end{bmatrix} = \begin{bmatrix} \mathbf{H}_{ij}^T \\ \mathbf{J}_{ij}^T \end{bmatrix} \mathbf{R}^{-1} \begin{bmatrix} \mathbf{H}_{ij}^T \\ \mathbf{J}_{ij}^T \end{bmatrix}^T. \quad (8.25)$$

$$\mathbf{e}_{ij} = \delta_{ij} - \mathbf{x}_j \ominus \mathbf{x}_i + \mathbf{H}_{ij} \mathbf{x}_i + \mathbf{J}_{ij} \mathbf{x}_j \quad (8.26)$$

Here,  $\boldsymbol{\Omega}$  and  $\boldsymbol{\eta}$  are respectively the information matrix and the information vector of a measurement between two nodes. The singleton potentials are generally set to the unity except for the first pose, which is fixed at the origin.

## 8.4 Approximate Covariance Computation

As stated in [Subsection 8.2.1](#), loopy belief propagation on general graphs computes correct marginal means and generally incorrect covariances. In the case of SLAM, it has been proved that the algorithm provides overconfident estimates, which cause data association to fail.

In this section, we introduce an approach for computing the marginal covariances on a loopy graph. Our algorithm is able to obtain conservative estimates while considering the loopy structure of the problem. We show that our approach is able to fuse information arising from loops by means of exact inference on a spanning tree. Our results are supported by an extensive set of experiments.

Before describing our algorithm, we introduce an information fusion framework for dealing with unknown correlations between different estimates. This framework has been introduced by Uhlman and Julier under the name of Covariance Intersection [[Julier & Uhlmann, 1997](#)].

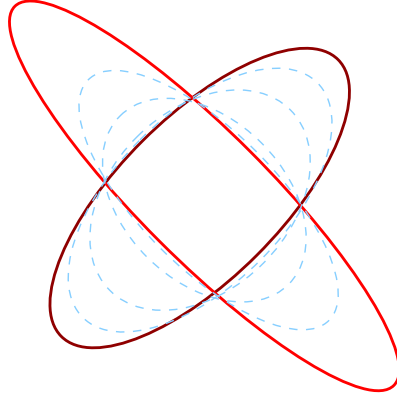


Figure 8.2: The shape of the CI update. The thick outer ellipses represent the covariances of **A** and **B**. The dashed ellipses represent resulting covariances of **C** by using different values of  $\omega$ .

### 8.4.1 Covariance Intersection

Covariance Intersection is a fusion rule for combining two different estimates when the correlations between them are unknown. Suppose we have two consistent estimates  $\langle \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1 \rangle$  and  $\langle \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2 \rangle$  for the same Gaussian random variable, expressed in terms of mean and covariance matrix. Furthermore suppose the cross correlation between the two covariance matrices  $\boldsymbol{\Sigma}_1$  and  $\boldsymbol{\Sigma}_2$  to be unknown.

Covariance Intersection combines the two estimates, in order to obtain a new one  $\langle \hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}} \rangle$ , according to the following equations

$$\hat{\boldsymbol{\Sigma}} = (\omega \boldsymbol{\Sigma}_1^{-1} + (1 - \omega) \boldsymbol{\Sigma}_2^{-1})^{-1} \quad (8.27)$$

$$\hat{\boldsymbol{\mu}} = \hat{\boldsymbol{\Sigma}} (\omega \boldsymbol{\Sigma}_1^{-1} \boldsymbol{\mu}_1 + (1 - \omega) \boldsymbol{\Sigma}_2^{-1} \boldsymbol{\mu}_2). \quad (8.28)$$

If  $\omega \in [0, 1]$ , the resulting estimate has been proved to be consistent. Moreover, it can be shown that the approach is optimal in the case in which the cross correlations are unknown [Julier & Uhlmann, 1997]. As pointed out by Julier and Uhlmann [Julier & Uhlmann, 1997], the intuition behind this update rule comes from its geometric interpretation. Let us consider the plot in Figure 8.2. This figure shows the covariance ellipses for the two estimates (thick outer ellipses) and the resulting update with different values for  $\omega$  (dashed inner ellipses). The optimal estimate, the one considering the cross correlations, is known to lie within the intersection region of the estimate ellipses, of **A** and **B**. As can be seen from the figure, the CI ellipses always circumscribe this region, but do not lie within it. This results in having a covariance matrix *bigger* (in a matrix sense) than the exact one, for any value of the cross correlation terms. Moreover, the CI ellipses intersect the corners of the intersection region, showing no other robust estimate can achieve a tighter result

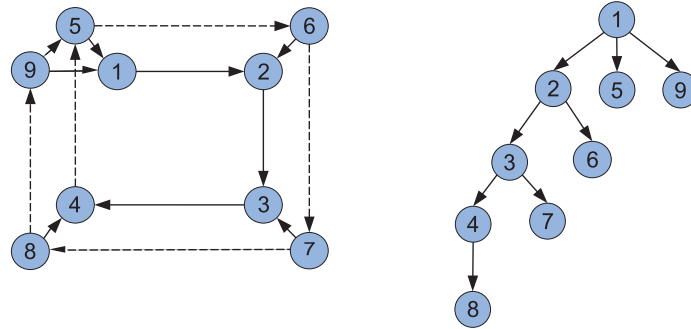


Figure 8.3: An example loopy graph (left) and its spanning tree (right). The off-tree edges are the dashed edges in the graph.

### 8.4.2 Loopy Intersection Propagation

A natural solution to obtain better covariances is to combine the conservativeness of inference on a spanning tree with the information coming from the off-tree edges. In this section, we will describe how this can be achieved using the Covariance Intersection framework. **Figure 8.3** shows an example graph. A spanning tree of this graph is depicted on the right part of the image. The off-tree edges (the ones which are not present in the spanning tree) are depicted as dashed edges in the graph.

Our goal is to obtain a tree approximation from the original graph. This tree takes into account the off-tree information. This can be expressed analytically by considering some operations on the joint Information Matrix of the GMRF.

Let  $\Omega$  be this joint Information Matrix and  $\hat{\Omega}$  its tree approximation. For any tree-structured Information Matrix there exist a symmetric matrix  $\mathbf{K}$  such that

$$\hat{\Omega} = \Omega - \mathbf{K}. \quad (8.29)$$

The matrix  $\mathbf{K}$  acts to remove edges from the graph, therefore it is referred to as *cutting matrix*. Some elements of the cutting matrix, as the off-diagonal elements corresponding to eliminated edges, are uniquely defined by the choice of the tree. However, other entries, as the block diagonal elements, are not constrained. We will focus our attention on a restricted class of cutting matrices, called *regular* cutting matrices. For a regular cutting matrix  $\mathbf{K}$  corresponding to an embedded tree, all off-diagonal entries not corresponding to cut edges must be zero. The block diagonal entries for nodes from which no edge is cut must be zero. The off-diagonal entries corresponding to cut edges must be equal to the original matrix ones.

In order to derive a tree approximation of the GMRF, we have to analyze the structure of the corresponding cutting matrix. For the sake of simplicity, we will restrict the analysis on cutting a single edge, being the extension to multiple edges straightforward.<sup>1</sup>

When using the naive spanning tree approximation, we can express the matrix cutting an

<sup>1</sup>The overall cutting matrix can be decomposed into the sum of single edge ones, while preserving its regularity condition.

edge between the node  $i$  and the node  $j$  as

$$\mathbf{K}_{ij} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Omega}_{ij}^{[ii]} & \mathbf{0} & \boldsymbol{\Omega}_{ij}^{[ij]} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Omega}_{ij}^{[ji]} & \mathbf{0} & \boldsymbol{\Omega}_{ij}^{[jj]} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}. \quad (8.30)$$

Cutting this edge results in subtracting the information of an edge from the overall information matrix of the system.

To consider the off-tree information while keeping the regularity condition of the cutting matrix, we are forced to modify its block diagonal entries. Let  $\mathbf{P}_{ij}^{[i]}$  and  $\mathbf{P}_{ij}^{[j]}$  be the information about node  $i$  and  $j$  arising from the off-tree edge. The modified cutting matrix will be of the following form

$$\mathbf{K}_{ij} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Omega}_{ij}^{[ii]} - \mathbf{P}_{ij}^{[i]} & \mathbf{0} & \boldsymbol{\Omega}_{ij}^{[ij]} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Omega}_{ij}^{[ji]} & \mathbf{0} & \boldsymbol{\Omega}_{ij}^{[jj]} - \mathbf{P}_{ij}^{[j]} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}. \quad (8.31)$$

Note that subtracting information from the cutting matrix results in adding this information to the tree approximation  $\hat{\boldsymbol{\Omega}}$  according to [Equation 8.29](#).

In the a graphical model perspective, those two pieces of information,  $\mathbf{P}_{ij}^{[i]}$  and  $\mathbf{P}_{ij}^{[j]}$ , can be seen as a prior knowledge about the nodes, acting as singleton potential functions. In other words, we approximate the GMRF with its spanning tree by considering the off-edge information as *prior* knowledge in this approximation.

### 8.4.3 Algorithm

Our approach transforms the graph into a tree augmented with prior information. First it computes a spanning tree on the GMRF. Second, it performs BP on the computed tree, to obtain open loop estimates for the information matrices  $\{\mathbf{M}_i\}$  of all nodes. Third, for each edge  $\langle i, j \rangle$  in the graph which does not appear in the tree it computes the priors  $\mathbf{P}_{ij}^{[i]}$  and  $\mathbf{P}_{ij}^{[j]}$  for the nodes  $i$  and  $j$ . These priors aim to recover part of the information which has been lost when removing the edge.  $\mathbf{P}_{ij}^{[i]}$  and  $\mathbf{P}_{ij}^{[j]}$  are computed considering the mutual information introduced by the cut edge based on the estimates  $\mathbf{M}_i$  and  $\mathbf{M}_j$  computed by the first application of BP. Let  $\mathbf{E}_{ij}^{[i]}$  and  $\mathbf{E}_{ij}^{[j]}$  be these estimates. They can be computed as follows.

$$\begin{aligned} \mathbf{E}_{ij}^{[i]} &= \boldsymbol{\Omega}_{ij}^{[ii]} - \boldsymbol{\Omega}_{ij}^{[ij]} (\mathbf{M}_j + \boldsymbol{\Omega}_{ij}^{[jj]})^{-1} \boldsymbol{\Omega}_{ij}^{[ji]} \\ \mathbf{E}_{ij}^{[j]} &= \boldsymbol{\Omega}_{ij}^{[jj]} - \boldsymbol{\Omega}_{ij}^{[ji]} (\mathbf{M}_i + \boldsymbol{\Omega}_{ij}^{[ii]})^{-1} \boldsymbol{\Omega}_{ij}^{[ij]}. \end{aligned} \quad (8.32)$$

Intuitively,  $\mathbf{E}_{ij}^{[i]}$  is obtained by propagating the edge information from the BP estimate  $\mathbf{M}_j$  of the node  $j$  along the cut edge.  $\mathbf{E}_{ij}^{[j]}$  is computed in a symmetric way.

For each node in a cut edge, we have therefore two estimates:  $(\mathbf{E}_{ij}^{[i]}, \mathbf{M}_i)$  and  $(\mathbf{E}_{ij}^{[j]}, \mathbf{M}_j)$ . We can compute improved estimates by applying covariance intersection, for each off-tree edge, as:

$$\begin{aligned}\hat{\mathbf{M}}_i &= \omega_i \mathbf{M}_i + (1 - \omega_i) \mathbf{E}_{ij}^{[i]} \\ \hat{\mathbf{M}}_j &= \omega_j \mathbf{M}_j + (1 - \omega_j) \mathbf{E}_{ij}^{[j]}.\end{aligned}\quad (8.33)$$

In our implementation, we choose  $\omega_i$  and  $\omega_j$  so that the determinants of  $\hat{\mathbf{M}}_i$  and  $\hat{\mathbf{M}}_j$  are minimal. In other words, we select the smaller covariance which can be obtained by CI.

We compute the *priors*  $\mathbf{P}_{ij}^{[k]}$  as the difference between the improved estimate and the BP one as

$$\mathbf{P}_{ij}^{[k]} = \hat{\mathbf{M}}_k - \mathbf{M}_k. \quad (8.34)$$

Here the  $\mathbf{P}_{ij}^{[k]}$  represent the desired priors coming from the suppressed edge  $\langle i, j \rangle$ .

The final step consists in performing a final inference using BP on the spanning tree in which we injected these terms. It is worth noticing, that our algorithm has the best performance on the incremental spanning tree defined by Grisetti *et al.* [Grisetti *et al.*, 2007a] because the off-edge information is used immediately. Moreover, this tree is as easily maintained as any minimum one.

Note that the prior for a node  $i$  is computed by considering the contribution of all the cutted edges connected to node  $i$ . Whereas suppressing a single edge leads to a conservative estimate, suppressing multiple edges may lead to overconfident estimates. As shown in [Subsection 8.5.1](#), the level of overconfidence increases with the connectivity of the network.

The data flow of the resulting algorithm, called Loopy Intersection Propagation, is explained in [Algorithm 13](#), while

---

**Algorithm 13:** Loopy Intersection Propagation

---

**Input:** The SLAM GMRF:  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  and  $\Psi = \{\psi_i(), \psi_{ij}() | i, j \in \mathcal{V}\}$

**Output:** The approximated marginal covariances  $\hat{\mathbf{M}}_i, i \in \mathcal{V}$

- 1 Compute the spanning tree of the graph  $G_{\mathcal{T}} = (\mathcal{V}, \mathcal{E}_{\mathcal{T}})$ ;
  - 2 Perform Belief Propagation on the spanning tree;
  - 3 **forall** off-tree edges  $e \in \mathcal{E} - \mathcal{E}_{\mathcal{T}}$  **do**
  - 4     | Compute the prior beliefs according to [Equation 8.34](#);
  - 5 **end**
  - 6 Update the tree with the prior beliefs;
  - 7 Perform Belief Propagation on the resulting model;
- 

## 8.5 Experiments

In this section we evaluate the performance of loopy belief propagation (LBP), for the mean, and loopy intersection propagation (LIP), for the covariance, within the probabilistic map

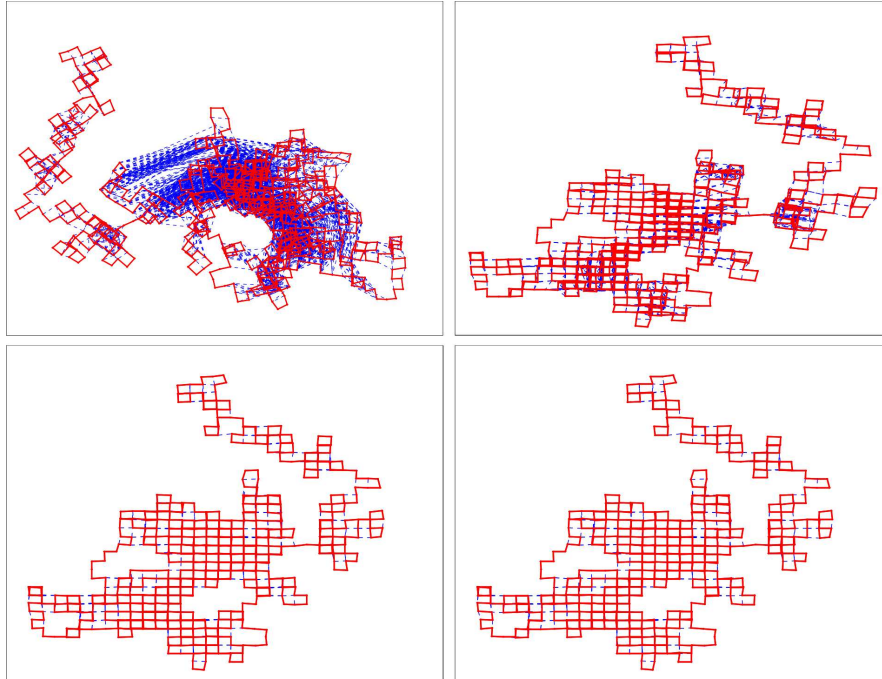


Figure 8.4: Iterations of LBP on a random generated graph with 1000 nodes. The top left figure shows the perturbed graph.

optimization framework. Since LBP has already been proved to converge in the GMRF, in this section we focus on demonstrate that the use of LBP is equivalent to a non-linear optimization algorithm. As for LIP, we will compare the estimated covariance matrices with respect to standard belief propagation and loopy belief propagation. All those algorithms have a complexity linear in the number of edges of the graph. Therefore, we are only interested in measuring the quality of the approximation. Given a node, we want to compare the approximate marginal covariance  $\hat{\Sigma}$  with the exact one  $\Sigma$ <sup>2</sup>. This can be done by considering the norm of the matrix difference

$$\|\hat{\Sigma} - \Sigma\|_F, \quad (8.35)$$

where  $\|\cdot\|_F$  is the Frobenius norm.

Furthermore, given a conservative and an overconfident estimate, we prefer the conservative one, since it allows to better deal with data association. An estimate is conservative if it is *bigger* than the exact one, thus being  $\hat{\Sigma} - \Sigma \geq 0$ . Measuring the conservativeness means to measure how far the matrix  $\hat{\Sigma} - \Sigma$  is to be positive definite. This can be done by considering the value of the smallest eigenvalue: it is negative in the case of overconfidence and positive in the case of conservativeness.

We performed experiments on real world datasets and on simulated ones. Using simulated data we obtained quantitative results both for the mean and the covariance computation. With

<sup>2</sup>The exact covariances are computed by inverting the information matrix of the GMRF

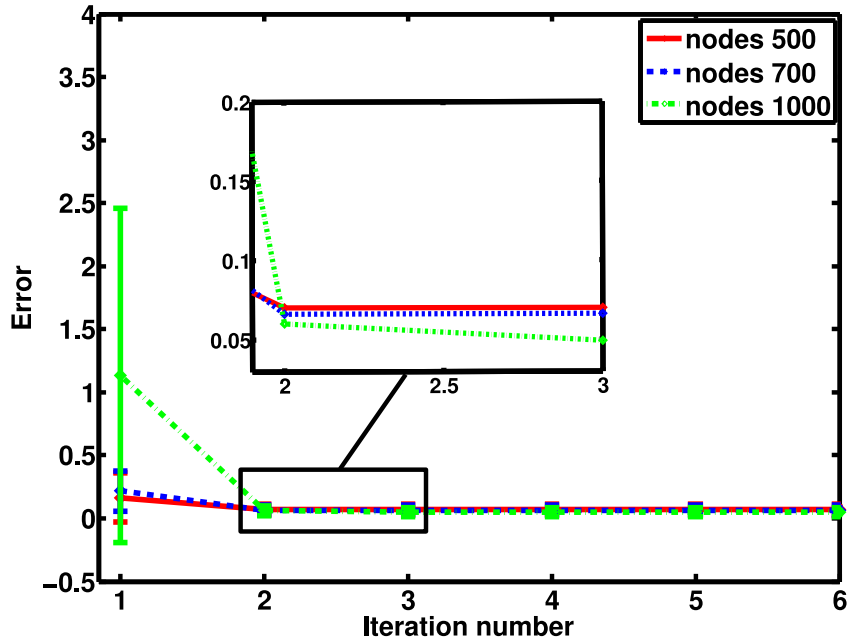


Figure 8.5: Average and standard deviation of the relative error between nodes in random generated graphs of 500, 700 and 1000 nodes, with precision 1000. The plot shows the variation of this error during several iterations

real dataset, we only obtain qualitative results for the mean, as no groundtruth is available, while keeping the qualitative ones for the covariance.

### 8.5.1 Statistical Experiments

We performed statistical experiments on simulated networks of different sizes. The networks were randomly generated by simulating a random walk in the  $\mathcal{SO}(3)$  space. Loops were simulated by considering the Euclidean distance between nodes.

The mean computation was compared to the ground truth with respect to both the number of nodes and the system noise. As for the number of nodes, the experiments were performed using networks with size from 50 to 1000, adding 50 nodes each time. As for the noise level, we considered spherical noise with precision values from 50, to 1000, with a step of 50. For both experiments, we iterated the linearization and LBP step until convergences. We then measured the average relative error between each pair of nodes of each iteration. The plots in [Figure 8.5](#) and [Figure 8.6](#) shows the average value and 95% confidence intervals for some of those parameters values.

[Figure 8.4](#) shows the behavior of LBP with respect to the graph with 1000 nodes. The picture on the top left shows the perturbed graph, while the remaining ones show the corrected



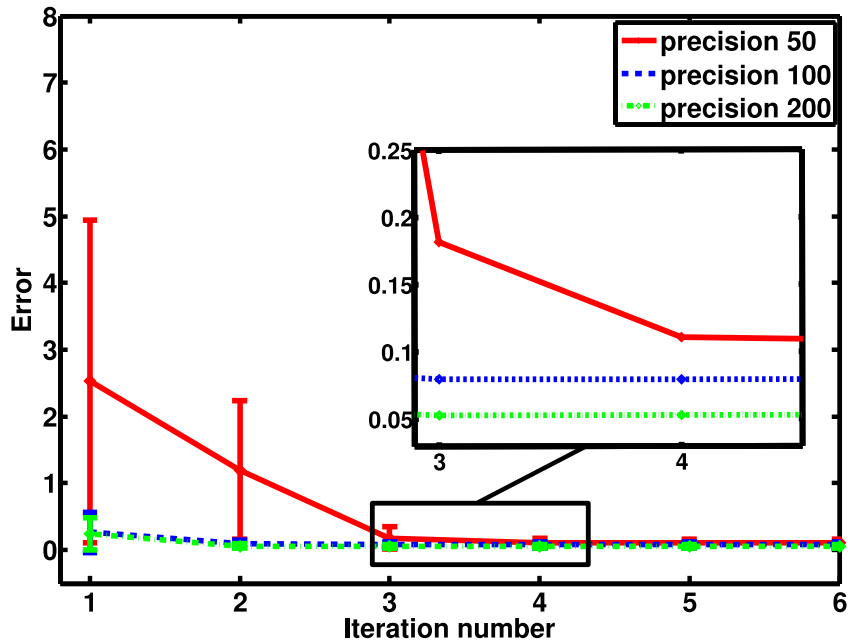


Figure 8.6: Average and standard deviation of the relative error between nodes in random generated graphs of 250 nodes, with precision 50, 100 and 200. The plot shows the variation of this error during several iterations

graph after 1, 2 and 3 iterations. The red links represent the robot path (local constraints), while the blue ones represent data association (global constraints).

We then compared LIP with LBP and BP on a spanning tree. We used networks with 500, 1,000, 3,000, and 5,000 nodes. The plots in [Figure 8.7](#) and [Figure 8.8](#) show the average values and the 95% confidence intervals.

[Figure 8.7](#) shows the approximation error of the three approaches, computed according to [Equation 8.35](#). As can be seen, our approach scales better than LBP and BP. This is due to the increasing number of loops occurring in the network as its size grows: Whereas BP does not use loop information, our approach does consider it in a better way than LBP.

As for the overconfidence, [Figure 8.8](#) shows the evolution of the minimum eigenvalue of the error matrix. BP always produces conservative estimates, while LBP produces overly overconfident ones. Our approach lies in the middle with a small level of overconfidence.

### 8.5.2 Real World Data

We analyzed the behavior of our algorithm on graphs extracted from standard datasets available on radish [[Howard & Roy, 2003](#)]. We use LBP to compute the mean value of the distribution and we compared LIP with LBP and BP on a minimum spanning tree for the covariance.

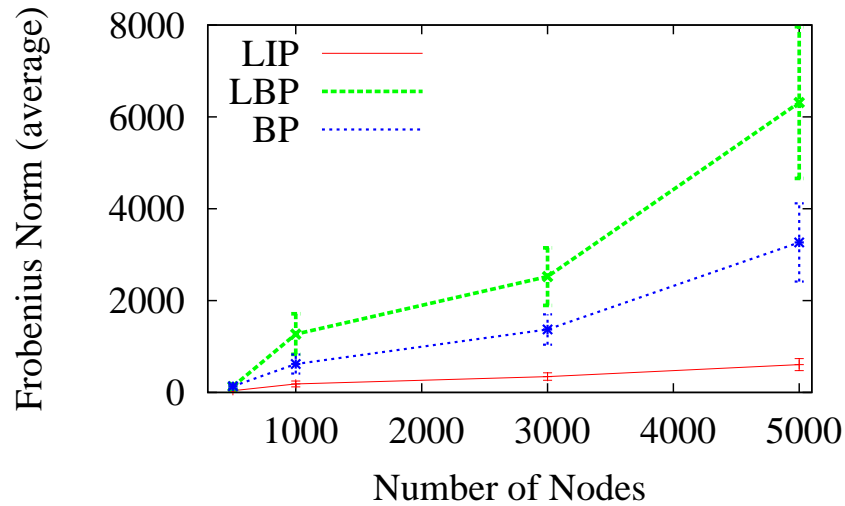


Figure 8.7: Average and standard deviation of the Frobenius norm of the node estimates of a randomly generated network. This measures the quality of the approximation of the three approaches as a function of the size of the network.

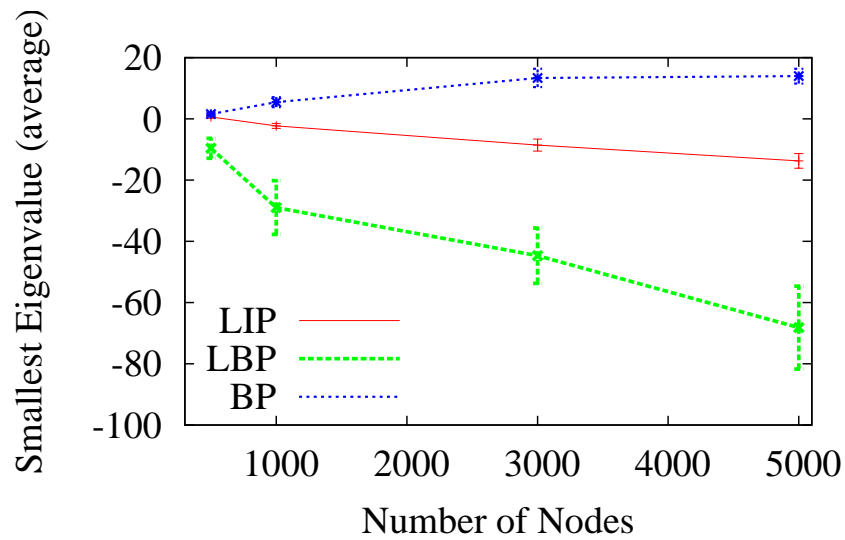


Figure 8.8: Average and standard deviation of the minimum eigenvalue of the difference matrix of nodes estimates of a randomly generated network. This measures the conservativeness of the estimate.

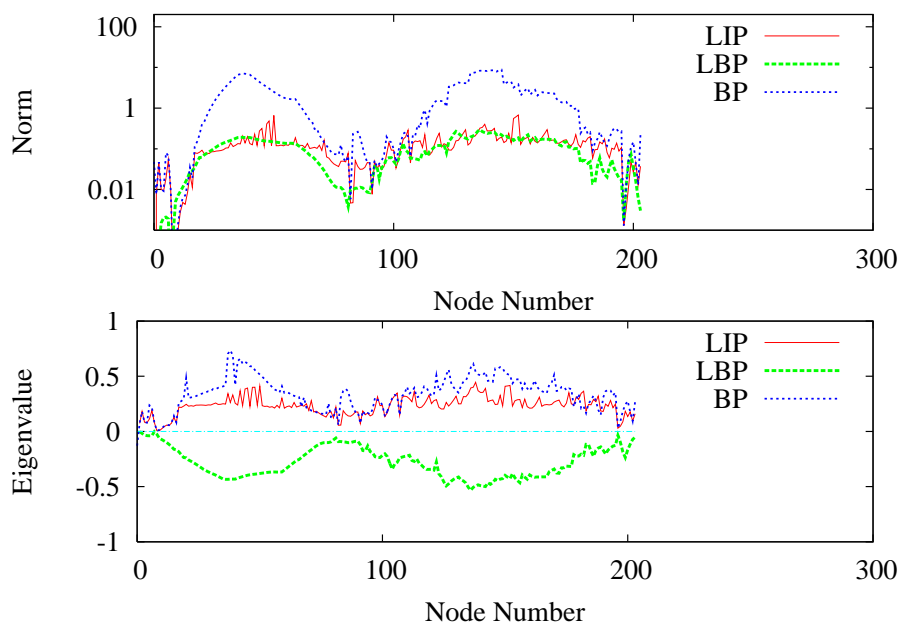


Figure 8.9: Analysis of the covariance approximation on the Intel dataset. The upper plot shows the approximation error and the lower one shows the conservativeness analysis of the different nodes. Note that LBP is overconfident, BP on a spanning tree is overly conservative, and our approach (LIP) provides an intermediate result, being in general closer to the exact estimate.

For each node of the network we measured the distance between the estimate and the exact value of the covariance, according to Equation 8.35. In all the cases our approach provided a better estimate than BP on the spanning tree. Furthermore it provided more conservative estimates than LBP. Quantitative results for the Intel dataset are depicted in Figure 8.9. Qualitative results for the Intel and Aces datasets are shown in Figure 8.10 and Figure 8.11.

## 8.6 Connections with Previous Work

The work described in this chapter belongs to the family of ML algorithms. One of the first approaches of this type has been proposed by Lu and Milios [Lu & Milios, 1997a]. Later, Howard *et al.* [Howard, Matarić, & Sukhatme, 2001b] used Gauss-Seidel relaxation to localize the robot and build a map. Duckett *et al.* [Duckett, Marsland, & Shapiro, 2002] proposed Gauss-Seidel relaxation to minimize the error in the network of constraints. Their approach has been subsequently extended by Frese *et al.* [Frese, Larsson, & Duckett, 2005] by the introduction of the multi-level relaxation (MLR) framework, which applies relaxation on different resolutions.

Olson *et al.* [Olson, Leonard, & Teller, 2006] addressed the problem by using gradi-

ent descent on a network described in a form which allows for efficient analytical updates. Graphical SLAM [Folkesson & Christensen, 2004] builds a graphical model of the smoothing problem. It optimizes the graph by defining an energy function for each node and then minimizing this energy.

Whereas these methods mainly focus on estimating the most likely configuration of the map, they leave open how to estimate the uncertainty of the solution. To the best of our knowledge, the only approach which computes both the ML configuration of the nodes and their marginal distribution has been proposed by Ranganathan *et al.* [Ranganathan, Kaess, & Dellaert, 2007]. They model the smoothing problem as a Gaussian Markov random field (GMRF) and use loopy belief propagation on this model. An exact algorithm for those covariances can be found in the work of Kaess *et al.* [Kaess, Ranganathan, & Dellaert, 2007]. Their algorithm, however, is guaranteed to be efficient only in the case of band-diagonal matrices, and can be more expensive for general sparsity patterns. Moreover, LIP equations can be integrated in the LBP loop for the mean computation.

## 8.7 Conclusions

In this chapter we presented a probabilistic framework for map optimization. We showed how the constraints arising from incremental mapping (local constraints) and loop closing (global constraints) can be fused together in a unified model. The model we choose is the Gaussian Markov Field. We showed that this model naturally represent a delayed state formulation of SLAM and that inference on it can be effectively used as a non linear optimization routine.

Moreover, we presented a novel algorithm for marginal covariance computation in the graph representation of SLAM. Our approach has been validated by an extensive set of experiments. In general, the estimated covariances are conservative and when they are overconfident, their overconfidence level is close to 0. Furthermore, our approach provides estimates closer to the exact ones with respect to other techniques of the same family like loopy belief propagation or belief propagation on a spanning tree.

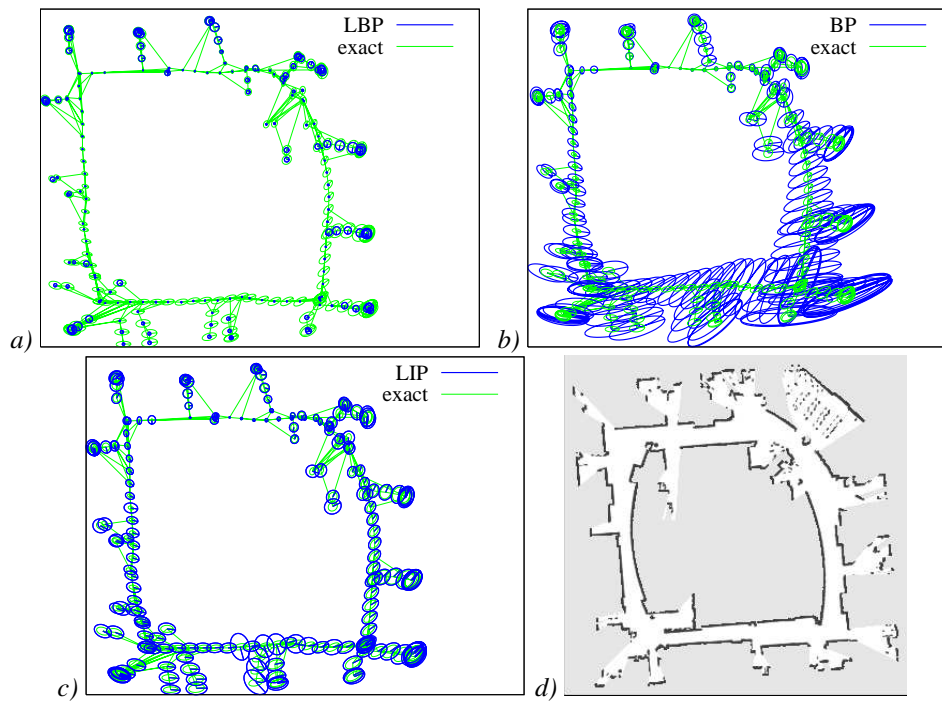


Figure 8.10: Mean and marginal covariances comparison on the Intel Lab dataset: the exact covariances, computed by inverting the full matrix, are depicted in red (solid line), the approximated ones are in green (dashed line). Results obtained by *a)* Loopy Belief Propagation. *b)* Belief Propagation on a Minimum Spanning Tree. *c)* Loopy Intersection Propagation. *d)* The grid map of the environment. Notice how the distribution mean reflects the environment topology.

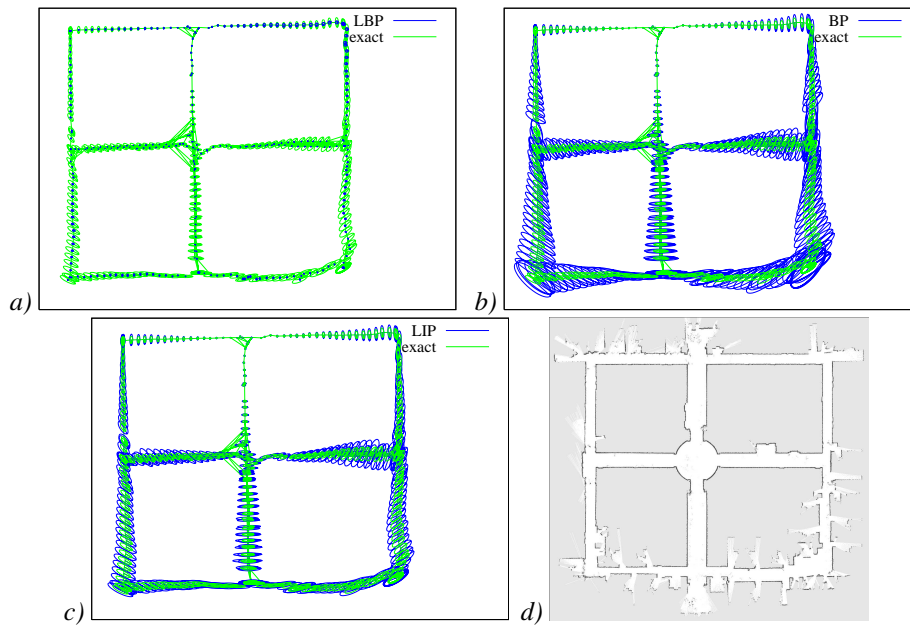


Figure 8.11: Mean and marginal covariances comparison on the Aces dataset: the exact covariances, computed by inverting the full matrix, are depicted in red (solid line), the approximated ones are in green (dashed line). Results obtained by *a)* Loopy Belief Propagation. *b)* Belief Propagation on a Minimum Spanning Tree. *c)* Loopy Intersection Propagation. *d)* The grid map of the environment. Notice how the distribution mean reflects the environment topology.

# Chapter 9

## Discussion

### 9.1 Conclusions

A key problem in mobile robotics is the knowledge of the environment and the robot position within it. The capability of learning maps is a precondition for every truly autonomous mobile vehicle. This task, is strictly connected with the task of localizing the robot within the environment. The connection lies in the fact that for a good map estimate it is needed the position of the robot and viceversa. To this end, the problem of robot localization and mapping in unknown environment, is one of the most studied problems in mobile robots research.

Up to now, SLAM has been addressed by two main solutions: filtering and optimization. However, not always the structure of the problem is taken into account. Optimization methods take some numerical algorithm “off the shelf” and apply it directly to the problem. Filtering approaches often use a Bayesian filter as a “black box”. Lately, some ideas about decoupling the problem into smaller ones are emerging. However, those ideas rely on heuristics, based on empirical considerations.

This thesis focuses on the development and improvement of Simultaneous Localization and Mapping algorithms. In particular, we are interested in reliable and effective techniques for unstructured large scale environments. The purpose of our research is to devise contextual models of the SLAM process, that take into account different situations.

We began the work by showing a exhaustive taxonomy of the diverse solutions for the SLAM problem. We described how the environment can be described, showing the pros and cons of the different representations: landmark, grid, topological and hybrid maps. We then explain the diverse class of algorithms. We started describing techniques about incremental mapping, such as scan matching and visual odometry, followed by global optimization techniques. We then explained the different filtering algorithms for SLAM: the Kalman filter, the information filter and the particle filter. Finally, some techniques for topological and hybrid mapping were described.

In [Chapter 5](#) we presented an introspection analysis that allows efficient optimizations for Rao-Blackwellized SLAM on grid maps. We are able to update the complex posterior with substantially less resources by performing the computations only for a set of representatives instead of for all potential hypotheses. We proposed an alternative way for representing

a distribution over grid maps which needs only a fraction of the memory resources used by previous approaches. Moreover, we proposed an efficient way for updating such a representation. The key idea is based on an analysis of the mapping process which allows us to perform filter updates *conditioned* to the state of the mapping system: **localization, mapping or loop closing**. Using this insight, we are able to obtain clusters of particles that share a compact map representation as well as an informed proposal distribution to sample the next generation of particles. With our optimizations, we are able to maintain between one and two orders of magnitude more samples and at the same time require less memory and computational resources compared to other state-of-the-art Rao-Blackwellized mapping techniques.

We then extended our introspective analysis from a filtering perspective to the whole problem. This latter analysis brought us to a decomposition of the SLAM process into three main aspects

- Incremental Mapping
- Loop Closure
- Map Optimization

In **Part II** we address those three main problems. The main idea of this decomposition is that we have two main source of information in the mapping process. At first, we have observations about the robot incremental motion. This information is obtained usually with odometry sensors and corrected with techniques such as scan matching. We call this kind of information *local constraints*. The reason of the name is easily explained: they link together observations that are close in time and generate maps that are locally consistent. A second source of information comes from the data association. This kind of information is obtained mainly by using statistical gating and exteroceptive sensors. We call this kind of information *global constraints*, and they link together information far away in time, but that refers to the same spatial position. The term global is used as those constraints are needed to ensure a global consistent map.

The incremental mapping problem has been addressed in **Chapter 6** We described how to obtain an incremental estimation of the robot motion both in static and dynamic environments. As for static environment, we revise ICP, a well known technique for scan matching. We showed how this technique can be used to solve the scan matching, a non linear optimization problem. We then showed how to compute a linear approximation of the optimization algorithm, describing state of the art techniques. As for dynamic environments, we introduced CRF-Clustering, a novel technique for clustering dependent data into homogeneous partitions and showed its capability to detect and predict the motion of moving objects from range data. Previous techniques were based on observability and visibility heuristics, and on keeping track of the consistency of the free space. While they have nice results on typical situations, they dramatically fall when their assumptions are violated. Moreover, we showed that typical clustering algorithm such as K-means fail in clustering very noisy and correlated data, as it does not consider their spatial dependency. On the other hand, our technique explicitly reasons about the underlying motion of the objects and their spatial properties, thus being more effective for the problem.

The problem of detecting and estimating globally consistent constraints has been addressed in **Chapter 7**. This aspect is the most sensitive one of the SLAM process. It is not always possible to rely on metrical information for the gating, as the robot can be in gross error. We think that some ad-hoc techniques for this problem should be used and we presented



an algorithm, called Frozen-Time Smoother, which can efficiently solve the loop closing problem when it is formulated as a smoothing problem, and a precise incremental estimate of the robot motion is available. Using this estimate, the generalized Hough transform can be slightly modified to compute the distribution of the pose at a certain “freezing time”, given a sensor reading collected much later in time. We compared the FTS with the closest technique available in the literature. Under the same assumptions of the FTS, we modified the Monte Carlo filter to obtain an approximated solution to the smoothing problem. The experiments suggest that a naive implementation of FTS is more efficient than an extremely optimized state-of-the-art Monte Carlo filter working under the same assumptions.

Once the local constraints from the incremental estimate and the global ones from the loop closing algorithm have been provided, they have to be fused together in order to obtain a consistent map estimate. This is explained in [Chapter 8](#), where we describe a probabilistic framework for map optimization. We showed how the constraints arising from incremental mapping (local constraints) and loop closing (global constraints) can be fused together in a unified model. The model we choose is the Gaussian Markov Field. We showed that this model naturally represent a delayed state formulation of SLAM and that inference on it can be effectively used as a non linear optimization routine. Moreover, we presented a novel algorithm for marginal covariance computation in the graph representation of SLAM. Our approach provides estimates, closer to the exact ones, with respect to other techniques of the same family like loopy belief propagation or classic belief propagation on a spanning tree.

All the algorithms presented in this thesis have been implemented and extensively tested, both on simulated and real data.

## 9.2 Future Works

Despite the encouraging results presented in this thesis, there are different aspects that could be improved. The main issues are pointed out in the following subsections.

### 9.2.1 3D Environments

The current state of the art in SLAM has shown that is possible to build accurate maps of large scale planar environments. However the problem of mapping 3D environments is mostly open. There are several difficulties that have to be considered when moving from a planar environment to a 3D space. First, the robot pose space grows from  $\mathcal{SO}(2)$  to  $\mathcal{SO}(3)$ , thus passing from three to six variables. Second, an increased amount of memory is required for storing a 3D map a rather than a 2D map. Finally, a huge amount of data has to be processed every time.

As future work, we would like to extend the results of this thesis about the problem decomposition to a full three dimensional scenario. In order to do so, we need to develop fast algorithms for scan matching and data association, as well as compact map representations. We think that both the delayed state representation and the “shared” map described in [Chapter 5](#) can be an efficient data structure for this purposes. Finally, we want to explore good linearization and inference algorithms for the probabilistic map optimization framework, adapted for the 3D case.

### 9.2.2 Multi Hypothesis Tracker for Data Association

The ideas presented in [Part II](#) do not keep track of different topological hypothesis of the map. The current system works on the best hypothesis given by the loop closing algorithm, acting like it was the only possibility. This is also the common view of Kalman filtering techniques, where only the maximum likelihood hypothesis is chosen with the data association. On the other hand, works on topological mapping showed that is possible to track different topologies of the environment, obtaining an a posteriori estimate of the most correct one.

As a future work, we would like to extend the introspective decomposition with a multi hypothesis tracker on the space of topologies. This can be done, by simply consider the different data association options from the loop closing algorithm. Some ideas on how to estimate heterogeneous features have been presented in our previous work [[Tipaldi et al., 2007](#)] and summarized in [Appendix B](#).

### 9.2.3 Simultaneous Localization, Mapping and Tracking

The motion clustering framework described in [Chapter 6](#) was mainly developed in order to classify points according to their source of motion. In this way, we were able to discriminate between points arising from the environment and points arising from the moving object. In current applications, however, there is an incoming need to also track this moving objects over time. The problem of Simultaneous Localization and Mapping with Moving Object Tracking (SLAMMOT) has been introduced in [[Wang et al., 2007](#)], where the authors showed that the two problem are independent conditioned on the classification of the points.

As a future work, we would like to incorporate a tracking algorithm within our SLAM framework. Once the different tracks have been detected and tracked, we can learn motion patterns from them, and use this information to classify the different motion behaviors. We think that this is an interesting problem, especially with unmanned vehicle, so that they can predict the motion of the others vehicle in the environment and react accordingly.

## **Part III**

# **Appendix**



## Appendix A

# Conditional Random Field for Semi-Supervised Clustering

In this appendix we will explain how Conditional Random Fields can be used as an effective tool for semi-supervised clustering. The model can easily incorporate any kind of constraints (e.g. pairwise, group). The constraints are soft, whose confidence level can be learned (via Maximum Likelihood) or given by an expert. The algorithm is an iterative one and pretty straightforward. It can deal with generalized constraints in both the point and parameter space.

### A.1 Semi-Supervised Clustering

Semi-supervised clustering is the problem of partitioning the data points into a specified number of clusters, where the supervision is in the form of pairwise constraints. Typically, those constraints appear in the form of *must-link* (points should be in the same cluster) and *cannot-link* (points should be in different clusters). This kind of constraints is relatively easy to obtain instead of having the class label, and results in improved performance when the separation boundary of different clusters is not well defined.

Clustering problems can be categorized as generative or discriminative. In generative clustering approaches, data come from a parametric model and the goal is to find the parameters that maximize the likelihood of the data given the model. Discriminative approaches, on the contrary, try to cluster the data in order to maximize within-cluster similarity and minimize between-cluster similarity, based on a particular similarity metric.

Moreover, Semi-supervision can be applied in two main forms. Some researchers use the supervision to guide the algorithm towards a partition that does not violate the constraints [Wagsta *et al.*, 2001; Basu, Banerjee, & Mooney, 2002]. Others use the supervision in order to modify the distance function between points. This new distance considers points in a cannot-link (must-link) relation farther (closer) than points not affected by a constraint [Bilenko & Mooney, 2003; Klein, Kamvar, & Manning, 2002; Xing *et al.*, 2003].

A unified probabilistic model that combines together *constraint-based* and *distance-based* approach is presented in [Basu *et al.*, 2006]. The approach is based on Hidden Markov Random Fields (HMRF). The clustering objective function is derived from the joint probability density of the observed points, the cluster parameters and the hidden labels. The approach is based on a generative approach to clustering, where points are *generated* by the model and clustering is performed by maximizing the likelihood of the data given the model.

In the next section, we provide a novel unified framework for semi-supervised learning, which is based on a discriminative model, called Conditional Random Field (CRF). We will explain why a discriminative approach is better suited for this kind of problems and will show how is it possible to express more general constraints within our model.

## A.2 CRF-Clustering

CRF-Clustering is a semi-supervised clustering algorithm. The approach is an unified framework where constraints and distances are combined together in a single probabilistic model. Moreover, the approach lies in between discriminative and generative clustering. More specifically, data points are not generated by some parametric model neither we use a fixed metric for discriminate them.

Our approach is partly generative and partly discriminative. It is *discriminative*, in the sense used for classification problems: cluster assignments are modeled within a Conditional Random Field, whose parameters are learned from labeled data. It is *generative*, as some feature functions of the CRF model are cluster-dependant (they return a vector whose dimension is the number of clusters) and parametric (the parameters depend on the cluster).

### A.2.1 The Model

In this section we provide a more formal description of the problem and its solution. The clustering setting considered is similar to the partitional prototype based one. Data points are partitioned into a pre-specified number of cluster, where each cluster is identified by a set a parameters. Those parameters are used in conjunction with a distortion function to provide a distance measure to be minimized.

More formally, our clustering considers a set of  $n$  data points  $X = (x_1, \dots, x_n)$ , each  $x_i \in \mathcal{D}$  being a general point in the space  $\mathcal{D}$ . The space is equipped with a distortion function  $d_\theta$  used to compute a distance between a point and a cluster:  $d_\theta : \mathcal{D} \rightarrow \mathbb{R}^+$  where  $\theta$  are the clusters' parameters. Supervision is provided in terms of pairwise constraints between data points. In this section we will describe how to model two kind of constraints, typical of current semi-supervision, while in the next one we will show how to model more general ones. The constraints considered here are *must-link* constraints  $C_{ML} = \{(x_i, x_j)\}$  and *cannot-link* constraints  $C_{CL} = \{(x_i, x_j)\}$ . A constraint  $(x_i, x_j) \in C_{ML}$  implies that  $x_i$  and  $x_j$  are labeled to belong to the same cluster, while  $(x_i, x_j) \in C_{CL}$  implies the two points are labeled to belong to different clusters. Those constraints are soft, meaning that each of them is associated with a violation cost  $\omega_{ij}$  representing the strength of the constraint itself.

As for the probabilistic model, we have the following random variables:

- observations,  $X = \{x_1, \dots, x_n\}$ , representing the data points;
- hidden labels,  $Y = \{y_1, \dots, y_n\}$ , representing the labels of the data points;

- hidden parameters,  $\Theta = \{\theta_1, \dots, \theta_k\}$ , representing the parameters of the distortion functions.
- constraints,  $C = C_{ML} \cup C_{CL}$ , defining the graph structure of the CRF model.
- a graph  $\mathcal{G} = (V, E)$  encoding the stochastic dependences among the variables. Cliques on this graph represent groups of correlated variables.

The graphical model is built in the following way. Each random variable  $y_i \in Y$  is associated with a set of neighbours  $\mathcal{N}_i$ . The neighbours are defined as the labels of the points involved in a constraint,  $\mathcal{N}_i^C = \{y_j | (x_i, y_j) \in C\}$ , the corresponding data point,  $x_i$ , and the cluster parameters  $\Theta$ , resulting in

$$\mathcal{N}_i = \mathcal{N}_i^C \cup \{x_i\} \cup \Theta \quad (\text{A.1})$$

The resulting random field over the labels and the parameters is a Conditional Random Field (CRF). According to the Hammersley-Clifford theorem, we have that the the joint distribution is a Gibbs distribution of the form:

$$P(Y, \Theta | X, C) = \frac{1}{Z} \exp \left\{ - \left( \sum_{(y_i, y_j) \in \mathcal{N}^C} \omega_{ij} \phi_C(y_i, y_j) + \sum_{i=1}^n \sum_{\theta_k \in \Theta} \omega_k d_{\theta_k}(x_i) \right) \right\} \quad (\text{A.2})$$

where  $Z$  is the partition function (normalizer),  $\omega_{ij}$  is the violation cost of the constraint  $(x_i, x_j)$ ,  $\phi_C(\cdot, \cdot)$  is the *constraint* potential function and  $d_{\theta_k}(\cdot, \cdot)$  the distortion function of cluster  $k$ . Note that  $\omega_k$  is a scaling parameter for the cluster function. It is usually set to 1 and shared among the clusters. However, it can also be learned from data and used to force the data points to belong to a specified cluster (this is another form of supervision). The  $1/2$  normalizer avoids the double counting of the symmetric constraints.

The clustering solution is provided by maximizing the distribution in [Equation A.2](#). Here lies the first difference with a generative approach to clustering. In a generative approach, the solution is provided by maximizing the data likelihood, which is

$$P(X | \Theta) = \mathbb{E}_Y [P(X, Y | \Theta)] \quad (\text{A.3})$$

However, it has been shown [[Basu et al., 2006](#)] that when dealing with constraints is better to maximize the full joint posterior

$$P(X, Y, \Theta | C) = P(\Theta) P(Y | \Theta | C) P(X | Y, \Theta) \quad (\text{A.4})$$

which is equivalent to jointly maximize the data likelihood and the probability of labels that respects the constraints, while regularizing the model parameters.

The contribution of modelling the conditional distribution, instead of the joint one, is twofold. On a first viewpoint, we can see that the conditional distribution is proportional to the joint one, by using the Bayes rule

$$P(Y, \Theta | C, X) = \frac{1}{P(X | C)} \cdot P(X, Y, \Theta | C) \propto P(X, Y, \Theta | C) \quad (\text{A.5})$$

In this case, maximizing the conditional distribution is the same as maximizing the joint one. Moreover, generative clustering assumes that data points are generated independently. While

this assumption is often reasonable, we think that is not the case when dealing with constraints. The role of constraints is to put two or more points in the same cluster, thus correlating them. Using a CRF, the independence assumption is not made, thus resulting in a better modelling tool.

Secondly, modelling the problem with the distribution in Equation A.2, we are allowed to express more general constraints than simple must or cannot link. How to model this type of constraints is explained in the following section

### A.2.2 Generalized Constraints

In this section we show how to model more general constraints than simple must and cannot link ones. In current semi-supervised clustering approaches, the label distribution is modeled independently of the data points. This distribution is intended to express the probability of a given label configuration and the constraints can only be expressed in terms of points  $i$  and  $j$  must or cannot be in the same cluster. When jointly modelling labels and parameters and conditioning that distribution on the data points, we have all the possible informations available in the same model and we can express constraints in terms of points that satisfy this property must or cannot be in the same cluster. We call these kind of constraints *generalized* constraints and we divide them in two main categories<sup>1</sup>:

- *parameter-based* generalized constraints;
- *point-based* generalized constraints.

**Parameter-Based Generalized Constraints** These constraints are related to the cluster parameters and involve knowledge in the parameters configuration (e.g. points with similar parameters must be in the same cluster). These constraints are expressed in terms of functions  $\phi_{\Theta} : \Theta \times \Theta \times Y \times Y \rightarrow \mathbb{R}^+$ . These functions are defined over the graph clique composed of the two labels' nodes and the parameters' ones ( $N^{\Theta}$ ). This is the most complex constraint, due to the strong dependency among labels and parameters (to learn the parameter we need the labels and viceversa) and their different nature (discrete vs. continue)

**Point-Based Generalized Constraints** These constraints are related to the data points and involve knowledge in some points' subspace (e.g. points which are similar w.r.t some distance function must be in the same cluster). These constraints are expressed in terms of functions  $\phi_X : X \times X \times Y \times Y \rightarrow \mathbb{R}^+$ . These functions are defined over the graph clique composed of the two labels' nodes and the corresponding data points' ones ( $N^X$ ). It is worth to notice, here, that while the function is defined over a 4D domain, it is, in reality, only a function over the two labels. The data points are given and related to the label index, so they can be treated as constants.

---

<sup>1</sup>notice that these are not the only type of generalized constraints. We can, for instance, use group constraints, where we say that a group of point must be in the same cluster. More generally, we can use as a constraint any positive definite function over any possible clique of the graph



**CRF-Clustering with Generalized Constraints** When using the generalized constraints the full posterior described in [Equation A.2](#) becomes

$$P(Y, \Theta | X, C) = \frac{1}{Z_2} \exp \{ -(\Phi_C + \Phi_\Theta + \Phi_X + \Phi_D) \} \quad (\text{A.6})$$

where

$$\Phi_C = \sum_{(y_i, y_j) \in \mathcal{N}^C} \omega_{ij} \phi_C(y_i, y_j) \quad (\text{A.7})$$

$$\Phi_\Theta = \sum_{(y_i, y_j, \theta_k, \theta_l) \in \mathcal{N}^\Theta} \omega_{ijk} \phi_\Theta(y_i, y_j, \theta_k, \theta_l) \quad (\text{A.8})$$

$$\Phi_X = \sum_{(y_i, y_j, x_i, x_j) \in \mathcal{N}^X} \omega_{ij}^x \phi_X(y_i, y_j, x_i, x_j) \quad (\text{A.9})$$

$$\Phi_D = \sum_{i=1}^n \sum_{\theta_k \in \Theta} \omega_k d_{\theta_k}(x_i) \quad (\text{A.10})$$

Unfortunately, the full use of generalized constraints increase the complexity of the model. Theoretically, we should add a link in the graphical model for every nodes, making the embedded graph fully connected. However, efficiency can be brought back by using a delaunay triangulation to form the links.

### A.2.3 Inference

In this section we will explain how to perform inference on the CRF-clustering model. The main problem, here, is that this model is a mixed continue-discrete one. Inference in such a model is known to be complex, and cannot be always performed in closed form. Moreover, even iterative algorithms like belief propagation needs to be adapted as messages are of different dimension and nature.

Under the assumptions that: 1) the parameters of the distortion functions can be obtained easily (e.g. with a least square technique) when the labels are given, 2) the parameter generalized constraints are in a particular form, it is possible to obtain a simple iterative algorithm that find an optimal solution. The algorithm is shown in [Algorithm 14](#)

---

#### Algorithm 14: CRF-Clustering

---

```

1 for  $i = 1$  to  $max_{iteration}$  do
2   | perform MAP inference on the “label” CRF with the max-sum algorithm
3   | perform ML estimation on the “parameter” CRF
4 end

```

---

Where the “label” CRF is the original CRF conditioned on the parameters. And the “label” CRF is the original one conditioned on the labels. As for the “label” one, the parameters are treated as constant with the value of the previous iteration. The resulting model is only a discrete one and the max-sum algorithm can be used. Once we obtained the labels, we can

perform standard ML estimation on the parameters, as the “parameter” CRF is a disconnected graph with a separate clique for every cluster. However, particular care has to be taken when dealing with the parameter generalized constraints. While there is no problem in having them in the “label” CRF, as the MAP inference algorithm can deal with them, some problems arise in the “parameter” one. Simply put, in order to have this algorithm work we restrict the parameter generalized constraints to have the form

$$\phi_{\Theta}(y_i, y_j, \theta_k, \theta_l) = d(\theta_k, \theta_l)\delta(y_i, y_j) \quad (\text{A.11})$$

where  $d(\theta_k, \theta_l)$  is any distance function which is always positive and equals to 0 iff  $\theta_k = \theta_l$  and  $\delta(y_i, y_j)$  is the Kronecker delta function. Under this assumption, this term disappears from the objective function in the “parameter” CRF. The proof is quite simple. Once we have the label fixed, we only have two cases: 1) the points are assigned to the same cluster; 2) the points are assigned to two different clusters. In case 1), we have that the term  $d(\theta_k, \theta_l)$  is equal to 0, as the two points belong to the same cluster and  $\theta_k = \theta_l$ . In case 2), we have that the term  $\delta(y_i, y_j)$  is equal to 0, as the points belong to different clusters. Finally, the algorithm converges, as it minimizes the objective function at every step.

## Appendix B

# Multi Hypothesis Data Association

In this appendix we present a technique to estimate the state of heterogeneous features from inaccurate sensors. The proposed approach exploits the reliability of the feature extraction process in the sensor model and uses a Rao-Blackwellized particle filter to address the data association problem.

### B.1 Problem formulation

Let  $F = \{f^1, \dots, f^K\}$  be a set of  $K$  features which are present in the environment, where  $K$  may be either known or not (see [Section B.3](#) for further details on this). Each feature has associated a state  $x_t^k$ , which can evolve during time. The nature of the state is strictly connected with the class of the corresponding feature. Therefore, such states neither belongs to the same space nor have the same dimension. The state evolution of each feature is modeled by a stochastic process ruled by the distribution:

$$p(x_{t+1}^k | x_t^k) \tag{B.1}$$

The features persist over time. The robot is able to observe a subset of the features at time. Those features are detected by a complex algorithm, and a reliability measure about the detection is provided<sup>1</sup>. The number of features is modelled as a Poisson distribution, with parameter  $\lambda_n V$ , where  $V$  is the dimension of the explored space and  $\lambda_n$  is the scattered rate of the features. There are also false alarms and the number of false alarms also follows a Poisson distribution with parameter  $\lambda_f$ , where  $\lambda_f$  is the false alarm rate.

Without loss of generality, we restrict ourselves to consider just one observation for time step. Let  $z_t = \langle y_t, Rel(y_t) \rangle$  the reading at time  $t$ . This reading is composed by two parts:  $y_t$  is an observation of the feature<sup>2</sup>;  $Rel(y_t)$  is a vector of numerical values that encodes the reliability of the detection process for each feature. For the sake of clarity, we use the notation  $Rel^k(y_t), k = 1, \dots, K$  to denote the probability of the readings being generated

---

<sup>1</sup>The reliability is intended as a measure on how likely the feature is detected and associated to an already existing one

<sup>2</sup>More precisely,  $y_t$  is a raw data, which is transformed into a feature observation according to that feature's class

by the  $k$ -th feature,  $Rel^0(y_t^j)$  to denote the probability of the reading being a false alarm and  $Rel^{K+1}(y_t^j)$  to denote the probability of the readings being generated by a new feature. It is worth noting, that  $Rel^0(y_t^j)$  and  $Rel^{K+1}(y_t^j)$  do not depend on the detection process, but are instead modelled by the Poisson distributions defined above.

The observations  $y_t$  are generated from

$$\begin{cases} p(\Psi^k(y_t)|x_t^k) & \text{if the observation is from } x_t^k \\ u_t & \text{otherwise} \end{cases} \quad (\text{B.2})$$

where  $p(\Psi^k(y_t)|x_t^k)$  is the likelihood of the state given the observation,  $\Psi^k$  is a function that maps the raw data  $y_t$  into the measurement space of the  $k$ -th feature and  $u_t \sim \text{Unif}(\mathcal{R})$  is a random process for false alarm,  $\mathcal{R}$  being the domain of  $y^t$ . This likelihood depends on the class associated to the feature and in general is different among the features. In the rest of the paper, we will omit writing the function  $\Psi$  for notational simplicity.

Association among observations and features is obtained through a joint association event [Oh, Russell, & Sastry, 2004]  $\alpha_t = \{T_t^0, T_t^1, \dots, T_t^{K_t}\}$ , where  $T_t^k$  is the set of observations associated to the feature  $k$ . At every time step, the system associates an observation  $z_t$  with a feature: for instance, if  $z_t$  is associated with the  $k$ -th feature, then the correspondent track  $T_t^k = T_{t-1}^k \cup \{z_t\}$  will be updated. Therefore the goal of the system is to provide  $\mathbb{E}(x_t^k|z_{1:t})$ , which is dependent on  $P(\alpha_t|z_{1:t})$ .

In the following, for notational simplicity, we will make use of the term label to address the association between observations and features. We map an association event into a sequence of labels  $L_{1:t} = \{L_1, \dots, L_t\}$  where  $L_i = k$  if the observation  $i$  is associated to the feature  $k$ . Using this notation, we can derive the incremental update equation of the filter.

## B.2 Rao-Blackwellized Particle Filters

The previous problem can be well described within the Bayesian framework, resulting in estimating the joint posterior over the associations and features' state,  $p(X_{1:t}, L_{1:t}|z_{1:t})$ , given the history of observations. Once obtained this distribution, it is straightforward to obtain  $p(X_t|z_{1:t})$ . One way is to simply marginalize out the association. This is similar to a Joint Probabilistic Data Association Filter, which simulates an exact Bayes filter and compute the expected posterior by just marginalizing out the last association made [Shalom & Fortmann, 1988]. A second way is to obtain  $p(X_t|z_{1:t}, \hat{L}_{1:t})$ , being  $\hat{L}_{1:t}$  a Maximum A Posteriori estimator for  $p(L_{1:t}|z_{1:t})$ . This approach is in a way similar to the Multiple Hypothesis Tracker; the main difference lies in the fact that RBPF is a fully probabilistic approach, while MHT uses some deterministic rules to add and delete association tracks.

Exact inference in such a model is not possible, since the number of possible associations grows hyper-exponentially with the number of measurements. As a matter of fact, this number is equal to the number of set partitions of the set of measurements, which is known as the *Bell number* [Nijenhuis & Wilf, 1978].

However, only a small fraction of the set partitions is feasible with respect to the real associations. Based on this observation, it is reasonable, both from a theoretical and practical perspective, to approximate the distribution of the associations with a sum of different samples. The reason behind this is that the mass of the distribution is concentrated only in a small portion of the state space. In other words, we want to focus our attention on the most probable

associations, which are represented by the samples drawn from the association distribution, by using Monte Carlo methods [Doucet *et al.*, 2000a].

The key idea of the Rao-Blackwellized particle filter for this problem is to estimate a posterior  $p(L_{1:t}|z_{1:t})$  about potential labels  $L_{1:t}$  of the measurements given the observations  $z_{1:t}$ , and to use this posterior to compute a posterior over features' state and feature association.

$$p(X_{1:t}, L_{1:t}|z_{1:t}) = p(X_{1:t} | L_{1:t}, z_{1:t})p(L_{1:t} | z_{1:t}). \quad (\text{B.3})$$

This can be done efficiently, when the posterior over features' state  $p(X_{1:t} | L_{1:t}, z_{1:t})$  can be computed analytically given the knowledge of  $L_{1:t}$  and  $z_{1:t}$ . Luckily, there are several situations where this assumption holds (like for Linear Gaussian or Hidden Markov models). This technique is known as Rao-Blackwellized Particle Filter and is proved to reduce the variance of the estimate, according to the Rao-Blackwell Theorem [Doucet *et al.*, 2000b].

In the next section we will show how to use the Rao Blackwellized Particle Filters in this context, by providing the proposal distribution and the weight computation, first when the number of features is fixed and known, then when their number is not known in advance and can increase over time.

### B.3 State Estimation with RBPF

In the previous section, we described the general framework of Rao Blackwellized Particle Filter. Here we show how to instantiate the framework in the context of heterogeneous features' state estimation. To this end, we need to define what information the samples (particles) represent and the nature of the distributions involved.

Each sample is characterized by the following  $N + 2$ -uple:

$$s_t^{(i)} = \langle w_t^{(i)}, L_{1:t}^{(i)}, \theta_t^{1,(i)}, \theta_t^{2,(i)}, \dots, \theta_t^{N,(i)} \rangle \quad (\text{B.4})$$

where  $w_t^{(i)}$  represents the importance weight,  $L_{1:t}^{(i)}$  the association history and each  $\theta_t^{n,(i)}$  the sufficient statistic of the corresponding feature, e.g. mean and variance if they are represented as Gaussians.

Recalling that the association is independent from the past reliability given the last one<sup>3</sup>, we can write the distribution in the following recursive way

$$p(L_{1:t}|z_{1:t}) \propto p(y_t|L_{1:t}, y_{1:t-1}, Rel(y_{1:t})) \cdot p(L_{1:t}|Rel(y_{1:t}), y_{1:t-1}) \quad (\text{B.5})$$

$$= p(y_t|L_{1:t}, y_{1:t-1})p(L_t|Rel(y_t), L_{1:t-1}) \cdot p(L_{1:t-1}|z_{1:t-1}) \quad (\text{B.6})$$

We now have a recursive distribution, which is well suited to be implemented in the Sequential Monte Carlo framework. We choose as proposal the distribution:

$$p(L_t|Rel(y_t), L_{1:t-1}) \quad (\text{B.7})$$

While this is not the optimal proposal, as we do not use the measurements  $y_t$ , the reliability allows us to highly reduce the association space. This is because the reliability can be seen

<sup>3</sup>The past reliabilities are embedded in the previous labels, thus resulting in this independence property

as a direct observation of the label, as it reflects the information provided by the detection algorithm. Experimental results show the differences between using or not such information (see [Section B.4](#))

After the sampling stage, we apply the importance sampling principle, obtaining the following weight distribution:

$$w_t = w_{t-1}p(y_t|L_{1:t}, y_{1:t-1}) \quad (\text{B.8})$$

which is easy to evaluate given the fact that we have already computed the predicted state distribution  $p(X_t|L_{1:t}, y_{1:t})$  using the sufficient statistics embedded within the particles.

In the end, we adopt an adaptive resampling schema, by considering the effective number of particles [[Liu, 1996](#)]. This number,

$$N_{eff}(t) = \frac{1}{\sum_{i=1}^N (w_t^{(i)})^2} \quad (\text{B.9})$$

is an approximative measure which tells us how well the set of samples approximates the goal distribution and is strongly related to the variance in the particle weights. We perform a resampling step when this quantity falls below a certain threshold,  $N_{eff} < N_{tsh}$ .

### B.3.1 Fixed and known number of features

When the number of features to estimate is fixed and known, the problem is slightly simpler, as the association is constrained to one of the existing classes.

With respect to the proposal distribution, we have two important distributions to take into account. The first is given by the frequency of the different features in the environment. The second is given by the reliability and reflects our degree of belief about the association provided. We can use the fact that in a fixed and known number of features, the new label  $L_t$  is given by the product  $p(Rel(y_t)|L_t)p(L_t|L_{1:t-1})$  as the reliability does not depend on the previous labels. We can obtain  $p(L_t|L_{1:t-1})$  by marginalizing the multinomial distribution over the past associations. The parameters of this distribution have to be estimated, in order to reflect the real labels' frequency. To do so, we use a MAP estimate, using the Dirichlet distribution as a conjugate prior.

As for the weight, we need to define the conditional likelihood distribution, which strongly depends on the label associated to the reading. We can define the weight, by marginalizing with respect to the estimated state of the feature, obtaining:

$$p(y_t|L_{1:t}, y_{1:t-1}) = \int p(y_t|x_t^{L_t})p(x_t^{L_t}|T_t^{L_t})dx_t^{L_t} \quad (\text{B.10})$$

This integral can be computed analytically for some distribution, such as Gaussians and discrete distributions. Otherwise, numerical or stochastic methods can be used to obtain a close approximation.

### B.3.2 Variable and unknown number of features

The previous framework can be extended to handle an unknown number of features. First of all, we notice that the main difference in dealing with a known or unknown number of

features lies in the sampling procedure, namely in the distribution  $p(L_t|Rel(y_t), L_{1:t-1})$ . In this scenario, we need to focus also on the fact that observations can come from an unknown feature, and model this fact appropriately. We use the Dirichlet Process, in order to estimate the probability of having a new class and assigning the observation to already existing classes. This distribution is often used in the infinite mixture model, and is considered a natural extension of the standard Dirichlet distribution when the number of classes is not known in advance [Ranganathan & Dellaert, 2006; Blackwell & MacQueen, 1973].

Using the Dirichlet Process as our prior over the parameters of  $p(L_t|L_{1:t-1})$ , and the independence of the reliability of time  $t$  from previous labeling, we can still reduce ourselves to:

$$p(L_t|Rel(y_t), L_{1:t-1}) = p(Rel(y_t)|L_t)p(L_t|L_{1:t-1}) \quad (\text{B.11})$$

when:

$$p(L_t|L_{1:t-1}) = \begin{cases} \frac{|T_{t-1}^i|}{|L_{1:t-1}|+c} & i = 1, \dots, K \\ \frac{c}{|L_{1:t-1}|+c} & i = K + 1 \end{cases} \quad (\text{B.12})$$

with  $c$  encoding our belief on the number of features in the environment.

The importance weight defined in Equation B.10 for the fixed case is still valid in this situations. We just want to notice that in the case of the new feature, the integral reduces to the computation of the normalizing factor of the distribution (as the state is equal to the observation due to the initialization).

## B.4 Experiments

We use an abstract simulator to perform an extensive quantitative analysis of correctness and completeness of the association algorithm. We use a Markov Chain to simulate the observations arrival. This chain is tuned in order to simulate a robot path inside an environment. This is achieved by giving higher probability to persistent move in the chain and it results in a burst of observations of one feature, followed by a burst of another feature and so on. The state of the features is given by a simple one dimensional Gaussian, as we decided to focus our attention on the association process, rather than on the estimation one. The observation of the state are sampled from a Gaussian distribution with different values for the variance. The reliability values for the feature association are sampled from a Dirichlet distribution, whose parameters are tuned in order to obtain slightly higher values of reliability for the correct association, simulating a real detection algorithm.

We performed several experiments, varying the number of features to be tracked, the peakness of the Dirichlet distribution and the variance of the observations. In all those cases, we compute two measures, in order to evaluate the performances.

The first metric is related to the correctness of the association, and is given by:

$$\frac{TruePositive}{TruePositive + FalsePositive} \quad (\text{B.13})$$

This value measures the percentage of correct associations made by the algorithm, with respect to the complete set of associations provided.

The second metric is related to the completeness of the association, and is given by:

$$\frac{TruePositive}{TruePositive + FalseNegative} \quad (\text{B.14})$$

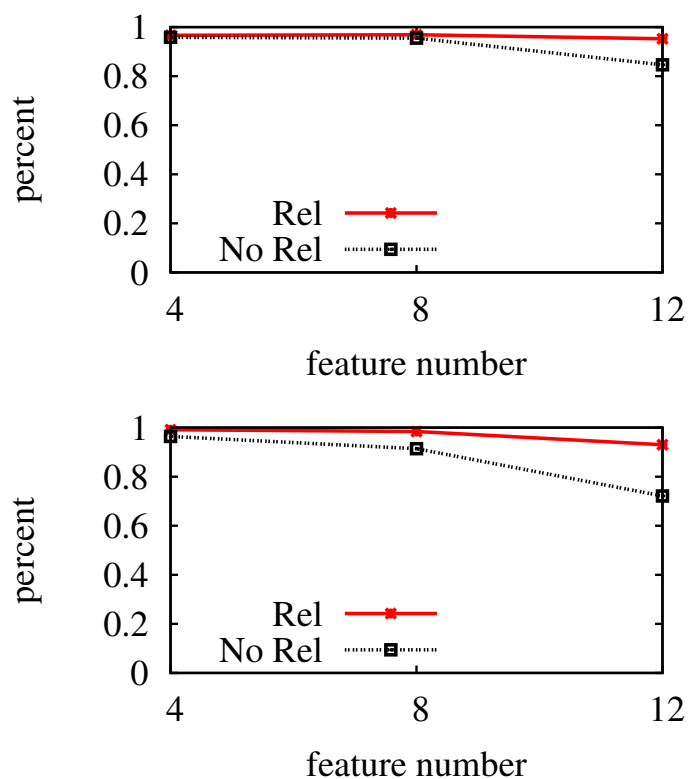


Figure B.1: Number of features: The graphs show the variation of the two metrics with respect to the number of features and the use of reliability. We show the results regarding correctness (top) and completeness (bottom)

This value measures the percentage of correct associations made by the algorithm, with respect to the complete set of correct associations.

In order to prove the efficacy of using the reliability, we also performed the same experiments without using the reliability in the data association process, notice that, such an approach is similar to the one presented in [Sarkka, Vehtari, & Lampinen, 2004]. We use the same parameters and the input data for both the approaches. The data set was composed of 700 simulated observations, with 5 runs with different random seeds. The number of particles used in the experiments was 1000, which allows for real time execution. The features are scattered within the environment at about  $10m$  of distance.

In the first experiment, we analyze the variation of the performances when varying the number of features. When the number of features increases the performance of both the algorithms decreases, as one would expect. However, this decrease in performance is much more evident when the reliability is not taken into account, as can be seen in Figure B.1. This is due to the increased combinatorial space when more features are presented. The reliability provides us some kind of "hint" about the right associations, which reduces the search space.



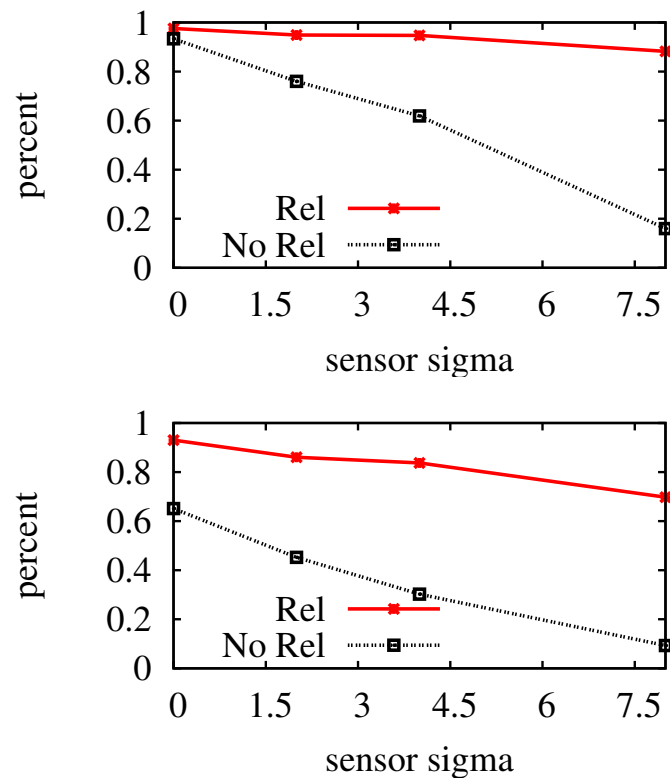


Figure B.2: Precision of the sensor: The graphs show the variation of the two metrics with respect to the precision of the sensor and the use of reliability. We show the results regarding correctness (top) and completeness (bottom)

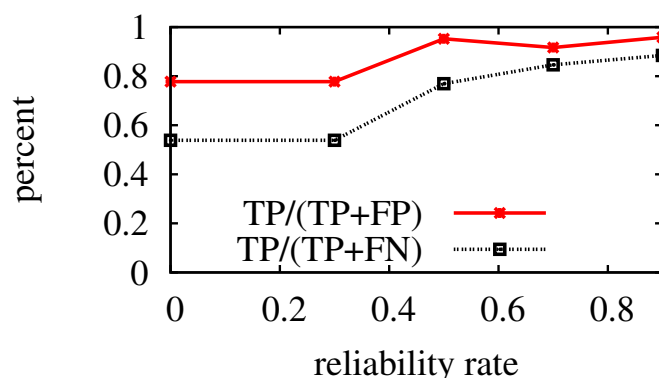


Figure B.3: Reliability: The graphs show the variation of the two metrics with respect to the peakiness of the reliability. Notice that a reliability of 0 means not using the reliability.

In the second experiment, we vary the precision of the sensor. The sensor is related to the analytical part of the features' state and its variance influences the importance weight computation. Figure B.2 shows the result. As we can see, less precision in the sensor has some impact on the precision of the algorithm. This is due to the fact that when features are too close in the state space, there is not so much difference in the importance weight of a wrong association. As before the results show that even if there is a decrease in performance, such decrease is much more evident when the reliability is not taken into account.

Finally, we performed some experiments to show how the performance varies according to different levels of reliability. In Figure B.3 the results are plotted. As one would expected, the performance increases when the reliability grows. In the experiments we let this value to vary from 0.4 percentage of getting the real association up to a 0.9 one. When the reliability value is 0 reliability is not used in the algorithm.

Further experiments with high dimensional features results in similar classification rate, showing good robustness of the association with respect to the feature structure.

# List of Algorithms

1	Sequential Importance Sampling (SIS)	14
2	Sampling Importance Resampling (SIR)	15
3	Systematic Resampling	16
4	Approximated Particle Filter	75
5	localizationUpdate()	76
6	explorationUpdate()	77
7	loopUpdate()	78
8	Merge Clusters	100
9	Monte Carlo Smoothing v1	113
10	Monte Carlo Smoothing v2	115
11	Frozen-Time Smoother	117
12	ght(refMap, localMap, $\hat{s}$ )	118
13	Loopy Intersection Propagation	136
14	CRF-Clustering	155



# List of Figures

2.1	Kalman filter. The predict (top) and update (bottom) phases of the Kalman filter in the onedimensional case. the thick line in both graphs represent the estimation on the state and covariance for the prediction and update equation.	9
2.2	Sampling from a proposal distribution. The picture shows a proposal distribution and the samples drawn from it. The more dense are the samples in a region, the higher is the probability density in that region. We compute the probability mass falling in an interval $\Delta p$ by summing the weights of the samples falling in the interval.	12
2.3	The Importance Sampling principle. The picture shows the samples, weighted according to the Importance Sampling principle. The ratio between the proposal and the target distribution is sketched in red. The particles are weighted according to this mismatch and their size reflects the weight value.	13
3.1	Example of a landmark map. The picture shows the map of the Victoria Park dataset. The cyan dots represented the features while the blue line the robot path. Courtesy of Juan Nieto.	23
3.2	Example of grid map. The picture shows the map of the ACES data set. The gray value represent the probability of a cell of being occupied. The darker a cell is, the higher is the probability.	24
3.3	Example of a semantic/topological map. The picture shows a typical indoor environment. The blue nodes represent rooms in the environment, while the red ones represent the corridors. The edges represent the connections among the different rooms. Courtesy of Oscar Martinez Mozos.	25
3.4	Example of a hybrid map. The picture shows the map of the ACES data set. The graph nodes represent the topological skeleton, while the small local maps represent its metrical part.	26
4.1	SLAM Problem seen as a DBN, the map is <i>static</i> . The observed variables are depicted in gray	46
4.2	This picture illustrates two different proposal distributions. The robot moves along the black line. Left: samples generated according to the odometry motion model. Right: samples generated according to the scan-matcher error model.	48

4.3	Particles distributions typically observed during mapping. In a featureless open space the proposal distribution is the raw odometry motion model (a). In an open corridor, the particles distributes along the corridor (b). In a dead end corridor, the uncertainty is small in all of the directions (c). . . . .	49
4.4	The RBPF results of a long navigation, using two different proposal distributions. . . . .	50
4.5	The effect of an unneeded resampling step. The left picture depicts the correct map, and trajectory. The robot moves from $A$ to $B$ , and it observes the white object at location $l$ . In the middle picture, a possible 3 particles RBPF behavior is depicted. The red and the green trajectory are wrong, while the blue one is correct. In the right picture, a resampling occurs, and only the wrong red trajectory survives. The real trajectory and landmark are represented by the dotted lines. All the robot trajectories descending from the red one share the same wrong map. Notice that even if the final robot location of the green trajectory is correct, the error inherited by the red path persists. . . .	51
4.6	This figure depicts the trajectory tree, and the compressed DP-SLAM tree. On the right the original trajectory evolution is depicted, on the right the compressed tree obtained by i) deleting the nodes which are not reachable from the last level nodes, and ii) collapsing the paths among consecutive branches. The tree is bounded both in depth and in width by the number of particles. . . . .	53
5.1	The mapping process automaton. . . . .	56
5.2	Introspective analysis of RBPF: exploration (left), loop closing (center) and localization (right). The triangles represents the different robot poses. The best particle is shown in black, while the others in dark yellow. The brown rectangle depict the bounding box of the current reading . . . . .	57
5.3	Introspective analysis of KF: exploration (left), loop closing (center) and localization (right). The black triangle represent the robot pose with its covariance ellipse. The map landmarks with their respective covariance ellipses are depicted in dark yellow. The red stars represent the current observation. . . .	57
5.4	Particle cluster example: (left) The posterior about the robot poses. (right) The corresponding topologies and posterior after clustering. . . . .	60
5.5	The map representation used in our approach. . . . .	61
5.6	Proposal distribution in the exploration situation: Image (a) depicts the pose of the robot and its local map for one particle. The proposal computed is represented by the green/gray ellipse. Image (b) depicts the local map for another particle. The proposal computed in the robot reference frame is shown in (c), and (d) illustrates the proposal of the $i$ -th particle translated into the reference frame of the $j$ -th. . . . .	65
5.7	Proposal distribution in the localization situation. Left: The robot moves according to the dashed arrow. The maximum likelihood pose computed for a robot pose. In red are the observations. Right: We can obtain the maximum likelihood pose for other particles by applying the robot translation vector to a local map relative reference frame. . . . .	67
5.8	Local vs. loop map. The local map is used for the proposal distribution. The loop map for the weight computation. . . . .	68

5.9	Weight approximation. The graphs show the evolution of the weight variation $w_t^{(i)}/w_{t-1}^{(i)}$ of three different particles during a loop closure. The blue line shows the weight variation computed using Equation 5.20, while the red line shows the weight variation computed using Equation 5.21. The observation along the trajectories of the particles under the two different weighting strategies and the sequence of the observations are the same. The top images show typical results, the bottom one depicts one of the worst result during our experiments. . . . .	70
5.10	Fingerprint computation. a) Two trajectory/map instances before closing a loop. b) The computation of the two fingerprints and the corresponding grouping. The clustering of the topologies is done through the fingerprints. . .	72
5.11	The red/gray circles and lines in these three images represent the nodes and edges of $\mathcal{G}^{[s]}$ . In the left image, $\mathcal{I}(s)$ contained two nodes and in the middle image the robot closes the loop to reduce its uncertainty. After this it continues to explore new terrain (right image). . . . .	73
5.12	Some of the maps generated by the proposed approach: the Intel Lab, the ACES Building at the university of Texas, the Edmonton convention center, and the Brucecon mines. . . . .	82
5.13	The map learned by our approach, using data acquired at the MIT Killian Court. The top picture shows the final result. The bottom picture shows the internal algorithm representation. . . . .	83
5.14	This plot depicts the number of patches in the memory and the number of clusters over time for the MIT dataset using 1500 particles. . . . .	84
6.1	A sequence of three consecutive scans obtained by a car navigating in a urban environment. The car is facing an intersection while another one is passing by. The different colors indicate different scans. The laser returns caused by the moving car are indicated by the red square. . . . .	92
6.2	Graphical representation of the CRF-Clustering model. The hidden states $\mathbf{x}_i$ indicate the corresponding clusters for each of the pairs of points in the scan. The observations $\mathbf{z}_i$ correspond to local features such as the distance between the points in the pair after the transformation given by the a cluster parameters (rotation and translation). $R_j$ and $T_j$ indicate respectively rotation and translation for the clusters. . . . .	96
6.3	Inference procedure in the CRF clustering model. The parameters of the clusters $R$ and $T$ are used to compute the local features $\mathbf{z}$ (top). With the observations $\mathbf{z}$ computed, a conventional message passing procedure in a chain graphical model is used to compute the MAP assignments for the clusters $\mathbf{x}$ (middle). Finally, observations and clusters assignments $\mathbf{x}$ are used to recompute the cluster parameters $R$ and $T$ (bottom). . . . .	99
6.4	Typical convergence behavior for CRF clustering. The pictures show the likelihoods for the clustering assignment. Dark red represents higher likelihoods and dark blue lower likelihoods. After the first iteration, the probability mass is distributed among 8 clusters. In the second iteration, clusters 1 and 8 are merged into cluster 4. In the third iteration, cluster 2 is merged into cluster 9, which gives the final result with 3 clusters only. This sequence corresponds to the scan of Figure 6.6. . . . .	101

6.5	Example of consistency-based (CBD) results (top) and CRF-Clustering results (bottom). Red dots indicate the current scan, blue dots the previous one (mostly hidden). The colored markers around dots indicate the cluster assignments. It can be noticed that CBD fails to detect the moving object (all points in the same green cluster). The same problem does not exist with CRF-Clustering which explicitly models the object motion. The static points are in the green cluster, moving object points in the purple cluster. The bounding box on the right image indicates the moving object being correctly detected.	104
6.6	Modified K-means results (top) and CRF-Clustering results (bottom). The different symbols indicate the cluster assignment and the red dots indicate the current scan. It can be noticed that the clustering solution found by K-Means is significantly noisy (various different clusters). This is avoided with CRF-Clustering with the use of the neighborhood dependencies. The bounding box on the bottom image indicate the moving object being correctly detected.	106
7.1	An incremental estimate of robot position in a huge, ambiguous environment. As can be seen, the approach was able to correct the local errors, but not the global ones. This results in a map which is still inconsistent.	110
7.2	This picture illustrates the difference between modelling loop closing as smoothing or as filtering. Figure 7.2(a) shows the the odometry data of a robot moving in a square environment. The two paths A–B and E–F correspond to the same corridor; assume that the robot can relocalize itself while traversing this corridor. If one models loop closing as filtering, one would get the constraint $B = F$ which corresponds to the resulting map in (b). If one models loop closing as smoothing, the result is the constraint $A = E$ , which results in the more consistent map in (c).	111
7.3	Example of Frozen-Time Smoother behavior.	121
7.4	Example of approximated Monte Carlo smoothing behavior	122
7.5	Results for the Aces environment. In all the plots, the $x$ axis measures the linear distance along the chunk in meters; each chunk is about 30 meters long. The red dots are individual samples; the black bold line is the samples average.	123
7.6	Results in the Intel environment. In all the plots, the $x$ axis measures the linear distance along the chunk in meters; each chunk is about 30 meters long. The red dots are individual samples; the black bold line is the samples average.	124
8.1	Message flow of BP on a simple graph at time $t$ for node 1. In the left image node 1 computes its marginal from the messages sent by its neighbors. In the right image, node 1 calculates the new messages and send them . The messages are computed according to Equation 8.9 and Equation 8.10.	129
8.2	The shape of the CI update. The thick outer ellipses represent the covariances of <b>A</b> and <b>B</b> . The dashed ellipses represent resulting covariances of <b>C</b> by using different values of $\omega$ .	133
8.3	An example loopy graph (left) and its spanning tree (right). The off-tree edges are the dashed edges in the graph.	134



8.4	Iterations of LBP on a random generated graph with 1000 nodes. The top left figure shows the perturbed graph. . . . .	137
8.5	Average and standard deviation of the relative error between nodes in random generated graphs of 500, 700 and 1000 nodes, with precision 1000. The plot shows the variation of this error during several iterations . . . . .	138
8.6	Average and standard deviation of the relative error between nodes in random generated graphs of 250 nodes, with precision 50, 100 and 200. The plot shows the variation of this error during several iterations . . . . .	139
8.7	Average and standard deviation of the Frobenius norm of the node estimates of a randomly generated network. This measures the quality of the approximation of the three approaches as a function of the size of the network. . . . .	140
8.8	Average and standard deviation of the minimum eigenvalue of the difference matrix of nodes estimates of a randomly generated network. This measures the conservativeness of the estimate. . . . .	140
8.9	Analysis of the covariance approximation on the Intel dataset. The upper plot shows the approximation error and the lower one shows the conservativeness analysis of the different nodes. Note that LBP is overconfident, BP on a spanning tree is overly conservative, and our approach (LIP) provides an intermediate result, being in general closer to the exact estimate. . . . .	141
8.10	Mean and marginal covariances comparison on the Intel Lab dataset: the exact covariances, computed by inverting the full matrix, are depicted in red (solid line), the approximated ones are in green (dashed line). Results obtained by <i>a)</i> Loopy Belief Propagation. <i>b)</i> Belief Propagation on a Minimum Spanning Tree. <i>c)</i> Loopy Intersection Propagation. <i>d)</i> The grid map of the environment. Notice how the distribution mean reflects the environment topology.	143
8.11	Mean and marginal covariances comparison on the Aces dataset: the exact covariances, computed by inverting the full matrix, are depicted in red (solid line), the approximated ones are in green (dashed line). Results obtained by <i>a)</i> Loopy Belief Propagation. <i>b)</i> Belief Propagation on a Minimum Spanning Tree. <i>c)</i> Loopy Intersection Propagation. <i>d)</i> The grid map of the environment. Notice how the distribution mean reflects the environment topology. . . . .	144
B.1	Number of features: The graphs show the variation of the two metrics with respect to the number of features and the use of reliability. We show the results regarding correctness (top) and completeness (bottom) . . . . .	162
B.2	Precision of the sensor: The graphs show the variation of the two metrics with respect to the precision of the sensor and the use of reliability. We show the results regarding correctness (top) and completeness (bottom) . . . . .	163
B.3	Reliability: The graphs show the variation of the two metrics with respect to the peakiness of the reliability. Notice that a reliability of 0 means not using the reliability. . . . .	164



# List of Tables

5.1	The behavior of the RBPF during mapping . . . . .	58
5.2	The behavior of the KF during mapping . . . . .	58
5.3	This table summarizes the number of particle required for generating a good map, for several environments, using the presented approximated approach, and a straightforward RBPF implementation in [Grisetti, Stachniss, & Burgard, 2005] . . . . .	80
5.4	Comparison of memory and computational resources based on the MIT dataset using a PC with a 1.3 GHz CPU. . . . .	81
6.1	Comparison between CBD, K-means and CRF clustering . . . . .	103



# Bibliography

- [Agrawal & Konolige, 2006] Agrawal, M., and Konolige, K. 2006. Real-time localization in outdoor environments using stereo vision and inexpensive gps. In *ICPR '06: Proceedings of the 18th International Conference on Pattern Recognition*, 1063–1068. Washington, DC, USA: IEEE Computer Society.
- [Anguelov *et al.*, 2002] Anguelov, D.; Biswas, R.; Koller, D.; Limketkai, B.; Sanner, S.; and Thrun, S. 2002. Learning hierarchical object maps of non-stationary environments with mobile robots. In *Proc. of the Conf. on Uncertainty in Artificial Intelligence (UAI)*.
- [Arumampalam *et al.*, 2001] Arumampalam, S.; Maskell, S.; Gordon, N.; and Clapp, T. 2001. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing* XX.
- [Atiquzzaman & Akhtar, 1994] Atiquzzaman, M., and Akhtar, M. W. 1994. Complete line segment description using the hough transform. *Image and Vision Computing* 12:267–273.
- [Bailey, Nieto, & Nebot, 2006] Bailey, T.; Nieto, J.; and Nebot, E. 2006. Consistency of the fastslam algorithm. *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on* 424–429.
- [Ballard, 1981] Ballard, D. H. 1981. Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition* 13(2):111–122.
- [Bar-Shalom & Li, 1995] Bar-Shalom, Y., and Li, X.-R. 1995. *Multitarget-Multisensor Tracking: Principles and Techniques*. Yaakov Bar-Shalom.
- [Basu, Banerjee, & Mooney, 2002] Basu, S.; Banerjee, A.; and Mooney, R. 2002. Semi-supervised clustering by seeding. In *Proc. of Int. Conf. on Machine Learning*.
- [Basu *et al.*, 2006] Basu, S.; Bilenko, M.; Banerjee, A.; and Mooney, R. 2006. Probabilistic semi-supervised clustering with constraints. In Chapelle, O.; Schölkopf, B.; and Zien, A., eds., *Semi-Supervised Learning*. Cambridge, MA: MIT Press.
- [Beevers, 2007] Beevers, K. R. 2007. Fixed-lag sampling strategies for particle filtering slam. In *2007 IEEE International Conference on Robotics and Automation (ICRA 2007)*, submitted.
- [Bengtsson & Baerfeldt, 2001] Bengtsson, O., and Baerfeldt, A. 2001. Localization in changing environments - estimation of a covariance matrix for the idc algorithm. In *Proc.*

- IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 4, 1931–1937.
- [Besag, 1975] Besag, J. 1975. Statistical analysis of non-lattice data. *The Statistician* 24.
- [Besl & MacKay, 1992] Besl, P. J., and MacKay, N. 1992. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
- [Biber & Strasser, 2003] Biber, P., and Strasser, W. 2003. The normal distributions transform: a new approach to laser scan matching. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*.
- [Bilenko & Mooney, 2003] Bilenko, M., and Mooney, R. 2003. Adaptive duplicate detection using learnable string similarity measures. In *Proc. of the 9th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*.
- [Blackwell & MacQueen, 1973] Blackwell, D., and MacQueen, J. B. 1973. Ferguson distributions via poly urn schemes. *Annals of Statistics* 1:353–355.
- [Blanco, Fernandez-Madriral, & Gonzalez, 2008] Blanco, J.-L.; Fernandez-Madriral, J.-A.; and Gonzalez, J. 2008. Toward a unified bayesian approach to hybrid metric–topological slam. *Robotics, IEEE Transactions on* 24(2):259–270.
- [Borges, 2000] Borges, G. A. 2000. A split-and-merge segmentation algorithm for line extraction in 2-d range images. In *ICPR '00: Proceedings of the International Conference on Pattern Recognition*, 1441. Washington, DC, USA: IEEE Computer Society.
- [Bosse *et al.*, 2003] Bosse, M.; Newman, P.; Leonard, J.; and Teller, S. 2003. An ALTAS framework for scalable mapping. In *Proc. of the IEEE Int. Conf. on Robotics & Automation*, 1899–1906.
- [Calisi *et al.*, 2007a] Calisi, D.; Farinelli, A.; Grisetti, G.; Iocchi, L.; Nardi, D.; Pellegrini, S.; Tipaldi, D.; and Ziparo, V. A. 2007a. Contextualization in mobile robots. *ICRA'07 Workshop on Semantic Information in Robotics*.
- [Calisi *et al.*, 2007b] Calisi, D.; Farinelli, A.; Grisetti, G.; Iocchi, L.; Nardi, D.; Pellegrini, S.; Tipaldi, D.; and Ziparo, V. A. 2007b. Uses of contextual knowledge in mobile robots. In *Proc. of the 10th Congress of the Italian Association for Artificial Intelligence (AI\*IA 2007)*, 543–554.
- [Campbell, 2004] Campbell, J. 2004. I.: Techniques for evaluating optical flow for visual odometry in extreme terrain. In *In: Proc. of Intelligent Robots and Systems (IROS)*, 3704–3711.
- [Castellanos *et al.*, 1999] Castellanos, J.; Montiel, J.; Neira, J.; and Tardos, J. 1999. The spmap: a probabilistic framework for simultaneous localization and map building. *IEEE Transactions on Robotics and Automatics*.
- [Censi & Tipaldi, 2008] Censi, A., and Tipaldi, G. D. 2008. Lazy localization using the frozen time smoother. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 3167–3172.

- [Censi, Iocchi, & Grisetti, 2005] Censi, A.; Iocchi, L.; and Grisetti, G. 2005. Scan matching in the hough domain. In *ICRA*.
- [Censi, 2007a] Censi, A. 2007a. An accurate closed-form estimate of icp's covariance. In *Proc. IEEE International Conference on Robotics and Automation*, 3167–3172.
- [Censi, 2007b] Censi, A. 2007b. On achievable accuracy for range-finder localization. In *Proc. IEEE International Conference on Robotics and Automation*, 4170–4175.
- [Censi, 2008] Censi, A. 2008. An icp variant using a point-to-line metric. In *Proc. IEEE International Conference on Robotics and Automation ICRA 2008*, 19–25.
- [Cheng, Maimone, & Matthies, 2005] Cheng, Y.; Maimone, M.; and Matthies, L. 2005. Visual odometry on the mars exploration rovers. *Systems, Man and Cybernetics, 2005 IEEE International Conference on* 1:903–910 Vol. 1.
- [Choset, , & J., 2000] Choset, H.; ; and J., B. 2000. Sensor-based exploration: The hierarchical generalized voronoi graph. *Int. Journal of Robotics Research* 19(2).
- [Comport, Malis, & Rives, 2007] Comport, A.; Malis, E.; and Rives, P. 2007. Accurate quadrifocal tracking for robust 3d visual odometry. *Robotics and Automation, 2007 IEEE International Conference on* 40–45.
- [Cummins & Newman, 2007] Cummins, M., and Newman, P. 2007. Probabilistic appearance based navigation and loop closing. In *Proc. IEEE International Conference on Robotics and Automation (ICRA'07)*.
- [Cummins & Newman, 2008] Cummins, M., and Newman, P. 2008. FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance. *The International Journal of Robotics Research* 27(6):647–665.
- [Dellaert, 2005a] Dellaert, F. 2005a. Square root SAM: Simultaneous localization and mapping via square root information smoothing. Technical Report GIT-GVU-05-11, Georgia Tech.
- [Dellaert, 2005b] Dellaert, F. 2005b. Square Root SAM: Simultaneous location and mapping via square root information smoothing. In *Proc. of Robotics: Science and Systems (RSS)*.
- [Dempster, Laird, & Rubin, 1977] Dempster; Laird; and Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B* 39(1).
- [Doucet *et al.*, 2000a] Doucet, A.; de Freitas, J.; Murphy, K.; and Russel, S. 2000a. Rao-Blackwellized particle filtering for dynamic bayesian networks. In *Proc. of the Conf. on Uncertainty in Artificial Intelligence (UAI)*, 176–183.
- [Doucet *et al.*, 2000b] Doucet, A.; Freitas, N.; Murphy, K.; and Russel, S. 2000b. Rao-blackwellized particle filtering for dynamic bayesian networks. In *in Proceedings of the 16th Annual Conference on Uncertainty in Artificial Intelligence (UAI 2000)*.
- [Doucet, De Freitas, & Gordon, 2001] Doucet, A.; De Freitas, N.; and Gordon, N. 2001. *Sequential Monte Carlo Methods in Practice*. Springer Verlag.

- [Doucet, 1998] Doucet, A. 1998. On sequential simulation-based methods for bayesian filtering. Technical report, Signal Processing Group, Dept. of Engineering, University of Cambridge.
- [Duckett, Marsland, & Shapiro, 2002] Duckett, T.; Marsland, S.; and Shapiro, J. 2002. Fast, on-line learning of globally consistent maps. *Journal of Autonomous Robots* 12(3):287–300.
- [Duda, Hart, & Stork, 2001] Duda, O.; Hart, P.; and Stork, D. 2001. *Pattern Classification*. Wiley-Interscience, second edition.
- [Dudek *et al.*, 1991] Dudek, G.; Jenkin, M.; Milios, E.; and Wilkes, D. 1991. Robotic exploration as graph construction. *Robotics and Automation, IEEE Transactions on* 7(6):859–865.
- [Elfes, 1989] Elfes, A. 1989. *Occupancy Grids: A Probabilistic Framework for Robot Perception and Navigation*. Ph.D. Dissertation, Department of Electrical and Computer Engineering, Carnegie Mellon University.
- [Eliazar & Parr, 2003] Eliazar, A., and Parr, R. 2003. Dp-slam: Fast, robust simultaneous localization and mapping without predetermined landmarks. In *Proc. 18th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 1135–1142.
- [Estrada, Neira, & Tardos, 2005] Estrada, C.; Neira, J.; and Tardos, J. 2005. Hierarchical slam: Real-time accurate mapping of large environments. *Robotics, IEEE Transactions on* 21(4):588–596.
- [Eustice, Singh, & Leonard, 2005] Eustice, R.; Singh, H.; and Leonard, J. 2005. Exactly sparse delayed-state filters. In *Proc. of the IEEE Int. Conf. on Robotics & Automation*, 2428–2435.
- [Eustice, Walter, & Leonard, 2005] Eustice, R.; Walter, M.; and Leonard, J. 2005. Sparse extended information filters: Insights into sparsification. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 641–648. To appear.
- [Fischler & Bolles, 1981] Fischler, M. A., and Bolles, R. C. 1981. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. of the ACM* 24:381–395.
- [Folkesson & Christensen, 2004] Folkesson, J., and Christensen, H. 2004. Graphical SLAM - a self-correcting map. In *Proc. of the IEEE Int. Conf. on Robotics & Automation*.
- [Fox *et al.*, 2003] Fox, D.; Ko, J.; Konolige, K.; and Stewart, B. 2003. A hierarchical bayesian approach to the revisiting problem in mobile robot map building. In *Proc. of the Int. Symposium of Robotics Research*.
- [Frese & Duckett, 2003] Frese, U., and Duckett, T. 2003. A multigrid approach for accelerating relaxation-based slam. In *In Proc. of the IJCAI Workshop Reasoning with Uncertainty in Robotics*.
- [Frese & Hirzinger, 2001] Frese, U., and Hirzinger, G. 2001. Simultaneous localization and mapping - a discussion. In *Proc. of the IJCAI Workshop on Reasoning with Uncertainty in Robotics*, 17–26.



- [Frese, Larsson, & Duckett, 2005] Frese, U.; Larsson, P.; and Duckett, T. 2005. A multilevel relaxation algorithm for simultaneous localisation and mapping. *IEEE Transactions on Robotics* 21(2):1–12.
- [Friedman, Fox, & Pasula, 2007] Friedman, S.; Fox, D.; and Pasula, H. 2007. Voronoi random fields: Extracting the topological structure of indoor environments via place labeling. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*.
- [Gamini Dissanayake *et al.*, 2001] Gamini Dissanayake, M. W. M.; Newman, P.; Clark, S.; Durrant-Whyte, H.; and Csorba, M. 2001. A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Transactions on Robotics and Automation*.
- [Gil *et al.*, 2006] Gil, A.; Reinoso, O.; Fernandez, C.; Vicente, A.; Rottmann, A.; and Mozos, O. M. 2006. Simultaneous localization and mapping in unmodified environments using stereo vision. In *Proceedings of the International Conference on Informatics in Control, Automation and Robotics*, 302–309.
- [Gilks & Berzuini, 2001] Gilks, W., and Berzuini, C. 2001. Following a moving target - Monte Carlo inference for dynamic Bayesian models. *Journal of the Royal Statistical Society, Series B* 61(1).
- [Gordon, Salmond, & Ewing, 1993] Gordon, N.; Salmond, D.; and Ewing, C. 1993. A novel approach to nonlinear nongaussian bayesian estimation. In *In Proceedings F*, 107–113.
- [Grisetti *et al.*, 2006] Grisetti, G.; Tipaldi, G. D.; Stachniss, C.; Burgard, W.; and Nardi, D. 2006. Speeding up rao blackwellized slam. In *Proc. of the IEEE Int. Conf. on Robotics & Automation*, 442–447.
- [Grisetti *et al.*, 2007a] Grisetti, G.; Stachniss, C.; Grzonka, S.; and Burgard, W. 2007a. Tree parameterization for efficiently computing maximum likelihood maps using stochastic gradient descent. In *Proc. of Robotics: Science and Systems (RSS)*. To appear.
- [Grisetti *et al.*, 2007b] Grisetti, G.; Tipaldi, G.; Stachniss, C.; Burgard, W.; and Nardi, D. 2007b. Fast and accurate slam with rao-blackwellized particle filters. *Robotics and Autonomous Systems, Special issue on Simultaneous Localization and Map Building* 55(1):30–38.
- [Grisetti *et al.*, 2008] Grisetti, G.; Rizzini, D.; Stachniss, C.; Olson, E.; and Burgard, W. 2008. Online constraint network optimization for efficient maximum likelihood map learning. *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on* 1880–1885.
- [Grisetti, Stachniss, & Burgard, 2005] Grisetti, G.; Stachniss, C.; and Burgard, W. 2005. Improving grid-based SLAM with Rao-Blackwellized particle filters by adaptive proposals and selective resampling. In *Proceedings of the IEEE International Conference on Robotics & Automation (ICRA)*.
- [Grisetti, Stachniss, & Burgard, 2006] Grisetti, G.; Stachniss, C.; and Burgard, W. 2006. Improved techniques for grid mapping with Rao-Blackwellized particle filters. *IEEE Transactions on Robotics*. In press.

- [Grisetti, 2006] Grisetti, G. 2006. *Scaling Rao-Blackwellized Simultaneous Localization And Mapping to Large Environments*. Ph.D. Dissertation, University of Rome 'La Sapienza', Dipartimento Di Informatica e Sistemistica.
- [Grzonka *et al.*, 2007] Grzonka, S.; Plagemann, C.; Grisetti, G.; and Burgard, W. 2007. Look-ahead proposals for robust grid-based slam. In *Proc. of the International Conference on Field and Service Robotics (FSR)*.
- [Gutmann & Konolige, 1999] Gutmann, J.-S., and Konolige, K. 1999. Incremental mapping of large cyclic environments. In *Proc. of the IEEE Int. Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, 318–325.
- [Gutmann, Weigel, & Nebel, 2001] Gutmann, J.; Weigel, T.; and Nebel, B. 2001. A fast, accurate, and robust method for self-localization in polygonal environments using laser-range-finders. *Advanced Robotics Journal*.
- [Gutmann, 1996] Gutmann, J.-S. 1996. Vergleich von algorithmen zur selbstlokalisierung eines mobilen roboters. Master's thesis, University of Ulm, Ulm, Germany. (in German).
- [Haehnel *et al.*, 2003] Haehnel, D.; Burgard, W.; Wegbreit, B.; and Thrun, S. 2003. Towards lazy data association in slam. In *Proc. of the Int. Symposium of Robotics Research*.
- [Hähnel *et al.*, 2003a] Hähnel, D.; Burgard, W.; Fox, D.; and Thrun, S. 2003a. An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 206–211.
- [Hähnel *et al.*, 2003b] Hähnel, D.; Triebel, R.; Burgard, W.; and Thrun, S. 2003b. Map building with mobile robots in dynamic environments. In *Proc. of the IEEE Int. Conf. on Robotics & Automation*.
- [Hähnel, Schulz, & Burgard, 2002a] Hähnel, D.; Schulz, D.; and Burgard, W. 2002a. Map building with mobile robots in populated environments. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*.
- [Hähnel, Schulz, & Burgard, 2002b] Hähnel, D.; Schulz, D.; and Burgard, W. 2002b. Map building with mobile robots in populated environments. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*.
- [Harris & Stephens, 1988] Harris, C. G., and Stephens, M. 1988. A combined corner and edge detector. In *Proc. 4th Alvey Vision Conf.*
- [Horn & Schunck, 1992] Horn, B. K. P., and Schunck, B. G. 1992. Determining optical flow. *Shape recovery* 389–407.
- [Howard & Roy, 2003] Howard, A., and Roy, N. 2003. The robotics data set repository (radish). [radish.sourceforge.net](http://radish.sourceforge.net).
- [Howard, Mataric, & Sukhatme, 2001a] Howard, A.; Mataric, M.; and Sukhatme, G. 2001a. Relaxation on a mesh: a formalism for generalized localization.

- [Howard, Matarić, & Sukhatme, 2001b] Howard, A.; Matarić, M.; and Sukhatme, G. 2001b. Relaxation on a mesh: a formalism for generalized localization. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 1055–1060.
- [Howard, 2004] Howard, A. 2004. Multi-robot mapping using manifold representations. In *IEEE International Conference on Robotics and Automation*, 4198–4203.
- [Iocchi, Pellegrini, & Tipaldi, 2007] Iocchi, L.; Pellegrini, S.; and Tipaldi, G. D. 2007. Building multi-level planar maps integrating LRF, stereo vision and IMU sensors. In *Proc. of IEEE International Workshop on Safety, Security and Rescue Robotics (SSRR)*.
- [Julier & Uhlmann, 1997] Julier, S., and Uhlmann, J. 1997. A nondivergent estimation algorithm in the presence of unknown correlations. In *Proc. of the American Conf. on Control (ACC)*.
- [Julier & Uhlmann, 2001] Julier, S., and Uhlmann, J. 2001. A counter example to the theory of simultaneous localization and map building. In *Proc. of the IEEE Int. Conf. on Robotics & Automation*, 4238–4243.
- [Julier, Uhlmann, & Durrant-Whyte, 1995] Julier, S.; Uhlmann, J.; and Durrant-Whyte, H. 1995. A new approach for filtering nonlinear systems. In *Proc. of the American Control Conference*, 1628–1632.
- [Kaess, Ranganathan, & Dellaert, 2007] Kaess, M.; Ranganathan, A.; and Dellaert, F. 2007. iSAM: Fast incremental smoothing and mapping with efficient data association. In *Proc. of the IEEE Int. Conf. on Robotics & Automation*. To appear.
- [Kalman, 1960] Kalman, R. 1960. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*.
- [Klein, Kamvar, & Manning, 2002] Klein, D.; Kamvar, S.; and Manning, C. 2002. From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In *Proc. of 19th Int. Conf. on Machine Learning*.
- [Koffka, 1935] Koffka, K. 1935. *Principles of Gestalt Psychology*. Lund Humphries, London.
- [Kouzoubov & Austin, 2004] Kouzoubov, K., and Austin, D. 2004. Hybrid topological/metric approach to slam. *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on* 1:872–877 Vol.1.
- [Kuipers & Byun, 1991] Kuipers, B., and Byun, Y.-T. 1991. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Journal of Robotics & Autonomous Systems* 8:47–63.
- [Kumar & Hebert, 2003] Kumar, S., and Hebert, M. 2003. Discriminative random fields: A discriminative framework for contextual interaction in classification. In *Proc. of the International Conference on Computer Vision (ICCV)*.
- [Lafferty, McCallum, & Pereira, 2001] Lafferty, J.; McCallum, A.; and Pereira, F. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of the Int. Conf. on Machine Learning*.

- [Lauritzen, 1996] Lauritzen, S. 1996. *Graphical Models*. Oxford Press.
- [Levin & Szeliski, 2004] Levin, A., and Szeliski, R. 2004. Visual odometry and map correlation. *cvpr* 01:611–618.
- [Lingemann *et al.*, 2004] Lingemann, K.; Nuechter, A.; Hertzberg, J.; and Surmann, H. 2004. High-speed laser localization for mobile robots. *Journal of Robotics & Autonomous Systems*.
- [Lisien *et al.*, 2003] Lisien, B.; Morales, D. S.; Kantor, G.; Rekleitis, I.; and Choset, H. 2003. Hierarchical simultaneous localization and mapping. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 448–453.
- [Liu & Nocedal, 1989] Liu, D., and Nocedal, J. 1989. On the limited memory BFGS method for large scale optimization. *Math. Programming* 45(3, (Ser. B)).
- [Liu, 1996] Liu, J. 1996. Metropolized independent sampling with comparisons to rejection sampling and importance sampling. *Statist. Comput.* 6:113–119.
- [Lowe, 2004] Lowe, D. G. 2004. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision* 60:91–110.
- [Lu & Miliotis, 1994] Lu, F., and Miliotis, E. 1994. Robot pose estimation in unknown environments by matching 2d range scans. In *CVPR94*, 935–938.
- [Lu & Miliotis, 1997a] Lu, F., and Miliotis, E. 1997a. Globally consistent range scan alignment for environment mapping. *Journal of Autonomous Robots* 4:333–349.
- [Lu & Miliotis, 1997b] Lu, F., and Miliotis, E. 1997b. Robot pose estimation in unknown environments by matching 2D range scans. *Journal of Intelligent and Robotic Systems* 18:249–275.
- [Malioutov, Johnson, & Willsky, 2006] Malioutov, D.; Johnson, J.; and Willsky, A. 2006. Walk-sums and belief propagation in gaussian graphical models. *Journal of Machine Learning Research* 7.
- [Mataric, 1992] Mataric, M. J. 1992. Integration of representation into goal-driven behavior-based robots. *IEEE Transactions on Robots and Automation* 8(3):304–312.
- [Matei & Meer, 1999] Matei, B., and Meer, P. 1999. Optimal rigid motion estimation and performance evaluation with bootstrap. In *Proc. Conf. Computer Vision and Pattern Recognition*, Fort Collins Co, 339–345.
- [Minguez, Montesano, & Lamiroux, 2006] Minguez, J.; Montesano, L.; and Lamiroux, F. 2006. Metric-based iterative closest point scan matching for sensor displacement estimation. *IEEE Transaction on Robotics* 22(5):1047–1054.
- [Modayil, Beeson, & Kuipers, 2004] Modayil, J.; Beeson, P.; and Kuipers, B. 2004. Using the topological skeleton for scalable global metrical map-building. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 1530–1536.

- [Montemerlo & Thrun, 2003] Montemerlo, M., and Thrun, S. 2003. Simultaneous localization and mapping with unknown data association using FastSLAM. In *Proc. of the IEEE Int. Conf. on Robotics & Automation*, 1985–1991.
- [Montemerlo *et al.*, 2002] Montemerlo, M.; Thrun, S.; Koller, D.; and Wegbreit, B. 2002. FastSLAM: A factored solution to simultaneous localization and mapping. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, 593–598.
- [Montemerlo, Koller, & Wegbreit, 2003] Montemerlo, M.; Koller, S. T. D.; and Wegbreit, B. 2003. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, 1151–1156.
- [Montesano, Minguez, & Montano, 2005] Montesano, L.; Minguez, J.; and Montano, L. 2005. Modeling the static and the dynamic parts of the environment to improve sensor-based navigation. In *Proc. of the IEEE Int. Conf. on Robotics & Automation*.
- [Montiel, Civera, & Davison, 2006] Montiel, J.; Civera, J.; and Davison, A. 2006. Unified inverse depth parametrization for monocular slam. In *Proceedings of Robotics: Science and Systems*.
- [Moravec, 1988] Moravec, H. 1988. Sensor fusion in certainty grids for mobile robots. *AI Magazine* 61–74.
- [Morency & Gupta, 2003] Morency, L.-P., and Gupta, R. 2003. Robust real-time egomotion from stereo images. *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on 2:II-719–22* vol.3.
- [Mozos & Burgard, 2006] Mozos, O. M., and Burgard, W. 2006. Supervised learning of topological maps using semantic information extracted from range data. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*.
- [Murphy, 1999] Murphy, K. 1999. Bayesian map learning in dynamic environments. In *Proc. of the Conf. on Neural Information Processing Systems (NIPS)*, 1015–1021.
- [Neira & Tardós, 2001] Neira, J., and Tardós, J. 2001. Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on Robotics and Automation* 17(6).
- [Neira, Tardos, & Castellanos, 2003] Neira, J.; Tardos, J.; and Castellanos, J. 2003. Linear time vehicle relocation in SLAM. In *Proc. of the IEEE Int. Conf. on Robotics & Automation*, 427–433.
- [Newman & Leonard, 2003] Newman, P., and Leonard, J. 2003. Consistent convergent constant time slam. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*.
- [Newman, Cole, & Ho, 2006a] Newman, P.; Cole, D.; and Ho, K. 2006a. Outdoor slam using visual appearance and laser ranging. *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on* 1180–1187.
- [Newman, Cole, & Ho, 2006b] Newman, P.; Cole, D.; and Ho, K. 2006b. Outdoor SLAM using visual appearance and laser ranging. In *Proc. of the IEEE Int. Conf. on Robotics & Automation*.

- [Newmann, 1999] Newmann, P. M. 1999. *On the Structure and Solution of the Simultaneous Localisation and Map Building Problem*. Ph.D. Dissertation, The University of Sydney.
- [Ni & Dellaert, 2006] Ni, K., and Dellaert, F. 2006. Stereo tracking and three-point/one-point algorithms: a robust approach in visual odometry. *Image Processing, 2006 IEEE International Conference on* 2777–2780.
- [Nijenhuis & Wilf, 1978] Nijenhuis, A., and Wilf, H. 1978. *Combinatorial Algorithms*. Academic Press.
- [Nister, Naroditsky, & Bergen, 2004] Nister, D.; Naroditsky, O.; and Bergen, J. 2004. Visual odometry. *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on* 1:I-652–I-659 Vol.1.
- [Oh, Russell, & Sastry, 2004] Oh, S.; Russell, S.; and Sastry, S. 2004. Markov chain monte carlo data association for general multiple target tracking problems. In *Proc. of 43rd IEEE Conference on Decision and Control (CDC)*.
- [Olson, Leonard, & Teller, 2006] Olson, E.; Leonard, J.; and Teller, S. 2006. Fast iterative optimization of pose graphs with poor initial estimates. In *Proc. of the IEEE Int. Conf. on Robotics & Automation*, 2262–2269.
- [Paskin, 2003] Paskin, M. 2003. Thin junction tree filters for simultaneous localization and mapping. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, 1157–1164.
- [Paz *et al.*, 2005] Paz, L.; Pinés, P.; Neira, J.; and Tardós, J. 2005. Global localization in SLAM in bilinear time. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*.
- [Pearl & Russel, 2000] Pearl, J., and Russel, S. 2000. Handbook of brain theory and neural networks. Technical report, UCLA Cognitive Systems Laboratory,.
- [Pearl, 1988] Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, Inc.
- [Pfister *et al.*, 2002] Pfister, S.; Kriechbaum, K.; Roumeliotis, S.; and Burdick, J. 2002. Weighted range sensor matching algorithms for mobile robot displacement estimation. In *Proc. IEEE International Conference on Robotics and Automation ICRA '02*, volume 2, 1667–1674.
- [Pitt & Shephard, 1999] Pitt, M., and Shephard, N. 1999. Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association* 94(446):590–599.
- [Rabiner, 1989] Rabiner, L. R. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. In *Proceedings of the IEEE*. IEEE. IEEE Log Number 8825949.
- [Ramos, Fox, & Durrant-Whyte, 2007] Ramos, F.; Fox, D.; and Durrant-Whyte, H. 2007. CRF-matching: Conditional random fields for feature-based scan matching. In *Proc. of Robotics: Science and Systems*.

- [Ranganathan & Dellaert, 2006] Ranganathan, A., and Dellaert, F. 2006. A rao-blackwellized particle filter for topological mapping. In *Proc. of the IEEE Int. Conf. on Robotics & Automation*.
- [Ranganathan, Kaess, & Dellaert, 2007] Ranganathan, A.; Kaess, M.; and Dellaert, F. 2007. Loopy SAM. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*.
- [Ranganathan, Menegatti, & Dellaert, 2006] Ranganathan, A.; Menegatti, E.; and Dellaert, F. 2006. Bayesian inference in the space of topological maps. *IEEE Transactions on Robotics* 22(1):92–107.
- [Ranganathan, 2008] Ranganathan, A. 2008. *Probabilistic Topological Maps*. Ph.D. Dissertation, College of Computing, Georgia Institute of Technology.
- [Richardson & Domingos, 2006] Richardson, M., and Domingos, P. 2006. Markov logic networks. *Machine Learning* 62(1-2).
- [Röfer, 2002] Röfer, T. 2002. Using histogram correlation to create consistent laser scan maps. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*.
- [Rosenberg & Hirschberg, 2007] Rosenberg, A., and Hirschberg, J. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proc. of Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.
- [Sarkka, Vehtari, & Lampinen, 2004] Sarkka, S.; Vehtari, A.; and Lampinen, J. 2004. Rao-blackwellized monte carlo data association for multiple target tracking. In *Proc. of 7th Int. Conf. on Information Fusion*, volume I, 583–590.
- [Savelli & Kuipers, 2004] Savelli, F., and Kuipers, B. 2004. Loop-closing and planarity in topological map-building. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*.
- [Schulz *et al.*, 2001] Schulz, D.; Burgard, W.; Fox, D.; and Cremers, A. B. 2001. Tracking multiple moving targets with a mobile robot using particle filters and statistical data association. In *Proc. of the IEEE Int. Conf. on Robotics & Automation*.
- [Schulz *et al.*, 2003] Schulz, D.; Burgard, W.; Fox, D.; and Cremers, A. B. 2003. People tracking with mobile robots using sample-based joint probabilistic data association filters. *I. J. Robotic Res.* 22(2):99–116.
- [Shalom & Fortmann, 1988] Shalom, Y. B., and Fortmann, T. E. 1988. *Tracking and Data Association*. Boston: Academic-Press.
- [Shi & Tomasi, 1994] Shi, J., and Tomasi, C. 1994. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition*, 593–600.
- [Sim *et al.*, 2006] Sim, R.; Elinas, P.; Griffin, M.; Shyr, A.; and Little, J. J. 2006. Design and analysis of a framework for real-time vision-based slam using rao-blackwellised particle filters. In *CRV '06: Proceedings of the The 3rd Canadian Conference on Computer and Robot Vision*, 21. Washington, DC, USA: IEEE Computer Society.

- [Smith, Self, & Cheeseman, 1990] Smith, R.; Self, M.; and Cheeseman, P. 1990. Estimating uncertain spatial relationships in robotics. In Cox, I., and Wilfong, G., eds., *Autonomous Robot Vehicles*. Springer Verlag. 167–193.
- [Stachniss & Burgard, 2005] Stachniss, C., and Burgard, W. 2005. Mobile robot mapping and localization in non-static environments. In *In Proc. of AAAI*. AAAI.
- [Stachniss & Grisetti, 2004] Stachniss, C., and Grisetti, G. 2004. Mapping results obtained with Rao-Blackwellized particle filters. <http://www.informatik.uni-freiburg.de/~stachnis/research/rbpfmapper/>.
- [Stachniss *et al.*, 2005] Stachniss, C.; Hähnel, D.; Burgard, W.; and Grisetti, G. 2005. On actively closing loops in grid-based FastSLAM. *Advanced Robotics* 19(10):1059–1080. To appear.
- [Stachniss, Hähnel, & Burgard, 2004] Stachniss, C.; Hähnel, D.; and Burgard, W. 2004. Exploration with active loop-closing for FastSLAM. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 1505–1510.
- [Stephenson, 2000] Stephenson, T. 2000. An introduction to bayesian network, theory and usage. Technical report, Dalle Molle Institute for Perceptual Artificial Intelligence.
- [Thrun *et al.*, 2004] Thrun, S.; Liu, Y.; Koller, D.; Ng, A.; Ghahramani, Z.; and Durrant-Whyte, H. 2004. Simultaneous localization and mapping with sparse extended information filters. *Int. Journal of Robotics Research* 23(7/8):693–716. To appear.
- [Thrun, Burgard, & Fox, 2000] Thrun, S.; Burgard, W.; and Fox, D. 2000. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. In *Proceedings of the IEEE Int. Conf. on Robotics and Automation (ICRA)*. San Francisco, CA: IEEE.
- [Thrun, Fox, & Burgard, 1998] Thrun, S.; Fox, D.; and Burgard, W. 1998. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning and Autonomous Robots (joint issue)*. To appear.
- [Thrun, 2002] Thrun, S. 2002. Robotic mapping: A survey. In Lakemeyer, G., and Nebel, B., eds., *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann.
- [Tipaldi *et al.*, 2007] Tipaldi, G. D.; Farinelli, A.; Iocchi, L.; and Nardi, D. 2007. Heterogeneous feature state estimation with rao-blackwellized particle filters. In *Proc. of IEEE International Conference on Robotics and Automation (ICRA)*, 3850–3855.
- [Tipaldi, Grisetti, & Burgard, 2007] Tipaldi, G. D.; Grisetti, G.; and Burgard, W. 2007. Approximated covariance estimation in graphical approaches to slam. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*.
- [Tomatis, Nourbakhsh, & Siegwart, 2001] Tomatis, N.; Nourbakhsh, I.; and Siegwart, R. 2001. Combining topological and metric: A natural integration for simultaneous localization and map building. In *Proceedings of the Fourth European Workshop on Advanced Mobile Robots (Eurobot 2001)*.



- [van der Merwe *et al.*, 2000] van der Merwe, R.; de Freitas, N.; Doucet, A.; and Wan, E. 2000. The unscented particle filter. Technical Report CUED/F-INFENG/TR380, Cambridge University Engineering Department.
- [Wagsta *et al.*, 2001] Wagsta, K.; Cardie, C.; Rogers, S.; and Schroedl, S. 2001. Constrained k-means clustering with background knowledge. In *Proc. of 18th Int. Conf. on Machine Learning*.
- [Walter, Eustice, & Leonard, 2007] Walter, M. R.; Eustice, R. M.; and Leonard, J. J. 2007. Exactly sparse extended information filters for feature-based slam. *Int. J. Rob. Res.* 26(4):335–359.
- [Wang & Thorpe, 2002] Wang, C., and Thorpe, C. 2002. Simultaneous localization and mapping with detection and tracking of moving objects. In *Proc. of the IEEE Int. Conf. on Robotics & Automation*.
- [Wang *et al.*, 2007] Wang, C.; Thorpe, C.; Hebert, M.; and Thrun, S. and Durrant-Whyte, H. 2007. Simultaneous localization, mapping and moving object tracking. *The International Journal of Robotics Research* 26(6).
- [Weiss & Freeman, 2001] Weiss, Y., and Freeman, W. T. 2001. Correctness of belief propagation in gaussian graphical models of arbitrary topology. *Neural Computation* 13(10):2173–2200.
- [Weiß, Wetzler, & von Puttkamer, 1994] Weiß, G.; Wetzler, C.; and von Puttkamer, E. 1994. Keeping track of position and orientation of moving indoor systems by correlation of range-finder scans. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 595–601.
- [Welch & Bishop, 2001] Welch, G., and Bishop, G. 2001. An introduction to the kalman filter. Technical report, University of North Carolina at Chapel Hill, Chapel Hill, NC 27599-3175.
- [Wolf & Sukhatme, 2005] Wolf, D., and Sukhatme, G. 2005. Mobile robot simultaneous localization and mapping in dynamic environments. *Autonomous Robots* 19:53–65.
- [Xing *et al.*, 2003] Xing, E.; Ng, A.; Jordan, M.; and Russell, S. 2003. Distance metric learning, with application to clustering with side-information. In *Advances in NIPS, vol. 15*.
- [Zhang, 1994] Zhang, Z. 1994. Iterative point matching for registration of free-form curves and surfaces. *International Journal of Computer Vision* 13(2):119–152.