Active Map Learning for Robots: Insights into Statistical Consistency

Ruben Martinez-Cantin

Ph.D. Dissertation

Main advisor D. José Ángel Castellanos Gómez

Departamento de Informática e Ingeniería de Sistemas
Centro Politécnico Superior
University of Zaragoza, Spain

September 2008

Resumen

El problema de aprendizaje de entornos desconocidos en robótica móvil recibe el nombre de SLAM (Simultaneous Localization and Mapping). Si además el robot tiene que tomar decisiones en este ámbito se denomina SLAM activo o simplemente, exploración. Dada la naturaleza del problema, los métodos actuales de SLAM están basados en una fuerte formulación probabilista, en la forma de filtros Bayesianos. Extensiones de esta formulación en el ámbito de la toma de decisiones en entornos inciertos permiten resolver el problema de forma activa y autónoma, usando, por ejemplo, MDPs (Markov Decision Processes).

Por un lado, la enorme dimensionalidad (temporal, espacial y estadística) del problema de SLAM hace que, matemáticamente, no exista solución completa. Existen soluciones subóptimas presentadas en la literatura actual basadas en aproximaciones lineales de primer orden, como el filtro de Kalman extendido (EKF) o métodos de muestreo, como el filtro de partículas Rao-Blackwellizado (RBPF), pero que sufren de cierta inconsistencia estadística que provoca resultados corruptos o totalmente erróneos a largo plazo. Por otro lado, esta enorme complejidad también implica que las técnicas actuales de MDPs -iteración de valor, iteración de política y búsqueda de política con árboles o gradientes- solo aportan soluciones parciales o se reducen a problemas muy básicos y no realistas.

En esta tesis se analizan las propiedades estadísticas del problema de SLAM y los algoritmos comúnmente empleados, especialmente las fuentes de error. En ese sentido, se investiga la aplicación de aproximaciones lineales de segundo orden como el filtro de Kalman unscented (UKF), métodos de muestreo más avanzados, como el filtro de partículas marginal (MPF), o estimación basada en aproximación estocástica. Estos algoritmos teóricamente mejoran la consistencia estadística de los anteriores, lo cual se demuestra con los correspondientes experimentos.

Por otro lado, se investigan nuevos algoritmos de búsqueda de políticas

más eficientes que permitan solucionar el problema completo de decisión. En este sentido se trabaja con algoritmos de optimización global eficiente para la búsqueda de políticas. Este trabajo esta relacionado con el aprendizaje activo de parámetros usando procesos Gaussianos.

La exploración inteligente de entornos desconocidos puede ampliar los campos de aplicación de la robótica, favoreciendo la integración de los robots en la vida cotidiana. El objetivo final es la combinación de todos estos sistemas, de manera fiable y robusta en una plataforma autónoma que permita al robot realizar tareas peligrosas o repetitivas, con poca o nula supervisión. De este modo, se consigue una mayor robustez frente a situaciones inesperadas y se suprime la necesidad de personal cualificado para su uso.

Abstract

The problem of learning unknown environments in mobile robotics is called SLAM (Simultaneous Localization and Mapping). Furthermore, if the robot has to take some decision, it is called active SLAM or just exploration. The current SLAM methods are rooted on the strong probabilistic formulation in the form of Bayesian filters. There are extensions of this formulation in the field of decision making in uncertain environments that allows to solve the problem in an active and autonomous way, using, for example, MDPs (Markov Decision Processes).

On the one hand, the huge dimensionality of the SLAM problem (temporal, spacial and statistical), make it mathematically intractable. There is no optimal solution. However, there are suboptimal solutions presented in the current literature based on approximations of the Extended Kalman Filter (EKF) or in sampling methods, like the Rao-Blackwellized Particle Filter (RBPF), but they suffer from certain statistical inconsistency that results in corrupt or totally wrong solutions in the long term. On the other hand, that huge complexity also implies that current MDPs techniques -value iteration, policy iteration and trees or gradients based policy search- only provide partial solutions or they can only solve unrealistic toy problems.

In this thesis, we analize the statistical properties of the SLAM problem and the most extended algorithms, especially their error sources. In this way, we investigate the applicability of second order linear approximations like the Unscented Kalman Filter (UKF), advanced sampling methods like the Marginal Particle Filter (MPF) or stochastic approximation based estimation. In theory, these algorithms outperform the statistical consistency of the previous ones, this is validated in the corresponding experiments.

Additionally, we investigate more efficient policy search algorithms to solve the complex problem of decision making. In this way, we work with efficient global optimization algorithms for policy search. This work is highly related with active parameter learning using Gaussian processes.

The intelligent exploration of unknown environments may broaden the application fields of robotics, improving the integration of robots in everyday life. The final target is the combination of all these systems, in a reliable and robust way, in an autonomous platform that allows the robot to perform dangerous or repetitive task, with few or none supervision. In this way, we achieve a better robustness to unexpected situations and we eliminate the necessity of qualified personnel for the use of the robot.

Acknowledgements

During the course of this thesis, I had the fortune of having two persons overseeing my work. First, I am indebted to my advisor José Ángel Castellanos, for believing in me, for all the support and enthusiasm during these years and for helping me to explore my own path while ensuring that I keep on track. I am also indebted to Nando de Freitas, who I had the honor to work with during my visits to UBC and afterwards. He is an excellent professor, an inspiring researcher and, above all, a good friend.

It has been also a pleasure to work in the *Robotics, Perception and Real-Time* group. I am specially indebted to José María Martínez Montiel, José Neira and Juan Domingo Tardós. I learned so much from you, and you have been a source of inspiration. I owe gratitude also to Josechu Guerrero and Carlos Sagüés. They introduce me in research and encourage me to start the thesis. Finally, I really thank Luis Montano for his support during this thesis.

During my time in Vancouver, I enjoyed working and discussing with the people at *LCI*. I would like to thank Eric Brochu, Peter Carbonetto, Mike Chiang, Pantelis Elinas, Per-Erik Forssén, Firas Hamze, Matt Hoffman, Hendrik Kueck, Jim Little, David Lowe, David Meger, Kevin Murphy, Kenji Okuma, Robert Sim and Julia Voguel. In particular, I would like to thank Arnaud Doucet for our fruitful discussions.

At the begging of this thesis, Carlos Orrite gave me the opportunity to work at the Computer Vision Lab, in Zaragoza. I wish to thank my friends and colleagues Jorge Raul Gómez, Juan José Gracia, José Elías Herrero, Jesús Martínez and Greg Rogez

I would like to thank Tim Bailey, Frank Dellaert, Dieter Fox, Udo Frese, Florent Lamiraux, Andrew Ng, Nick Roy and all the anonymous reviewers of my papers for their comments and contributions. Thanks to José Guivant and Eduardo Nebot, for making public the Victoria Park dataset, thanks to George Poyadjis for figure 4.1 and thanks to Rudolph van der Merwe for the ReBeL toolkit.

I also would like to thank all my labmates in Zaragoza, the *DIISasters*, for all the good time I spend there. I would need another section just to put all the names and to express my gratitude. In particular for this thesis, I would like to thank Javier Civera, Javier Minguez, Luis Montesano, Alejandro Mosteo, Ana Cristina Murillo, Lina María Paz and Pedro Piniés for our long discussions about robotics and also María López-Valdés for our long discussions about mathematics. Thank you.

Last but not least, I wish to thank my family and friends for their support, encouragement and patience during these years. And most of all, thank you María for all the love you gave me.

This thesis has been supported by the Dirección General de Investigación of Spain, projects DPI2003-07986 and DPI2006-13578. My stay at UBC was partly supported by Bancaja and NSERC.

Notation

Symbols

$\mathbf{x}$	vector of latent random variables.
$\mathbf{y}$	vector of observed random variables.
θ	vector of unknown parameters.
$\mathbf{x}_{1:t}$	Stacked vector $\mathbf{x}_{1:t} \equiv \{\mathbf{x}_1, \ldots, \mathbf{x}_t\}^T$.
$p(\mathbf{x})$	Distribution of $\mathbf{x}$.
$p(\mathbf{x}\|\mathbf{y})$	Conditional distribution of $\mathbf{x}$ given $\mathbf{y}$.
$\mathbf{z} \sim p(\mathbf{x})$	$\mathbf{z}$ is distributed acording to $p(\mathbf{x})$.
$\mathbf{A}$	Matrix formed by elements A_{ij}.
$\mathbf{A}^T$	Transpose of matrix $\mathbf{A}$.
$\mathbf{A}^{-1}$	Inverse of matrix $\mathbf{A}$.
$\|\mathbf{A}\|$	Determinant of matrix $\mathbf{A}$.
$\mathbb{E}_{p(\mathbf{x})}(\mathbf{x})$	Expectation of the random variable $\mathbf{x}$ according to the distribution $p(\mathbf{x})$.
$\mathbb{I}_S(\mathbf{x})$	Indicator function of the set S (1 if $\mathbf{x} \in S$, 0 otherwise).
$\delta_{\mathbf{x}^{(i)}}(d\mathbf{x})$	Dirac delta function with mean $\mathbf{x}^{(i)}$.
$\Gamma(\cdot)$	Gamma function.
$\exp(\cdot)$	Exponential funtion.
$\log(\cdot)$	Natural logarithmic funtion.
$\mathcal{O}(N)$	The computation complexity is order N operations.
$\oplus$	Euclidean vector composition.
$\ominus$	Euclidean vector inversion.

Distributions

Gaussian	$\mathcal{N}(\mathbf{x};\mu,\Sigma) = \lvert 2\pi\Sigma\rvert^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)\right)$
Inverse-Gamma	$\mathcal{IG}(\mathbf{x};\alpha,\beta) = \frac{\beta\alpha}{\Gamma(\alpha)}\mathbf{x}^{-\alpha-1}\exp(-\frac{\beta}{\mathbf{x}})\mathbb{I}_{[0,+\infty)}(\mathbf{x})$

Abbreviations

AMSE	Average Mean Square Error
ASSC	Adaptive Scale Sample Consensus
EI	Expected Improvement
EKF	Extended Kalman Filter
GP	Gaussian Process
KF	Kalman Filter
MAP	Maximum A Posteriori
MCMC	Markov Chain Monte Carlo
MDP	Markov Decision Process
ML	Maximum Likelihood
MPF	Marginal Particle Filter
OLC	Open Loop Control
OLFC	Open Loop Feedback Control
PCRB	Posterior Cramer-Rao Bound
pdf	probability density function
PF	Particle Filter
POMDP	Partially Observable Markov Decision Process
RBPF	Rao-Blackwellized Particle Filter
RLS	Recursive Least Square
SA	Stochastic Approximation
SIR	Sequential Importance Resampling
SLAM	Simultaneous Localization and Mapping
SMC	Sequential Monte Carlo
TLS	Total Least Square
UKF	Unscented Kalman Filter
w.r.t.	with respect to

Contents

Chapter 1

Introduction

Industrial robots are tied to a controlled and perfectly known environment. Its application is strict and clearly defined. In contrast, service robots have to interact with humans in all kind of situations and scenarios. That means an unknown, uncontrolled and evolving environment. It is impossible to predict in advance the vastness of scenarios, situations and events. There is no generative model of the world. Mobile robots must take decisions based only on internal models and sensory information, which can be proprioceptive or extereoceptive. However, sensing capabilities only provides partial information, no matter how accurate or broad they are.

In this set up, it is logical to claim that mobile robots require the ability to adapt and confront different scenarios, also those not specifically included in their internal models or behaviors. In such a way, the system must create, update and refine the models and behaviors to incorporate new information. Nevertheless, from a technical point of view, it is more important to consider the applicability of the system. For instance, to actually be successfully applied in different situations and events. Thus, the system must accomplish the required task even if the original conditions are not satisfied.

Analogously to psychological studies, the first and easiest approach to applicability comes from the behavior based research. Behaviors are functions that directly map observations into actions. However, due to the locality of sensor readings, decisions are limited to very simple local tasks, like obstacle avoidance. It is impossible to execute long term plans efficiently. Those are purely sensor driven systems where the uncertainty in the sensors and actuators is typically neglected.

Probabilistic approaches include uncertainty management in the algorithm. They allow to integrate past observations to discover some latent variables or

patterns, reducing the total amount of uncertainty. In addition, they can be used to compute ambiguous decisions based on partial information, which are required to interact with humans. Those decisions are based on certain parameters or models that are learnt based on the data gathered. In dynamic systems, like moving robots or people interaction, probabilistic approaches introduce the idea of state, which is a concept inherited from the physics and control literature. It is the minimal set of independent latent variables that represent the internal parameters of the system. In certain manner, probabilistic approaches build cognitive systems with internal representations of the world, splitting the observation/action loop in several cognitive processes.

Furthermore, when we split the decision process, we can add some *intuitions* about what we expect to find in certain parts of the cognitive process. This is specially interesting using Bayesian statistics, where the intuition can be expressed as prior information. Also, Bayesian methods can be easily extended to recursively update and refine the current models.

Finally, probabilistic methods provide a way to simplify the necessary data in order to make a decision. Actual robots and sensors provide huge amounts of data at high frequency. For example, a cheap camera produces more than 9 million pixels per second. Optimal decisions are intractable in that framework, but data mining can provide enough information to achieve an approximate decision under real-time constraints. For example, we can learn a parametric model to abstract the information provided and represent it in a low dimensional space; like learning the shape of an object before grasping it. Hence, decisions can be made on the low dimensional parametric model, instead of the whole dataset.

Let us assume, that a mobile robot has to accomplish a task like "fetch me a beer". The first thing to do is to navigate where the beer is. In every moment, the navigation system needs to know where is the location of the beer with respect to the robot. Also, it has to maintain the knowledge about the current location to come back. Finally, while the robot is moving, it can gather new information from different locations which may improve the knowledge of the environment for current and future tasks.

The problem to recursively determine the localization of an autonomous vehicle navigating in an unknown environment and, concurrently, to learn the underlying structure of the environment is called Simultaneous Localization And Mapping (SLAM). During the last decade, the robotics literature has been populated with scientific work on this field.

From a theoretical point of view, the SLAM problem can be viewed as the problem of estimating an autorregresive latent stochastic process $\{\mathbf{x}_t\}_{t\geq 1}$ -the

state of the robot- and a set of M parameters -the model of the environment- $\{\theta_n\}_{n=1}^{M}$, given a second stochastic process $\{\mathbf{y}_t\}_{t\geq 1}$. This is known as the dual estimation problem, that combines filtering and learning.

In SLAM, like most tracking applications, the data arrive sequentially. Then, a standard approach in control literature consist of modelling the latent process as a discrete-time dynamic system. Furthermore, the algorithm should find a recursive estimate as new data become available. The estimate comprises both the state and the parameters. From a Bayesian viewpoint, learning is basically the problem of finding the maximum likelihood (ML) $p(\mathbf{y}_{1:t}|\theta)$ or maximum a posteriori (MAP) distribution of the parameters $p(\theta|\mathbf{y}_{1:t})$, given all the available data. From a frequentist viewpoint, we are interested only in the optimal value of either function:

$$\theta^*_{ML} = \arg\max_{\theta} p(\mathbf{y}_{1:t}|\theta) \tag{1.1}$$

$$\theta^*_{MAP} = \arg\max_{\theta} p(\theta|\mathbf{y}_{1:t}) \tag{1.2}$$

Similarly, filtering is the problem of finding the belief (Bayesian) or the optimal value (frequentist) in the state space. In practice, most of the applications require only an estimate of the current state of the system, like the robot pose. Then, the solution consists of the recursive computation of the posterior distribution over the parameters and the *filtering distribution* over the state[1] of the state variables $\mathbf{x}_t$ given the observations in the past $\mathbf{y}_{1:t}$ and the initial distribution of the state $p(\mathbf{x}_0)$. Integrating new data in the current estimate can be done using Bayes rule. We can apply recursive estimation of the filtering distribution, provided that the marginal estimate is a sufficient statistic. For certain models, sufficient statistics are intractable. Then, the filtering distribution can be obtained through variable marginalization of the full posterior distribution. In SLAM literature, this is sometimes referred to as the full-SLAM problem.

For linear models, like:

$$\mathbf{y}_t = \mathbf{H}_t\theta_t + \mathbf{w}_t$$

a recursive least squares (RLS) algorithm provides an optimal performance of the recursive parameter estimation [Spall 03]. In fact, the recursive solution quickly converge to the batch solution. Assuming that the underlying distribution is Gaussian, the frequentist and Bayesian versions of RLS are equivalent, since the optimal value of the distribution is the mean and the learning rate is

[1] The filtering distribution of a discrete time dynamic system is the marginal distribution at current time step $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ of the full posterior distribution $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$.

proportional to the covariance, which needs to be computed recursively. Having the mean vector and the covariance matrix, the Gaussian distribution is completely defined. The extension of RLS to the filtering problem was done by R.E. Kalman in his celebrated Kalman filter (KF) [Kalman 60], assuming that a dynamic model of the latent process is provided:

$$\mathbf{x}_{t+1} = \mathbf{F}_t \mathbf{x}_t + \mathbf{v}_t \tag{1.3}$$

$$\mathbf{y}_t = \mathbf{H}_t \mathbf{x}_t + \mathbf{w}_t \tag{1.4}$$

His main contribution was to represent the signal processing problem as a state space dynamic system and find the dual solution to the control problem. Therefore, the latent variables do not represent the value of an arbitrary signal, but the evolution of the state of a dynamic physical system. Although this was intended as a mathematical trick for modeling data, we have seen that the notion of state is more intuitive in many applications, like robotics.

Thanks to the similarities between RLS and KF, joint state and parameter estimation can be easily implemented increasing the dimensionality of the latent space to include the parameters as static elements:

$$\mathbf{x}_{t+1} = \mathbf{F}_t \mathbf{x}_t + \mathbf{v}_t \tag{1.5}$$

$$\mathbf{y}_t = \mathbf{H}_t[\mathbf{x}_t, \theta_t] + \mathbf{w}_t \tag{1.6}$$

Thus, the problem becomes a single multidimensional estimation problem. This approach was first suggested in [Kopp 63, Cox 64]. A theoretical study with convergence and stability results were provided in [Ljung 79]. Ljung's work also related KF inference with a more general Stochastic Approximation (SA) algorithm. We introduce a similar theoretical analysis in section 4.3.1.

However, to guarantee that the estimated values of the posterior and filtering distribution are sufficient statistics, either Gaussianity of the underlaying density functions or linearity in the dynamic model is assumed. Furthermore, Kalman proved that under those assumptions, the estimated latent variables are the orthogonal projection of the observations in the state space. For instance, the solution is unbiased and optimal in terms of variance. These results also applies for the joint state and parameter estimation if the linear-Gaussian assumptions are still satisfied. Thus, as reported in [Dissanayake 01] a theoretical solution of the KF-based SLAM, given that the hypotheses are fulfilled, is consistent and optimal in terms of convergence. Dissanayake et al. [Dissanayake 01] proved three important convergence properties of the EKF-SLAM: (1) the determinant of any submatrix of the map covariance matrix decreases monotonically as observations are successively made; (2) in the limit

as the number of observations increases, the landmark estimates become fully correlated; and (3) in the limit, the covariance associated with any single landmark location estimate reaches a lower bound determined only by the initial covariance in the vehicle location estimate at the time of the first sighting of the first landmark. This sound approach in conjunction with considerable heuristic engineering has produced reasonable solutions; to the point that some researchers have begun voicing the opinion that *"the SLAM problem is solved"*.

While not endorsing the view that SLAM should occupy the center-stage, it is the thesis of this work that the existing SLAM solutions are built upon questionable assumptions and procedures, including linearity, Gaussian distributions, treating static maps with dynamic models and neglecting the variance increase due to sampling in spaces of increasing (potentially unbounded) dimension. This thesis provides some arguments as well as empirical evidence to substantiate this statement.

In general, SLAM models are nonlinear, even when the relationship between parameters and state elements is linear. It has been proved that the general nonlinear filtering problem do not admit a close form solution [Kushner 67]. Many probabilistic algorithms have appeared in the literature to find an efficient and reliable approximation to the solution. However, none of them have proved to be statistically consistent for the SLAM problem, which is nonlinear, nonergodic and nonstationary. In fact, all of them suffer from correlated errors every time step that produce an increasing global error. From an engineering view point, the approximation errors can be acceptable if the global error remains under a *reasonable* threshold. However, SLAM algorithms accumulate those errors, reaching any upper bound. Also, the lack of prior models of the environment results in a lack of error models. Thus, we are unable to compute tight upper bounds for long term navigation. Hence, the algorithms become unreliable from a practical point of view. The bottom line is that there is an urgent need for designing a principled SLAM framework so as to relax these assumptions and eliminate the need for brittle heuristics. It is the intent of this thesis to take a step in this direction and to note that the problem is still open.

An additional problem appears when the information obtained through the sensors is not enough to achieve the mission. For example, the estimation of the object location with respect to the robot may not be accurate enough to grasp it. Sometimes, the robot has to learn the environment actively, gathering enough information to achieve the required level of accuracy as we will introduce in chapter 5. Also, some observations provide more information for the filtering or posterior distribution than others. For example, if we use a

single camera, we do not have depth information. We need to move laterally to be able to triangulate the location of the object. Intuitively, when we are confronted to new tasks, objects or environments, the first reaction is to explore the situation, using conservative actions until we get enough information to be confident.

1.1 A formal introduction to SLAM

This section is intended to provide a precise mathematical formulation of the SLAM problem and introduce different approaches to the solution.

1.1.1 Modelling Dynamic Systems

In the time domain, a dynamic system can be compactly described as a set of state transitions. Intuitively, the state of a dynamic system contains some quantitative information (a set of numbers, a function, etc.) which is the least amount of data one has to know about the past behavior of the system in order to predict its future behavior. The transitions represent the evolution of the state, i.e. one must specify how one state is transformed into another as time passes [Kalman 60]. Although the trajectory of a robot is a continuous function, the controller and sensors provide discrete information, which is simpler to model in a time discrete model. Therefore, transition functions are represented as difference equations.

In Engineering and Physics, a dynamical system has some elements and perturbations that can not be directly included in the model or even observed. For example, many microscopic or internal effects in sensors and actuators or the unpredictable evolution of the environment. Those perturbations are considered as random noise [Thrun 05a]. The presence of this noise also explains the statistical dependence between random observations of the dynamic system at different times.

At a macroscopic level, we can consider macroscopic random phenomena as the combination of many microscopic effects. For instance, the pixel level noise in a digital image is the combination of the quantum properties of the light source, the reflected material and the CCD sensor, but also the injected noise in the electrical signal during transport and conversions.

Thus, under very week assumptions, the central limit theorem guarantees that the random variable resulting of the addition of all those microscopic events converges in distribution to a Gaussian distribution. Furthermore, microscopic phenomena tend to take place much more rapidly than macroscopic

phenomena; therefore, random effects would appear to be independent on a macroscopic time scale. Then, the random noise is called white, that is, all the elements in the sequence are independent

$$p(\mathbf{v}_t|\mathbf{v}_l) = p(\mathbf{v}_t) \qquad \forall\ t > l \tag{1.7}$$

Under this assumption, a first order difference equation represents a *Markov* sequence [Jazwinski 70].

1.1.2 Probabilistic State-Space SLAM

As we have seen in the previous section, the SLAM problem can be interpreted as a sequential estimation and optimization problem which involves a pair of nonstationary discrete-time stochastic processes, $\{\mathbf{x}_t\}_{t\geq 1}$ named *latent process*, and $\{\mathbf{y}_t\}_{t\geq 1}$, named *observation process*, both defined on a probability space. Those processes are related through a mapping function $\mathbf{y}_t = \psi(\mathbf{x}_t)$ which is also unknown. In general, the latent process $\{\mathbf{x}_t\}_{t\geq 1}$ represents the robot location and any other dynamic object included in the models, which is the state of the system. The evolution of the state is represented by a first order (Markovian) autoregressive model $\mathbf{x}_{t+1} = f(\mathbf{x}_t)$. The unknown observation mapping $\psi(\cdot)$ captures the information from the environment.

When the vehicle moves from position at step $t-1$ to position at step t, proprioceptive sensors provide measurements $\mathbf{u}_t$ of relations between robot locations. Thus, the stochastic state vector is updated according to the nonlinear autorregressive model:

$$\mathbf{x}_t = \mathbf{f}(\mathbf{x}_{t-1}, \mathbf{u}_t) \tag{1.8}$$

where the sensor reading $\mathbf{u}_t = \widehat{\mathbf{u}}_t + \mathbf{v}_t$ corresponds to the relative motion $\widehat{\mathbf{u}}_t$ perturbed by some (typically) white Gaussian noise $\mathbf{v}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_{\widehat{\mathbf{u}}_t})$, dependent on the actual motion. Given that any dead-reckoning measurement is independent on the state and the past dead-reckoning measurements, we can assume a nonstationary Markov process with the corresponding Markov transition density $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t)$.

The latent process $\{\mathbf{x}_t\}_{t\geq 1}$ is partially observed through the observation process $\{\mathbf{y}_t\}_{t\geq 1}$. On-board sensors provide, at time t, partial observations $\mathbf{y}_t$ related to the state vector $\mathbf{x}_t$ by the nonlinear measurement model:

$$\mathbf{y}_t = \psi(\mathbf{x}_t) + \mathbf{w}_t \tag{1.9}$$

where $\mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_t)$ represents a white Gaussian perturbation on the observations $\mathbf{y}_t$.

Again, we can assume that the observations conditioned on the latent process $\{\mathbf{x}_t\}_{t\geq 1}$ are independent, with the corresponding marginal density $p(\mathbf{y}_t|\mathbf{x}_t)$.

Depending on the sensor, the observation process can provide direct readings of the robot location -for example, GPS-, or relative measurements -for example, LIDAR or cameras- that observe environment evolution from the robot point of view. In the first case, the mapping function can be known a priori and the problem becomes only a tracking problem [Bar-Shalom 01]. But direct readings require an external facility that may not be accessible in some situations, reducing the applicability of the system. Therefore, we can consider that all observations are related to unknown elements in the environment, being the tracking problem a special case.

1.1.3 SLAM as a dimensionality reduction problem

The joint localization and mapping problem can be seen as a dimensionality reduction problem. The observations follow a manifold, which is the map $\psi(\cdot)$, related to a set of time-varying variables $\mathbf{x}_t$. Also, we have a model of the evolution of $\mathbf{x}_t$, i.e. the state transition function.

The SLAM problem consists in learning the manifold structure while computing the maximum likelihood, or maximum a posteriori, state variables. As can be seen in figure 1.1, this problem is ill-posed. Intuitively, SLAM can be thought as the problem of understanding a message in a completely new language, with unknown vocabulary and alphabet. That problem is, in principle, undecible. But we know that the message is written using some grammar and it requires certain logical structure, like verbs, nouns, etc. In our case, the dynamic model is the grammar, while the mapping function corresponds to the vocabulary.

However, robot mapping is a special case where the observation function can be defined in such a way that the joint estimation become computable. For example, the sensor model is known and that information can be included in the observation function. Also, the sensor noise can be determined beforehand using calibration. From a Bayesian viewpoint, these are the ingredients to fix a prior distribution on the mapping function $\psi(\cdot)$. The sensor model provides enough prior information on the variables to include the learning step in the Bayesian recursion presented in section 1.1.2, even when the actual prior distribution is unknown [Maybeck 82]. Finally, raw sensor data can be preprocessed beforehand. Then, observations can be represented in a *feature* space which can be accessible through the sensor model.

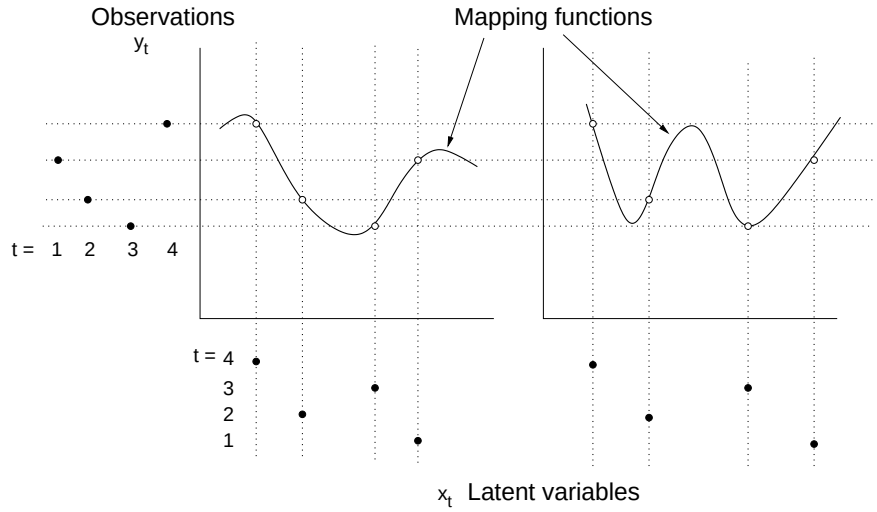


Figure 1.1: The general approach to SLAM problem requires to compute the latent variables and mapping function for a set of observations. The figure shows, in a simple 1D case, that different mapping functions and state evolutions are admissible for the same set of observations. The problem is ill-conditioned and does not have a unique solution no matter how many observations we can get, unless the latent process is deterministic.

Physical map models

The simplest sensor model for mobile robots is the obstacle detector, which returns information about obstacles in the field of view. It assumes that the sensor is based on some kind of ray or beam that impacts any surface with certain properties and it is reflected back towards the sensor. Sonars, lidars, radars and even cameras can be modeled in such a way. The resulting function is a coloring function that labels every point in the space with its corresponding property. To reduce the complexity of obstacle reconstruction, the environment is discretized and represented with a fixed resolution grid, where the only information that we store is the occupancy [Elfes 87]. A simple sensor model using occupancy grid is show in figure 1.2. Some authors has extended the occupancy grids to add new properties to each cell [Stachniss 06, Mozos 05]. Also, Paskin et at. [Paskin 05] propposed an mathematically elegant solution to occupancy mapping of structured environments without grid discretization.

Sensor data can also be modelled as a set of constraints in a graphical model [Gutmann 99, Thrun 05b, Dellaert 06]. Therefore, we can build the

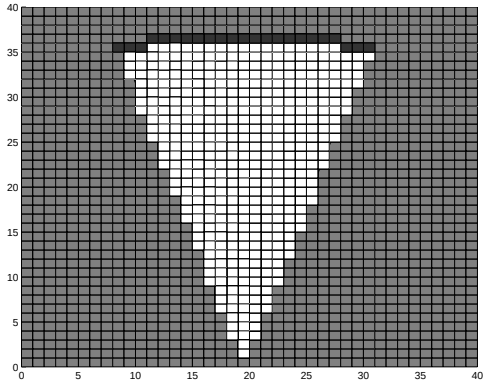


Figure 1.2: Occupancy map of a single observation. We assume that the robot is moving in a planar surface -the floor- and that an obstacle is any element above that surface, thus a simple 2D representation is enough.

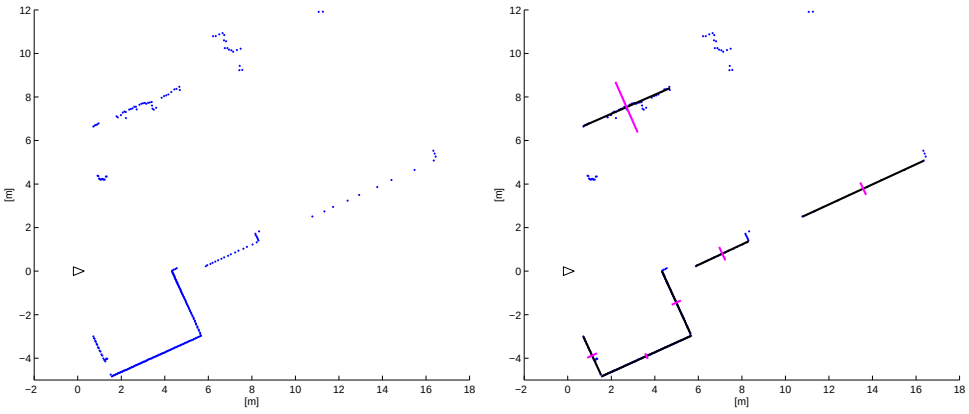


Figure 1.3: Laser scan in a manmade environment. Walls and structures are based on planar surfaces which produces alignments in consecutive scan points. Extracting these lines simplifies the data and increases the robustness of the SLAM algorithm.

map, computing the posterior distribution as a stochastic optimization problem. For example, similarities in subsequent observations provide relations between robot poses. Batch algorithms provide elegant solutions, but the computational cost is high and require to save and process all the data offline. Incremental approaches are more tractable, but rely on approximations or exploit local properties that reduce the global quality. Also, the trajectory increases every step, gathering new data, increasing the computational cost indefinitively even when the robot is navigating in a compact region. Moreover, the latent variables become nonstationary and global consistency can not be guaranteed, except for the batch algorithm. This property will be analyzed in detail in section 4.5.1.

Finally, sensor data can be preprocessed to identify salient features or abstract models of elements of the environment. Using a set of sparse features, we try to capture all the information contained in the raw data, reducing input dimensionality. Also, we can simplify the process of detecting and eliminating spurious readings. This has been extensively studied in related fields like computer vision. Dimensionality reduction can be directly applied to the SLAM problem [Brunskill 05]. Since the features are based on a physical model, we can also include prior knowledge about the type of features and abstract models. Then, the mapping problem is regularized. For example, we can expect to find sharp corners in images which correspond to object borders or surface boundaries. As can be seen in figure 1.3, planar surfaces and straight angles can be found in manmade environments [Martinez-Cantin 06b]. This method will be discussed in chapter 2.

Hereinafter, we are going the use the feature based maps. They provide a compact representation of the environment, the complexity is bounded in compact regions and they admit any level of accuracy, provided enough information is available.

1.2 Feature-based SLAM

In the seminal formulation of the probabilistic state-space SLAM [Smith 88], the stochastic variables represent a set of rigid body transformations that includes the location of the vehicle R with respect to an arbitrary reference B and a set of environment features $\mathcal{F} = \{F_1, \ldots, F_n\}$ w.r.t. the same reference. The feature locations and base reference are assumed to be fixed, i.e. the environment is static. Then, feature locations are not part of the dynamic state, but fixed parameters of the system. In the control literature, the joint state and parameter estimation is called adaptive filtering. As we have seen, the

most extended and simple approach consist in extending the state vector with the corresponding parameters [Ljung 79]. This is also the standard approach in SLAM literature. Then, the estimation process can be defined in terms of a Kalman filter approach, where the estimate includes both state variables -robot location- and parameters -structural map features-. That is, $\mathbf{r}_t = \mathbf{x}_R^B$ is the estimated location of the vehicle at time t w.r.t. the base reference frame B and $\theta = \mathbf{x}_{\mathcal{F}}^B$ is the estimated location of the features also w.r.t. B. For notation simplicity, we drop the base reference frame when the base reference is common or not relevant for the algorithm. Then, the joint state vector at time t is $\mathbf{x}_t = [\mathbf{r}_t, \theta]$.

When the vehicle moves from position at step $t-1$ to position at step t, proprioceptive sensors provide measurements of relations between robot locations $\mathbf{u}_t = \mathbf{x}_{R_t}^{R_{t-1}}$. Thus, the stochastic state vector is updated according to the nonlinear autorregressive model:

$$\mathbf{x}_t = \mathbf{f}_t(\mathbf{x}_{t-1}, \mathbf{u}_t) \tag{1.10}$$

where the sensor reading $\mathbf{u}_t = \widehat{\mathbf{u}}_t + \mathbf{v}_t$ corresponds to the relative motion $\widehat{\mathbf{u}}_t$ perturbed by some (typically) white Gaussian noise $\mathbf{v}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_{\widehat{\mathbf{u}}_t})$, dependent on the actual motion. Assuming that any dead-reckoning measurement[2] is independent on the state and the past dead-reckoning measurements, we can assume a nonstationary Markov process with the corresponding Markov transition density $\mathbf{p}(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t)$.

In the feature based SLAM framework, the model can be simplified to

$$\mathbf{r}_t = \mathbf{f}_t(\mathbf{r}_{t-1}, \mathbf{u}_t) \tag{1.11}$$

because the parameters θ remain constant.

The latent process $\{\mathbf{x}_t\}_{t\geq 1} = \{\mathbf{r}_t\}_{t\geq 1}$ is partially observed through the observation process $\{\mathbf{y}_t\}_{t\geq 1}$. On-board sensors provide, at time t, partial observations $\mathbf{y}_t$ related to the state vector $\mathbf{x}_t$ by the nonlinear measurement model:

$$\mathbf{y}_t = \mathbf{h}_t(\mathbf{x}_t) + \mathbf{w}_t = \mathbf{h}_t(\mathbf{r}_t, \theta) + \mathbf{w}_t \tag{1.12}$$

where $\mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_t)$ represents a white Gaussian perturbation on the observations $\mathbf{y}_t$. In general, measurements from on-board sensors provide observations of relations between the robot and some landmarks $\mathbf{x}_F^R$ where $F \in \mathcal{F}$. Again, we can assume that the observations conditioned on the latent process $\{\mathbf{x}_t\}_{t\geq 1}$ are independent, with the corresponding marginal density $\mathbf{p}(\mathbf{y}_t|\mathbf{x}_t)$.

[2] Provided by any propioceptive sensor like odometers.

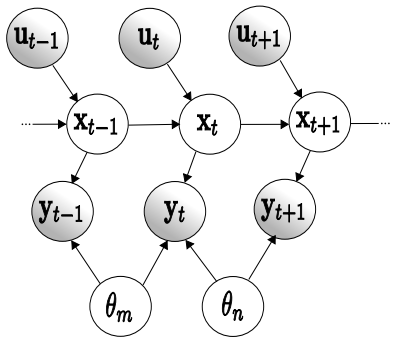


Figure 1.4: Simplified Bayesian network of SLAM process. Shaded circles represents observed variables. Data association and control policy is assumed known.

Finally, the initial belief $\mathbf{p}(\mathbf{x}_0)$ is also known and assumed to be Gaussian. In SLAM, the initial location is not important, since we are interested only in relations. Thus, the initial belief of the robot location collapses to a deterministic arbitrary location of the robot pose. The landmark initial belief is set during the first observation, convolving the sensor location uncertainty and the actual observation uncertainty. Furthermore, it has been proved that setting an artificial initial belief can create inconsistent maps due to the incongruent information loss that appears in the system [Castellanos 04]. The formulation of the SLAM problem can be represented as the Bayesian network shown in figure 1.4.

1.3 Recursive estimation in SLAM

From a Bayesian view-point, let us suppose that an estimate of the stochastic map $\mathbf{p}(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})$ is available at time $t-1$, then, the predicted stochastic map at time t results from the Chapman-Kolmogorov equation:

$$p(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t)p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})p(\mathbf{u}_t)\, d\mathbf{u}_t d\mathbf{x}_{t-1} \tag{1.13}$$

When new information about the state vector is available, it can be incorporated into the state using Bayes' rule:

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) = \frac{p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{y}_{1:t-1})}{\int p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{y}_{1:t-1})d\mathbf{x}_t} \tag{1.14}$$

An important term in equation (1.14) is its denominator, which can be

expresed as $p(\mathbf{y}_t|\mathbf{y}_{1:t-1})$ and it is called *innovation*[3]

$$p(\mathbf{y}_t|\mathbf{y}_{1:t-1}) = \int p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{y}_{1:t-1})d\mathbf{x}_t$$

The innovation can be viewed as the amount of information of the last observation which is new, that is, the information that was not included in previous observations. It plays an important role in sequential algorithms of filtering and learning.

In general, integrals from (1.13) and (1.14) are intractable. Therefore, suboptimal approaches have been applied to solve it. In this thesis we have studied different suboptimal methods in order to find their limitations and features. Chapter 3 presents some analitical techniques rooted on the celebrated Kalman filter. Later, chapter 4 introduces standard and advanced Monte Carlo methods for numerical integration.

It is important to remark that SLAM deals with the estimation of the *filtering distribution* $\mathbf{p}(\mathbf{x}_t|\mathbf{y}_{1:t})$ and not the whole stochastic process, i.e. the *posterior distribution* $\mathbf{p}(\mathbf{x}_{1:t}|\mathbf{y}_{1:t})$. But some SLAM algorithms estimate the filtering distribution as a by-product of the posterior distribution using incremental smoothing. However, those methods requires stronger assumptions to remain tractable. Thus, long term trajectories suffer from larger bias or inconsistencies after several loops. Therefore, it is out of the scope of this thesis, although some implementations in current literature provide excelent results.

1.4 Data Association

Up to now, we have assumed that the identity of every observation is given. In some applications, the correspondences between mapped landmarks and current observations form another set of hidden variables. Furthermore, the main problem in SLAM is the lack of a generative model. The assumption that the robot is in a totally unknown environment implies that there is no prior information about any landmark. In general, it is virtually impossible to distinguish the observation of a new landmark from the noisy observation of a near landmark already mapped. Therefore, data association is an ill-conditioned classification problem.

This lack of prior information is compensated using strong assumptions in the model. The most extended approach in Gaussian SLAM or similar approaches that represent landmark location as a Gaussian distributions is to

[3]also known as the *predictive likelihood* or *evidence*

compute the maximum likelihood data association. Novel landmarks are identified using a null hypothesis testing to a certain level of confidence. Implicitly, there is an assumption on the reliability of the feature detection system and the sparseness of the environment in the testing.

Recent applications of SLAM for computer vision use active search of features in image data [Davison 05]. There is also an active search for novel features. Then, the correspondence problem is trivial, but the assumption of sparseness remains.

In this work, except when otherwise expressed, we use the maximum likelihood data association. When the maximum likelihood estimate includes the joint posterior over a large dimensional space, like the EKF-SLAM, then Branch and Bound techniques are used for computational speedup [Neira 01].

1.5 Active SLAM

The last chapter of this thesis, extends the current work to the problem of active SLAM. In general, SLAM algorithms are independent on the navigation algorithm or the motion policy. However, a fully autonomous robot requires to consider SLAM results in its navigation policy. The intuition of this idea can be seen in figure 5.1.

The theoretical framework for active SLAM that we use is similar to a Partially Observable Markov Decision Process. A POMDP is a tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{R}, p\}$ where $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, respectively; $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ is the immediate reward which may be a random process; $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is the conditional transitional distribution. A *stationary policy* $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ is a mapping from states to action selection probabilities.

In our set up, the state and actions spaces are both known and continuous $\mathcal{S} = \mathbb{R}^{n_{\mathbf{x}}}$ and $\mathcal{A} = \mathbb{R}^{n_{\mathbf{u}}}$. The transition distribution has been previously defined in equation (1.8). However, there is no instant reward function because it depends on the belief. Intuitively, we can imagine two robots moving on a building. They follow exactly the same path, but one of the robots has the sensors turned off -*it is blind*-. At the end, the state is the exactly the same, but the belief is completely different. For the blind robot, the environment is completely new. But the other robot can use previously mapped areas to remain localized. Thus, they may be interested in following different trajectories to explore the rest of building. As a consequence, we can see that the policy is nonstationary. Also, since the environment is totally unknown, the policy can not be transfered for different applications. Therefore, policy learning has to be done online.

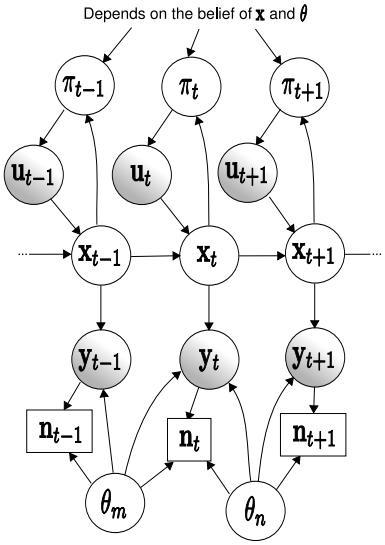


Figure 1.5: Bayesian network of SLAM process. Circles represents continuous variables and squares are discrete variables. Shaded circles represents observed variables. The robot state is represented as $\mathbf{x}$, the observations $\mathbf{y}$, the actions $\mathbf{u}$, the data associations labels $\mathbf{n}$, the map features θ and the local policy π

The problem of a Bayesian reward has been analyzed in the experimental design literature using information theoretic metrics. Using one of those metrics, we can redefine a new reward function $\mathcal{R} : \mathcal{B} \times \mathcal{A} \times \mathcal{B} \rightarrow \mathbb{R}$, where $\mathcal{B}$ in now the belief[4] space. In this case, the policy can be defined as $\pi : \mathcal{B} \times \mathcal{A} \rightarrow [0, 1]$, which is a mapping from believes to action selection probabilities. It is interesting to note that, for a time step t and all the observations up to this time step $\mathbf{y}_{1:t}$, the new formulation is equivalent to the original POMDP. Thus, we can use reinforcement learning techniques to find the local policy π_t, provided that we can compute the belief function at time t.

The general Bayesian network for active feature-based SLAM with unknown data association can be seen in figure 1.5.

[4]In Bayesian inference, the belief is equivalent to the filtering distribution at time t, that is, $p(\mathbf{x}_t|\mathbf{y}_{1:t})$

1.6 Publications

Parts of this thesis have been published in the following journal, conference and workshop papers:

Journal Papers

J.A. Castellanos, R. Martinez-Cantin, J.D. Tards and J. Neira, Robocentric Map Joining: Improving the Consistency of EKF-SLAM, Robotics and Autonomous Systems 55 (2007), 21-29.

Conference Papers

R. Martinez-Cantin, N. de Freitas, A. Doucet and J.A. Castellanos. Active Policy Learning for Robot Planning and Exploration under Uncertainty In Robotics: Science and Systems (RSS), 2007.

R. Martinez-Cantin, N. de Freitas, J.A. Castellanos Analysis of Particle Methods for Simultaneous Robot Localization and Mapping and a New Algorithm: Marginal-SLAM In IEEE International Conference on Robotics and Automation (ICRA), 2007.

R. Martinez-Cantin, N. de Freitas, J.A. Castellanos Multi-Robot Marginal-SLAM. In International Joint Conference on Artificial Intelligence (IJCAI), 2007 Workshop on Multirobotic Systems for Societal Applications.

R. Martinez-Cantin, J.A. Castellanos, J.D. Tards and J.M.M. Montiel. Adaptive Scale Robust Segmentation for 2D Laser Scanner. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2006.

R. Martinez-Cantin and J.A. Castellanos. Bounding Uncertainty in EKF-SLAM: The Robocentric Local Approach. In IEEE International Conference on Robotics and Automation (ICRA), pp. 430-435, 2006.

R. Martinez-Cantin and J.A. Castellanos. Unscented SLAM for Large-Scale Outdoor Environments. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 328-333, 2005.

Chapter 2

Feature detection and extraction

2.1 Introduction

Robot navigation, localization and mapping requires to deal with a high amount of data, sometimes redundant and commonly noisy. Feature detection in robotics tries to mimic the human capability of abstraction which reports three main advantages: (i) *data compression*: Features use a simple parametric model to represent the high amount of data, (ii) *denoising*: Since the feature model is estimated from several measurements, the uncertainty and bias of the final estimation is lower than any of the measurements, (iii) *distinguishability:* High level features and complex structures are easier to identify and match (despite partial observation).

In manmade environments the assumption of strong geometric constraints is reliable. For example, flat surfaces are very common (e.g. walls, doors, tables, bookshelves, etc.). Also, the assumption of a flat floor is feasible. In that situation, the vertical surfaces can be identified using a built-in 2D rangefinder sensor. That kind of sensors provides very reliable dense information. Then, flat surfaces from real world are transformed into straight lines on the sensor plane.

Sensor noise introduces some uncertainty in the measurements, which can be predicted using standard calibration techniques. However, this is not the main source of uncertainty. Depending on the construction technique or wear, real surfaces are not planar but have certain relief and hence it is an issue to define when a cluster of points, not exactly over a line, can be approximated as a straight segment. A key concept in the approximation step is the uncertainty

scale, i.e. the perpendicular dispersion of points belonging to the line, which is defined by the noise standard deviation. It is worth noting that this scale is a property of each scene segment, and hence it has to be determined from the data corresponding to the segment. For example, in Figure 2.1, the cluster of data points corresponding to the ivy covered wall on the left has a bigger scale that the clusters corresponding to the concrete walls on the right, despite being sensed approximately at the same relative location with respect to the sensor, this shows how scale is not only determined by the sensor but also by the scene. Right bottom part of the figure shows a successful clustering and the corresponding uncertainty scale σ estimate (magnified 30 times). Classical benchmark methods [Nguyen 05] do not care about scale computation, thus uncertainty models are constructed based on sensor noise, considering a perfect scene model, or considering the noise scale as an input to the algorithm.

Thereinafter, without losing generality, observations follow the next hypotheses based on the structural assumptions:

- The data points $\mathcal{Y} = \{y^{(1)}, \ldots, y^{(n)}\}$ satisfy $y^{(i)} \in \mathbb{R}^2 \ \forall i = 1, \ldots, n$, because we suppose a robot moving on flat surface (floor), with a 2D rangefinder parallel to the floor that provides range and bearing measurements.
- Environment structures are planes perpendicular to the floor, that appear as straight lines in the sensor plane.
- The perturbation for every data point is iid and perpendicular to the estimated lines, following a Gaussian distribution $X^{(i)} \sim \mathcal{N}(0, \sigma)$.
- The standard deviation σ of the error distribution corresponds to the *scale*, which can vary from one scene segment to another.

Once the scan has been segmented in clusters corresponding to the straight segments, the line fitting with these hypotheses can be optimally solved using Total Least Squares (TLS) [Pearson 01], which in this case is equivalent to Principal Component Analysis (PCA). The problem is that clustering and scale estimation are coupled and have influence on the line fitting, being a unique spuriously clustered point able to ruin down a straight segment estimate for the whole cluster.

Focusing on a cluster, the points belonging to the cluster are inliers. The points not belonging to the cluster are outliers. A first set of outliers are due to noise, or to non straight scene structures. A second group of outliers, the pseudo-outliers, correspond to other clusters. A robust method has to deal successfully with both type of outliers.

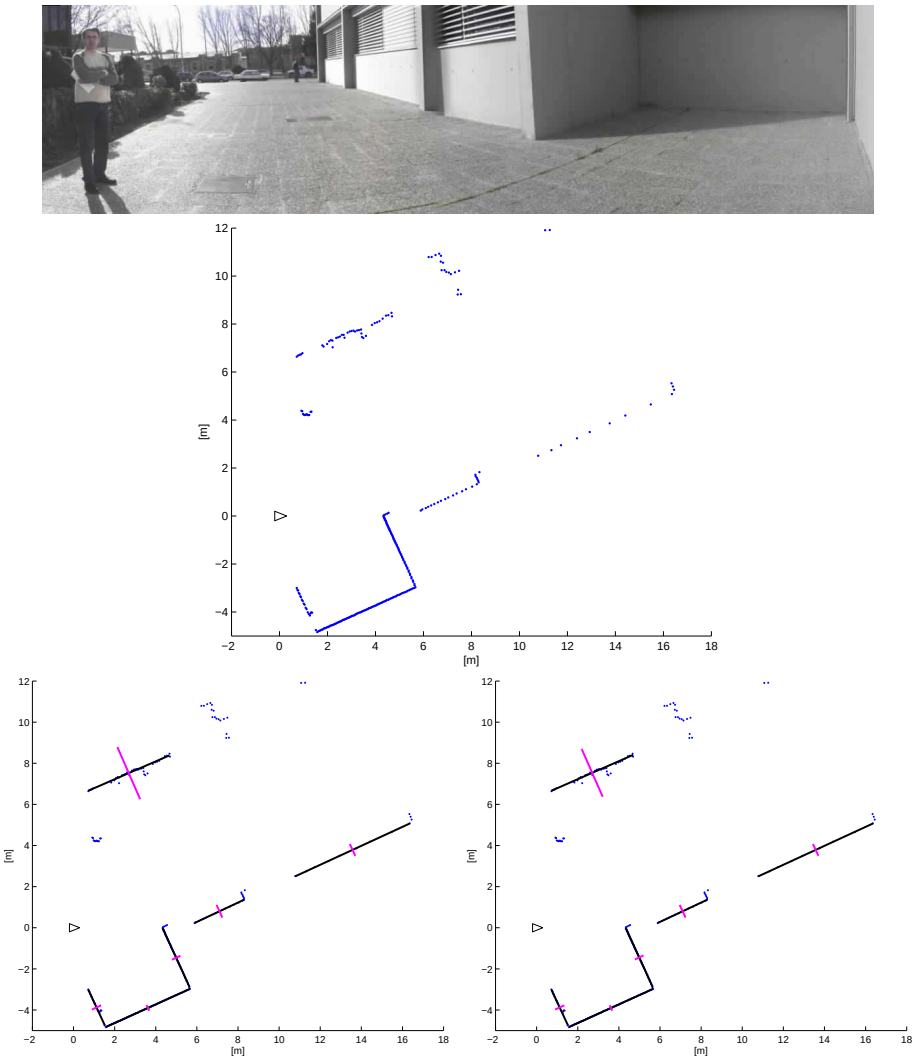


Figure 2.1: Top plot shows a panoramic photo of the sensed area; notice the person, the ivy covered wall behind him on the left, and the concrete walls on the right. In the middle, we can see the raw laser data. Bottom left plot shows the manually detected segments and bottom right shows the experimental results. The results seem indistinguishable up to the picture resolution. The uncertainty scale is plotted as a red orthogonal dash in the middle of the segments, 30σ is used to magnify the scale in order to make it visible. Note the different scales computed for the ivy covered wall and for the concrete walls.

We present an adaptation for 2D range scans of the robust algorithm, Adaptive Scale Sample Consensus, or ASSC. ASSC was introduced by Wang and Suter [Wang 04] for 3D images and fundamental matrix estimation. It combines the advantages of two classical segmentation algorithms: RANdom SAmpling Consensus (RANSAC) and Least Median Squares (LMedS) [Rousseeuw 87]. RANSAC is able to tolerate a high rate of outliers, but needs an estimate of the error scale for the clustering with a predefined scale, applied for all the clusters. On the other hand, LMedS can compute the scale of inliers from the data but can only deal with up to 50% of outliers. ASSC combines the advantages of both methods and therefore produces a data driven scale estimate per each cluster when segmenting, tolerating a high spurious rate as RANSAC.

ASSC is based on the Mean Shift [Cheng 95] search for maxima and minima in the probability density function (pdf) of the point error with respect to the straight segment defined by a cluster. The pdf is reconstructed considering the scanned points as samples of the error distribution and using the kernel method presented in section 2.3.1.

2.2 State of the art

The problem of line extraction in 2D range scans has been widely studied in the robotics literature. Several algorithms have appeared with different properties and performance [Nguyen 05]. In this section we introduce the most popular methods.

2.2.1 Line-building Algorithms

Line-building algorithms directly compute a suboptimal parametric solution of a multiple structure, exploiting the property that a single scan is an open chain of sorted points with no loops. Segmentation is done concatenating consecutive points that accomplish some heuristic line criteria. Usually, the criteria are very simple, like checking an error bound, being the fastest line detector algorithms. However, it requires a prior knowledge of inliers scale. Moreover, none probabilistic uncertainty model is assumed. Consequently, basic heuristic conditions work very well with simple pseudo-outliers, but dense environments and gross outliers require complicated rules being highly dependable on the application.

Examples of these algorithms are Split and Merge [Horowitz 76] and Incremental Line-Building [Taylor 96], also called Line-Tracking. Essentially, Split

and Merge recursively splits the original parameter model (e.g.: a straight line between the first and last point), when the maximum residue is higher than a fixed threshold. Resulting lines are thereupon merged following the same rule, i.e: the maximum residue of two collinear segments is lower than the threshold. On the other hand, the Incremental Algorithm starts with two close points (i.e: the first and second point), adding the next scan point to the end of the segment when the line criteria is validated. If the criteria is not achieved, then the current line is finished and a new line is started at the next point.

2.2.2 Hough Transform

The Hough transform [Hough 62] is based on a voting strategy to determine the *best fit* for a data subset. The main drawback of the method is that the parametric space must be discretized, consequently, the accuracy is highly affected in a real time application, since computational cost is $\mathcal{O}(nd)$ where n is the number of data points and d is the size of the voting grid. This method can be seen as a grid based maximum likelihood model selection algorithm.

The Hough Transform resembles to our approach regarding there is no prior assumption about the error bounds or the level of spurious. In addition, as a single majority voting strategy, it can intrinsically deal with both outliers and pseudo-outliers.

2.2.3 Random Sampling Segmentation Algorithms

In this case, the aim of the algorithms is to find a suboptimal probabilistic model to classifying the data points and to separate inliers from outliers.

If we consider that, given the model, every inlier error function is iid, with $X^{(i)}(\theta) \sim \mathcal{N}(0, \sigma)$, then, the inliers are taken to be those data points which normalized squared residual follows a χ^2 distribution

$$\left(\frac{X^{(i)}(\theta)}{\sigma}\right)^2 \leq \chi^2_{1-\alpha,n} \quad i = 1 \dots n \tag{2.1}$$

where $\chi^2_{1-\alpha,n}$ is a constant value for a fixed probability of false positives (i.e: $\alpha = 0.01$) and n degrees of freedom. Again, we assume that the scale estimate, or dispersion coefficient, corresponds to the estimated standard deviation σ. Therefore, a scale factor σ is mandatory to classify the data.

The clustering, or model selection step, is made by random sampling. We take some random subsets of points and compute several random parametric

models. The subsets are chosen to be minimal to compute each model. The winning model $\widehat{\theta}$ is that which maximizes an utility function U_θ. The optimal model is used to label the data using the scale parameter σ (see equation (2.1)). These algorithms can intrinsically handle multiple structures using a recursive search through the remaining data. Finally, a TLS algorithm is applied to every cluster separately in order to correct the bias.

Considering that p is the size of a subset to compute a minimal parametric model (e.g. 2 for a segment), then the number of samples to draw in order to find a subset with no outliers with probability P is [Rousseeuw 87]:

$$m = \frac{\log(1-P)}{\log[1-(1-\varepsilon)^p]} \tag{2.2}$$

where ε is the ratio of outliers.

In contrast to previously introduced algorithms the computational cost of random sampling approaches does not depend on the size of the data set but on the spurious rate, which is given by the application and not the data size. It is important to note, that these algorithms take samples in the data space, not the parametric space, like the Hough transform.

Least Median Squares (LMedS)

The median estimator is probably the most extended robust estimator due to its simplicity and efficiency when the ratio of inliers is higher than 0.5. In this case, the optimization criteria involve minimizing the median of the squared errors. Using a voting analogy, it is similar to ask for an absolute majority.

Thus, the scale estimate is given by [Rousseeuw 87]:

$$\widehat{\sigma} = 1.4826 \left(1 + \frac{5}{n-p}\right) \sqrt{med_{\widehat{\theta}} r^2} \tag{2.3}$$

where $med_{\widehat{\theta}} r^2$ is the minimum median, n is the number of samples and p is the dimension of the parameter space.

The main problem of LMedS appears in scenes with multiple models or segments, because the level of pseudo-outliers is noticeably larger than 0.5.

RANdom SAmpling Consensus: RANSAC

RANSAC assumes the inliers are the largest cluster for a predefined scale, which can be a heuristic threshold or obtained from sensor calibration. Consequently, the optimization criterium consists of maximizing the number of

inliers:

$$\widehat{\theta} = \arg\max_{\theta} n_{\theta} \tag{2.4}$$

where $\hat{\theta}$ is the parameter estimate and n_θ is the number of inliers. Provided a good scale estimate for every cluster, RANSAC is able to cope with large amounts of outliers. Nevertheless, the prior scale knowledge is sometimes unavailable or inaccurate. This *a priori* scale resembles to the Split and Merge and Incremental error bound.

Following the voting analogy, this algorithm performs several referendums and it selects the option with largest majority.

2.3 ASSC: A Kernel-Based Scale Estimator

Adaptive Scale Sample Consensus (ASSC) [Wang 04] is a modification of RANSAC involving an adaptive scale estimation. The data driven scale estimate is computed using mean shift method [Comaniciu 02]. In this case, the utility function takes into account both the number of inliers and the scale factor. Concluding the voting analogy, this algorithm merely requires a simple majority to achieve quorum.

Mean Shift is an algorithm based on kernel density estimation and kernel gradient estimation. It can be used for clustering, in this case, collinear data. Given a random sample of the parametric model, we use the residual space to identify the clusters of data. The actual inlier data is the first cluster found with an average error closer to 0. Subsequently, a LMedS algorithm is applied to the cluster in order to compute the final scale. In this case, LMedS converges to a solution because the cluster includes a single structure and few outliers.

2.3.1 Kernels for Density Approximation and Gradient Clustering

Nonparametric estimation of probability density functions [Duda 01] is a special case of kernel smoothing for regression (appendix B).

The seminal word of Parzen [Parzen 62], later extended by Cacoullos [Cacoullos 66] for the multivariate case, presented a set of multivariate kernel density esti-

mators[1] of the form:

$$\hat{p}_n(X) = (nh^d)^{-1} \sum_{i=1}^{n} k(h^{-1}(X - X^{(i)})) \tag{2.5}$$

where $X^{(1)}, \ldots, X^{(n)}$ is a set of n independent and identically distributed d-dimensional random vectors. Figure 2.2 shows a nice example of Parzen windows applied to the problem of segmentation.

Later, Fukunaga and Hostetler [Fukunaga 75], proposed an estimate of the density gradient as the gradient of the density estimate, which can be represented in terms of the differentiable kernel function:

$$\widehat{\nabla_X} p_n(X) = (nh^d)^{-1} \sum_{i=1}^{n} \nabla_X k(h^{-1}(X - X^{(i)})) \tag{2.6}$$

$$= (nh^{d+1})^{-1} \sum_{i=1}^{n} \nabla k(h^{-1}(X - X^{(i)})) \tag{2.7}$$

where ∇_X is the standard gradient with respect to the components of the density function space $X_1, \ldots, X_d$ and:

$$\nabla k(\mathbf{z}) = \left(\frac{\partial k(\mathbf{z})}{\partial z_1}, \ldots, \frac{\partial k(\mathbf{z})}{\partial z_d} \right)$$

is the gradient operator with respect to the kernel input space.

Under certain conditions, the estimate of the density gradient is unbiased and consistent [Fukunaga 75].

As can be seen in figure 2.3, the kernel gradient estimate can be used for clustering the modes of a density function.

This algorithm was generalized by Cheng [Cheng 95] to deal with different kernels and partial observation of the data.

2.3.2 Mean Shift Clustering

Armed with the kernel smoothing tools for density and gradient estimation, the Mean Shift paradigm provides a simple method to find maxima and minima of an unknown density function. It is based on the computation of the Mean Shift vector, which is basically a normalized version of the gradient:

$$M_h(X) \equiv H_h \frac{\widehat{\nabla} f(X)}{\widehat{f}(X)} \tag{2.8}$$

[1]Kernels for density estimation are sometimes called Parzen windows as a reference to the seminal work in the field

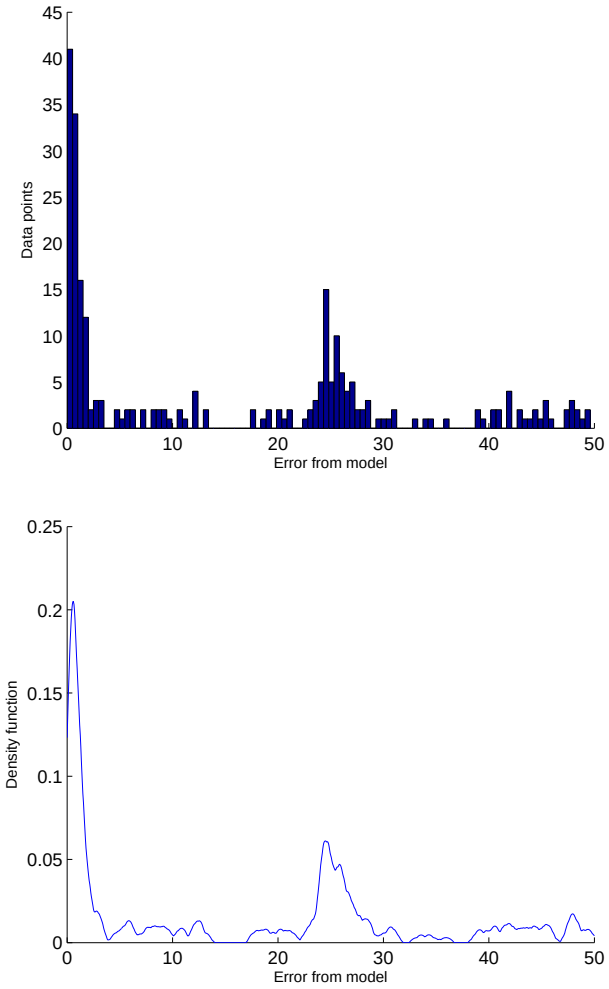


Figure 2.2: The top plot is an histogram of the data points sorted by the error to the sampled line model. It is very difficult to classify the data points based on that representation. However, using kernels or Parzen windows we can estimate the density function associated to the histogram. The bottom plot represents the smoothed function after applying the Epanechnikov kernel.

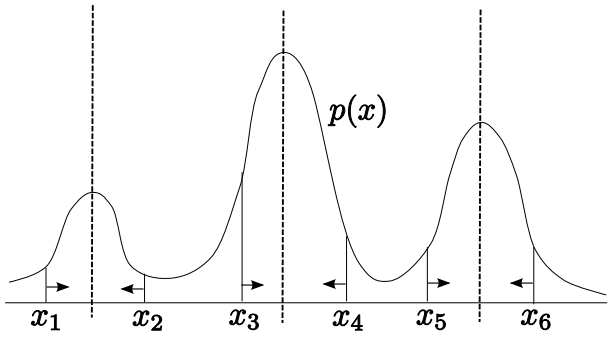


Figure 2.3: Example of mode clustering using kernel gradient information. Based on [Fukunaga 75]

where $\hat{f}(X)$ and $\widehat{\nabla} f(X)$ are the density and gradient estimated and H_h is a bandwidth coefficient related to the *amount of information* [Comaniciu 02]. It has been proved that mean shift vector points towards the direction of the maximum increase in the density. Consequently, the mode can be obtained using an iterative process:

$$X_{t+1} = X_t + M_h(X_t) \tag{2.9}$$

Given a set of samples $\mathcal{X} = \{X^{(1)}, \ldots, X^{(n)} | X^{(i)} \in \mathbb{R}^d\}$, we can compute the the kernel density estimator $\widehat{f}(x)$ and the kernel gradient estimator $\widehat{\nabla} f(x)$ using equation (2.5) and (2.6) respectively.

In our case, the random variable is the orthogonal error of a point with respect to a scene line segment, every scanned point defines a sample of the random variable. Thus, the density function $f(X)$ is a single dimensional function, defined on the positive real line, which is expected to have a large mode next to the origin representing the error of the inliers and other nodes representing other features or spurious data. In fact, if we assume that inliers are perturbed with some Gaussian noise, then, the normalized squared error or residual to the line follows a chi-square distribution.

We have selected the Epanechnikov kernel:

$$k^{Epa}(\xi) = \begin{cases} \frac{1}{2} c_d^{-1}(d+2)(1-\xi^T\xi) & \text{if } \xi^T\xi \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.10}$$

where c_d is the volume of a d-dimensional hypersphere of unit radius (i.e: $c_1 = 2, c_2 = \pi$). In our setup, being $\xi = (X - X')/h$, the kernel function is an hypersphere $S_h(X)$ defined in the sampling space centered at X with radius h

(see appendix B for a comparison of different kernels). For the Epanechnikov kernel, the amount of information H_h is:

$$H_h^e = \frac{h^2}{d+2} \tag{2.11}$$

where d is the dimension of the sample space.

Consequently, using the Epanechnikov kernel, the mean shift vector can be rewritten as follows:

$$M_h^e(X) = \frac{1}{n_e} \sum_{\mathcal{X}_e} X^{(i)} - X \tag{2.12}$$

where $\mathcal{X}_e = \{X^{(1)\cdots(n_e)} | X^{(i)} \in S_h(X)\}$ is the subset of n_e samples that fall within the hypersphere $S_h(X)$. As a result, the mean shift vector depends only on the samples and the kernel width, and it is easier to compute than the function or the gradient. It can be proved that the mean shift based optimization converges to the closest mode of the unknown density function [Wand 95], which is the desired cluster for the current parametric solution. In addition, the same method can be used to find the closest valley to the mode, following the opposite mean shift vector:

$$V_h^e(x) = -M_h^e(x) = x - \frac{1}{n_e} \sum_{\mathcal{X}_e} X_i \tag{2.13}$$

In the ASSC algorithm, this valley is used to define the bounds for the data cluster. In our case, the selected interval is $[0, x_{valley}]$, which captures the first mode from the origin which is the cluster of all data points that represent the feature model analized.

2.3.3 Adaptive Scale Sample Consensus (ASSC)

The main advantage of ASSC [Wang 04] is that the scale factor is computed for every cluster separately, instead of being a heuristic threshold.

As seen in previous section, Mean Shift methods are based on the smoothing of the sample distribution. Wide kernels produce oversmoothed functions, while narrow kernels yield peaked functions. Consequently, the performance of the mean shift vector is related to the kernel width h. For example, while searching the mode, it is interesting to slightly oversmooth the actual density function to avoid small local maxima during the gradient search. As a result, an oversmoothed bandwidth selector [Wand 95] has been chosen:

$$h = \left[\frac{4}{3n}\right]^{\frac{1}{5}} S(q) \tag{2.14}$$

where n is the total number of data points and $S(q)$ is a coarse preestimation of the standard deviation, that is, the scale of inliers. In ASSC algorithm, Wang and Suter suggested using a generalization of the median estimator based in percentiles smaller than 50% [Lee 98]:

$$S(q) = \frac{\delta_q}{\Phi^{-1}\left[\frac{1+q}{2}\right]} \tag{2.15}$$

where $q \in [0, 1]$ is the expected maximum ratio of inliers, δ_q is the half-width of the shortest window including the fraction q of total residuals and $\Phi^{-1}[\cdot]$ is the argument of the normal cumulative density function. Note that $S(0.5)$ is the median estimator. We have tested during the experiments that the performance of line extraction is not very sensitive to the q value, provided that the kernel bandwith is smaller than the size of the cluster. For example, a good policy is to use a pessimistic assumption in the ratio of inliers. Wang and Suter [Wang 04] suggest $q = 0.2$, which is analogous to rely on the 20% of data points with minimum error, for the initial scale estimation used to define h in equation (2.14).

Thereupon, mean shift is applied to find the bounds of the cluster. Finally, the actual scale of every cluster is recomputed using LMedS algorithm, taking into account only the data inside the cluster. Hence, we avoid the problem of LMedS with multiple structures.

Finally, ASSC is a random sampling algorithm (figure 2.4). Thus, we have to compute m parametric models and compare them using an utility function. For the ASSC, the utility function for the k-th parametric model $U(\theta_k)$ should be directly proportional to the number of points n_k that matches that model and inversely proportional to the dispersion of that points σ_k. Therefore, the simplest utility function is defined as:

$$U(\theta_k) = n_k/\sigma_k \tag{2.16}$$

In consequence, RANSAC is a particular case of ASSC for a fixed scale model. Thus, ASSC has the capability to handle multiple scale models even in a single scan.

Since ASSC has no restrictions about scale or cluster size, it needs a stop criteria to avoid selecting pure outlier data after having clustered all segments. For line extraction we can use some tests to find the linearity of data. For example, we know that the length of the line should be larger than the estimated noise. We have implemented a simple test based on the ratio between the segment length l_k and the estimated scale σ_k, i.e. $(l_i/\sigma_i) \geq \beta$. In the subsequent experiments, $\beta = 10$.

1. If there are still points to classify
   (a) For $k = 1 : m$
      i. Sample a line model θ_k from the data points.
      ii. Compute the resudials for the rest of the data points.
      iii. Compute the kernel size h.
      iv. Iterativelly follow the Mean Shift vector $M_h(X)$ from $X = 0$ until the peak is found. That is, $M_h(X_{peak}) = 0$.
      v. Iterativelly follow the oposite Mean Shift vector $V_h(X)$ from X_{peak} until the first valley is found. That is, $V_h(X_{valley}) = 0$.
      vi. Select the subset of n_k data points that have a residual $X^{(i)}$ less than the valley found.
      vii. Compute the LMedS of the subset $\mathcal{X}_k$ using:

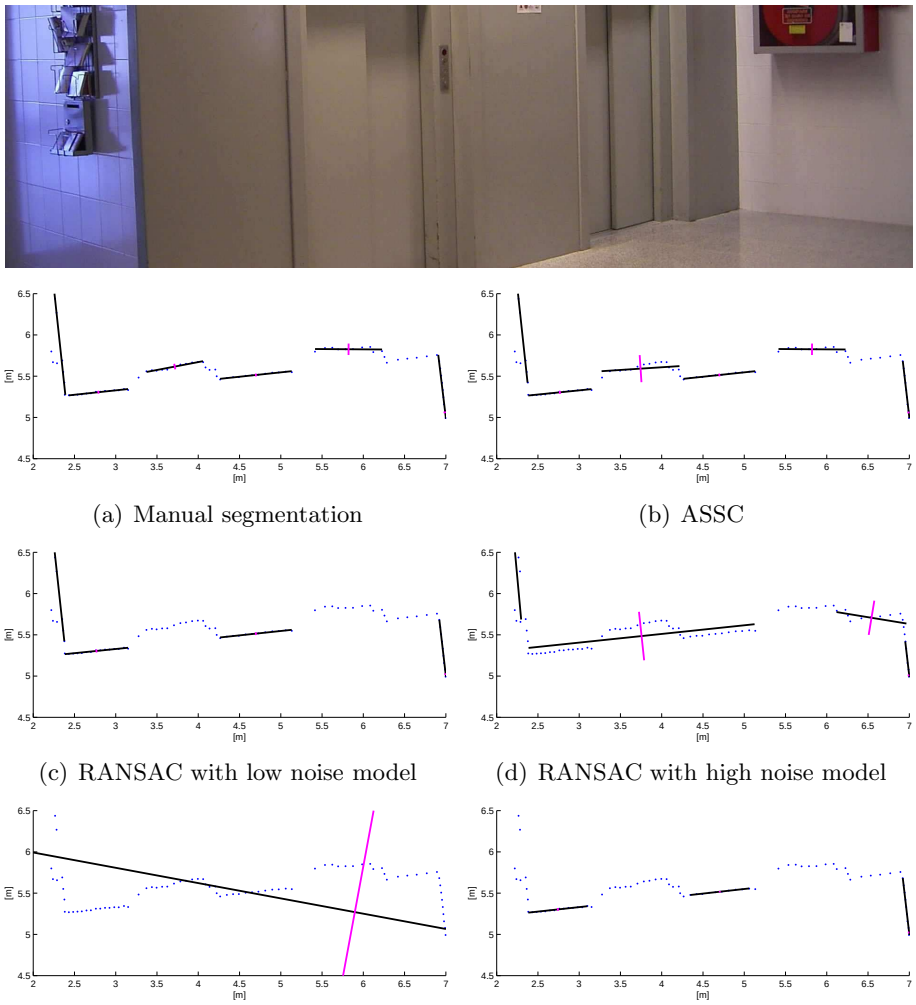
$$\sigma_k = 1.4826 \left(1 + \frac{5}{n-p}\right) \sqrt{med_{\theta_k} \mathcal{X}_k^2}$$

      viii. Define the score of the sample model as $U(\theta_k) = n_k/\sigma_k$
   (b) Select the data points included in the model that maximizes the utility function $\widehat{\theta} = \arg\max_{\theta_k} U(\theta_k)$
   (c) If data points satisfy the linearity test $(l_i/\sigma_i) \geq \beta$.
      i. Then, compute the TLS estimate of the data points included in $\widehat{\theta}$ and remove them from the data points to classify.
      ii. Else, terminate the classification.

Figure 2.4: ASSC algorithm.

2.4 Experiments

For the experimental results, a data set has been collected using a mobile platform equipped with a SICK laser range finder navigating along the Ada Byron building, at the University of Zaragoza. The robot has been driven along 600 meters indoor and outdoor manmade environment. The environment is interesting due to the presence of different walls (e.g. concrete, tiles, glass, steel). For example, an important piece of the trajectory is done next to a ivy-covered wall (figures 2.1, 2.6 and 2.7) which increases the noise level, and hence the scale, of those features. There are also some blinds and grassy slopes which provide poor and noisy reflections. The architecture of the building presents challenging staircase-shaped walls with acute and obtuse angles (figure 2.6 and 2.7). Furthermore, there are some curved elements like people,

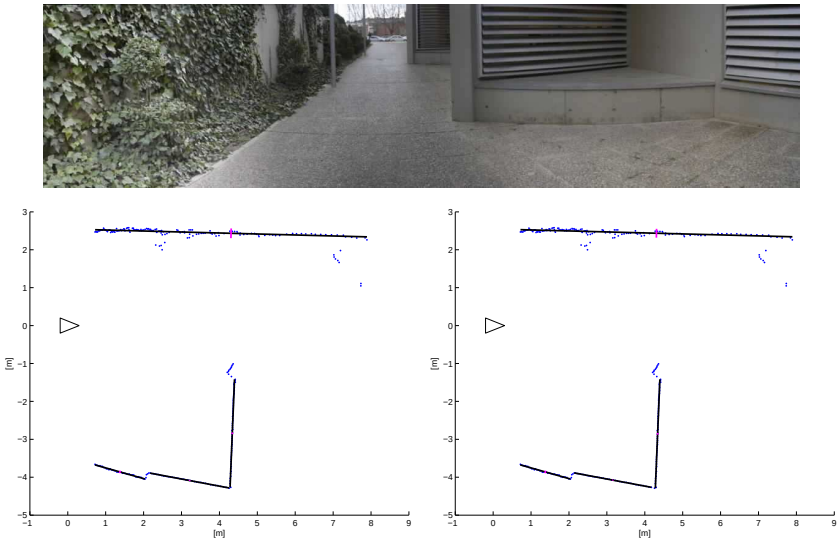


(a) Manual segmentation (b) ASSC

(c) RANSAC with low noise model (d) RANSAC with high noise model

(e) LMedS (f) Split and Merge + LMeds

Figure 2.5: Door segmentation experiment.



(a) Manual segmentation (b) ASSC

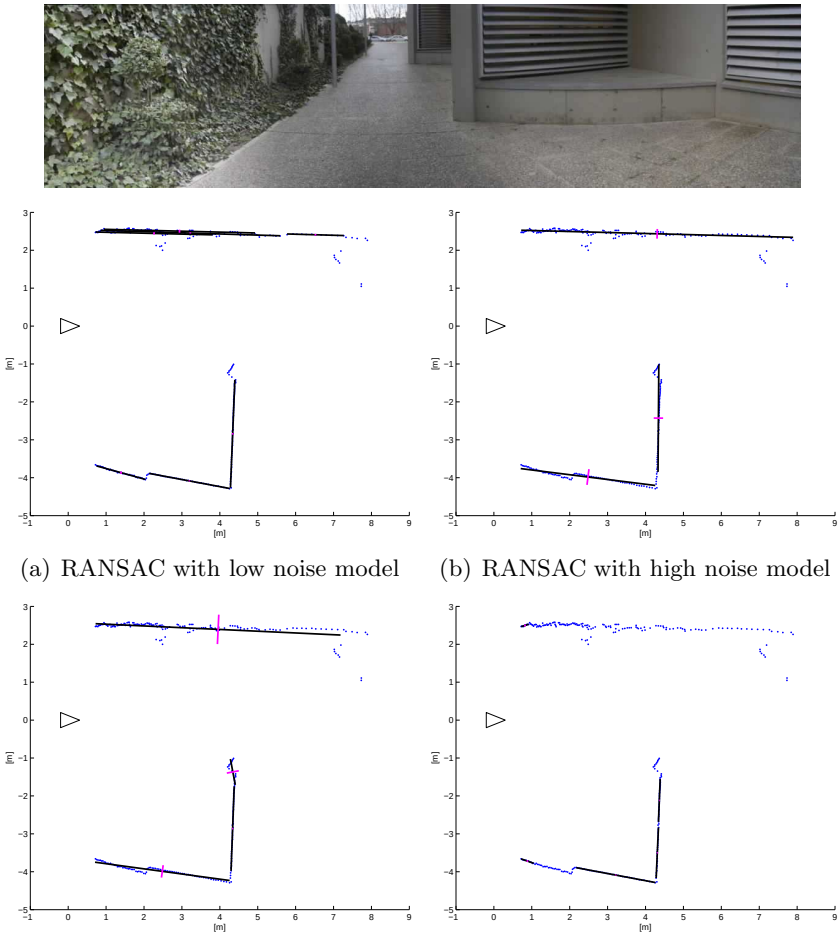
Figure 2.6: Corridor segmentation. Up wall: ivy Down walls: concrete and blinds. For ASSC, ivy wall has bigger uncertainty that the concrete wall.

baskets and decorative elements, which should be detected as outliers during segmentation.

Some random scans and some specially difficult scans have been selected to perform a manual segmentation of data points. The whole experiment consist of 38 labeled scans and 190 features. A total least squares (TLS) estimate has been computed with the labeled data as a benchmark for the algorithms.

We contrast the reliability of ASSC algorithm versus RANSAC with two different predefined scales (high and low), LMedS and a combination of Split and Merge and LMedS, as proposed in [Newman 02]. The only constrains imposed to segmentation algorithm are the minimum number of inliers $n_{min} = 10$ and the maximum distance between points to break the segment $d_{max} = 1m$. The manual benchmark takes into account these constrains. See table 2.1 for a comparison in estimate performance. If the actual segment length differs more than 10% compared to the extracted segment, we classify the segment as split or merged respectively.

ASSC offers a good compromise between all false positive, false negative, wrong split and wrong merged segments. Next in performance is RANSAC, having the problem of finding a unique predefined scale for all the segments. RANSAC with high level noise fails in the wall corners and doors, where it is



(a) RANSAC with low noise model (b) RANSAC with high noise model

(c) LMedS (d) Split and Merge + LMeds

Figure 2.7: Corridor segmentation. Up wall: ivy Down walls: concrete and blinds. For ASSC, ivy wall has bigger uncertainty that the concrete wall.

Table 2.1: Comparison of detection performance in scale algorithms.

Algorithm	False neg.	False pos.	Split segs	Merged segs
LMedS	32%	33%	38%	30%
S&M + LMedS	31%	3%	85%	0%
RANSAC high	16%	24%	19%	27%
RANSAC low	12%	11%	48%	5%
ASSC	10%	17%	19%	11%

difficult to distinguish the end of the current line and the beginning of another line; usually, one or two pseudo-outliers are included in the RANSAC clusters. On the other hand, RANSAC with low level noise oversplits noisy segments like ivy, providing an underestimate of the final segment. Finally, due to its low breaking point, LMedS algorithm is unstable in structures formed by several concatenated elements (i.e: more than 2) and Split and Merge discards or splits segments with outliers in the chain.

We have selected a specially difficult part of the experiment, which is very frequent in man made environments, to show the improvement of our approach: doors in a wall (figure 2.5). This situation is very deceptive since features are almost collinear. In addition, some intermediate points appear in the doorframe, hindering segmentation.

Usually, line extraction algorithms tend to include the doors as a part of the wall segment because the elements are collinear according to the error model or the algorithm can not deal with multiple structures. A simple approach would consist in adjusting the error model. Nevertheless, this could lead to overconfidence and inaccuracies in other parts of the environment with more texture. In contrast, ASSC is able to distinguish between texture and multiple structures, as shown in figure 2.5. Basically, our approach determines the best scale for each structure, which sometimes, is impossible to be determined previously.

Other interesting situation appears when the robot is moving along an outdoor corridor where on one side is a simple flat surface but covered with ivy and on the other side is a concrete wall, although with a complex structure (figure 2.6 and 2.7).

Figure 2.8 shows the histograms of lateral, orientation and length error distribution of the features detected in the whole experiment. It is worth noticing

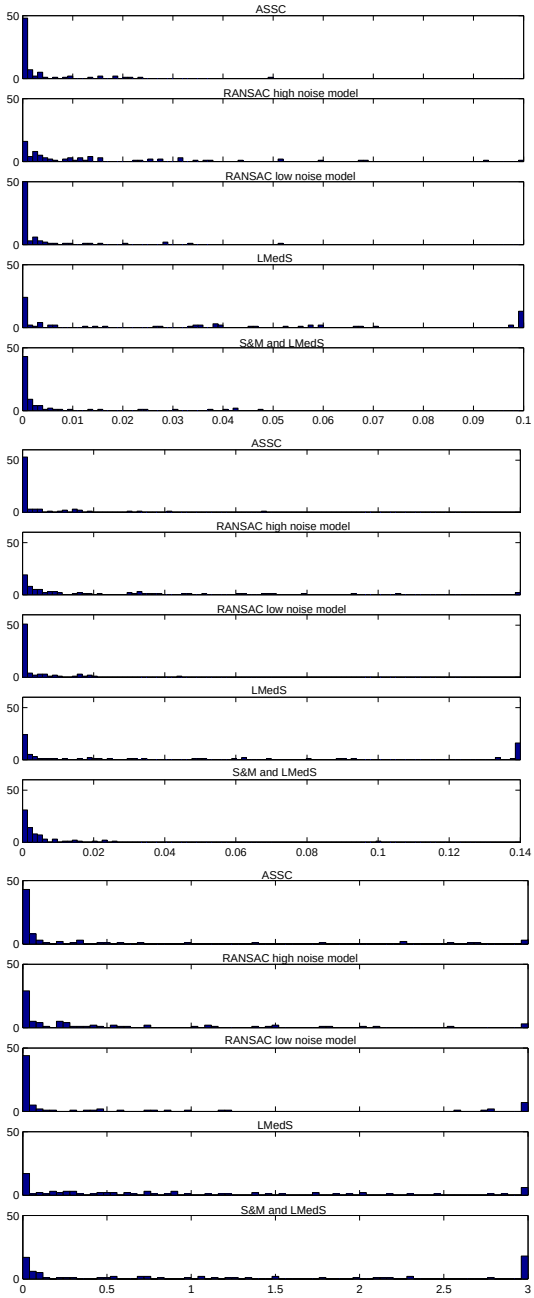


Figure 2.8: Error distributions: Top, lateral error; middle, orientation error; bottom, length error

that the histograms has been truncated to clearly represent the information. The last bin includes all error thereinafter. Our approach also improve the accuracy of classical approaches since it is able to select a tight uncertainty for every segment.

2.5 Conclusion

This chapter presents an algorithm for robust data segmentation involving multiple structures and high percentage of outliers. The main advantage is the capability to compute a data driven scale. This behavior improves, not only the segmentation but also the final accuracy. Furthermore, our approach does not require any previous calibration of the sensor or tuning of the parameters of the algorithm. This feature allows to reuse the code in different platforms, sensors and application. In addition, the algorithm has proved its robustness dealing with different objects in the same scene, for instance bumpy ivy-covered walls and flat concrete walls. Originally designed for 3D images, we have adapted and tested the ASSC algorithm for 2D range data. The method has been experimentally validated in both indoor and outdoor man-made environments for robotics applications, surpassing current algorithms performance both in estimation and detection.

Chapter 3

Gaussian-SLAM

3.1 Introduction

The Gaussian-SLAM approach is characterized by the existence of a discrete-time augmented state vector, composed of the location of the vehicle and the location of the map elements, recursively estimated from the available sensor observations gathered at time t, and a model of the vehicle motion, between time steps $t-1$ and t.

The optimal solution for the filtering distribution can be found using the celebrated Kalman filter, *even for nonstationary processes*. However, certain assumptions must be considered, i.e. (1) the latent and observable processes $\{\mathbf{x}_t\}_{t\geq 1}$, $\{\mathbf{y}_t\}_{t\geq 1}$ must be zero mean and Gaussian distributed or (2) the optimal estimate is a linear function of the observed variables and the loss function is quadratic.

We have seen in section 1.1.1 that the assumption of independent Gaussian process of primary sources $(\mathbf{v}_t, \mathbf{w}_t)$ is realistic in practice. An important property of Gaussian random signals is that linear combinations (and therefore, conditional expectations) on a Gaussian random process are Gaussian random variables. Therefore, a linear system guarantees assumption (1). But, SLAM system is nonlinear and it has been proved that the latent stochastic process does not remain Gaussianly distributed.

Assumption (2) relates the Kalman filter with the Recursive Least Squares (RLS) algorithm for linear estimation. Actually, the Kalman filter *is equivalent* to RLS for static parameter estimation. As commented in chapter 1, the Kalman filter provides an elegant solution to the joint state and parameter estimation problem. However, assumption (2) is again unrealistic for the SLAM problem. In fact, [Ljung 79] proved that the general adaptive filtering

problem is a nonlinear problem, even with a linear model. The nonlinearity comes from the implicit relationship between the parameters and the model.

A linearized framework can be developed to find a suboptimal filter similar to the Kalman filter. Due to its simplicity, the most relevant is the extended Kalman filter (EKF) [Bar-Shalom 01], which linearizes the dynamic and observation models using first-order Taylor expansion. Therefore, the Kalman filter can be straightforwardly applied to the linearized system. It can be also used to jointly estimate states and parameters.

However, higher order terms are neglected, introducing linearization errors and bias in the estimate. As discussed in [Jazwinski 70], the expected value of the neglected terms is proportional to the error variance and the second partial derivative of the nonlinear model functions that appeared in section 1.2. In that work, the author identifies two types of nonlinearity depending wether the nonlinearity is large due to the fact that the second partial derivatives are large (which is called *real nonlinearity*), or if it is large because the uncertainty is large (*induced nonlinearity*). This duality has also been studied for the camera models in [Civera 08]. As shown in the literature [Julier 01, Castellanos 04] the neglected terms frequently leads to filter divergence after only a few update steps. Deeper analysis over consistency issues in EKF-SLAM was conducted in [Bailey 06a]. In practice, the nonlinear error is only important if it is large enough compared to the actual system error. Then, the most extended approach is to *inject stabilizing noise* to reduce the importance of the linearization effect. Clearly, this method is only effective when the nonlinearity is mainly real, because the injected noise is actually inducing nonlinearity.

Within this framework, uncertainty is represented by Gaussian distributions associated to the state vector, the motion model and the sensor observations. This approach to SLAM dates back to the seminal work of Smith et al. [Smith 88] where they introduced the concept of *stochastic map* and they developed a rigorous solution to the SLAM problem using an extended Kalman filter (EKF) perspective. It is assumed that recursive propagation of the parameters of the distribtuion -the mean and the covariance- conveniently approximates the optimal solution of this estimation problem. Many successful implementations of this approach have been reported in indoor [Castellanos 99, Civera 08], outdoor [Guivant 01, Paz 07], underwater [Eustice 05, Ribas 08], air-borne [Kim 07] and planetary [Mourikis 07] applications.

The time and memory requirements of the basic EKF-SLAM approach result from the cost of maintaining the full covariance matrix, which is $\mathcal{O}(n^2)$ where n is the number of features in the map. Many recent efforts have

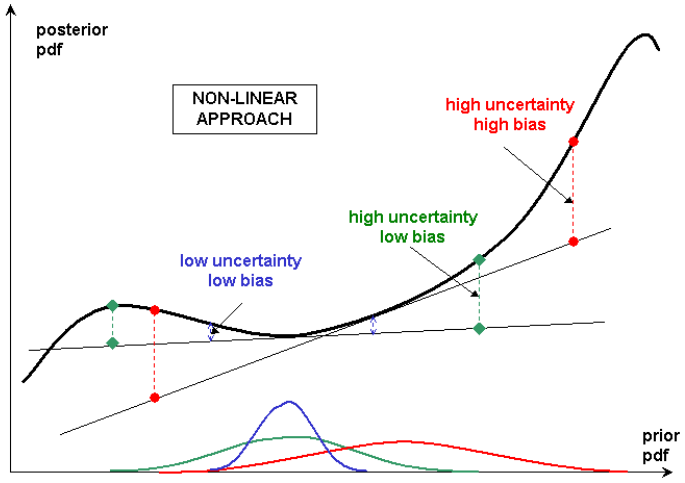


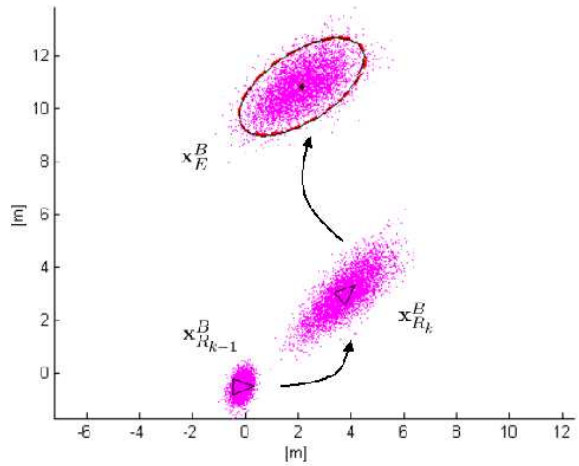
Figure 3.1: Influence of the uncertainty on linearization accuracy.

concentrated on reducing the computational complexity of EKF-SLAM[1] in large environments [Guivant 01, Tardós 02, Estrada 05, Bosse 03, Knight 01, Leonard 03, Walter 05, Paz 07, Thrun 04, Paskin 03, Frese 06]. However, only recently, the consistency issues of the EKF-SLAM algorithm have attracted the attention of the research community. They overlook the fact that, given that SLAM is a nonlinear problem, there is no guarantee that the computed covariances will match the actual estimation errors, which in practice, becomes the main consistency issue.

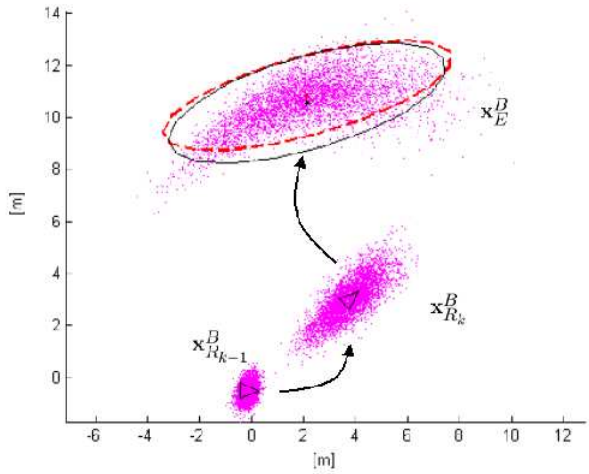
The classical EKF-SLAM linearizes both the motion and sensor models by using a first-order Taylor series expansion around the best available estimated state-vector, therefore, both the bias and the level of uncertainty in the estimated state-vector influence the accuracy of linearization. In a real application based on EKF, uncertainty must be bounded to control the approximation error as shown in figure 3.1. Additionally, figure 3.2 describe the influence of observation uncertainty on the Gaussianity assumption of the EKF approach. Clearly, the higher the uncertainty the worst the Gaussian approximation.

SLAM structure can be exploited to reduce the induced nonlinearity. Basically, global uncertainty always increase when the robot is exploring new areas,

[1]There has been also many works using the information filter, which, in general, is more computationally expensive, but it can exploit the inherent sparsity of the SLAM problem.



(a) Low observation noise



(b) High observation noise

Figure 3.2: Influence of observation uncertainty in the Gaussianity assumption for a mobile robot with uncertain location $\mathbf{x}_{R_k}^B$ observing a new landmark at $\mathbf{x}_E^B$. Dots represents Monte Carlo samples which are used as a ground truth. The ellipses are the 2σ boundaries for the posterior uncertainty computed by EKF (dashed) or Monte Carlo (solid). Clearly, for low noise levels, the EKF provides quasi-optimal estimates, but, for high noisy levels, the estimate is biased.

becoming huge when the robot has traveled long distances without revisiting. Consequently, several techniques has been proposed using small local maps, where the information is shared or combined less frequently. Also, partial observability can be exploited. Since all the sensors are mounted in the robot, all the information comes from elements near the robot, while the rest are not observed and, therefore, not included in the nonlinear equation. Thus, a robot centered representation would reduce the involved uncertainty except when a loop is closed. These approaches outperform standard approximations, also reducing the overall computational cost. In section 3.6.2 we present the *Robocentric Map Joining* approach [Castellanos 07, Martinez-Cantin 06a], whose advantages are threefold:

- It addresses the SLAM problem by building independent local maps of limited size, using the technique first proposed in [Tardós 02]. This technique was shown to greatly reduce the computational cost of SLAM. Here we show that, as the uncertainty inside a local map is bounded, the linearization errors are also reduced.

- In standard SLAM, the map is built using an absolute representation, i.e. with respect to an external reference frame B. After a certain time, the vehicle and the features currently observed have a growing absolute uncertainty that propagates to the measurement equations introducing errors. Here we build each local map using a *robot centered* representation [Castellanos 04], i.e. relative to a reference frame R attached to the vehicle. Except during loop closing, the features currently observed have an uncertainty in the order of the sensor error, which is much smaller. This also results in a reduction of the linearization errors.

- When the robot moves, its new pose is usually predicted by composing the old pose with the motion measured by odometry, which is frequently the least precise sensor. The new uncertainty is then computed by linearizing the composition around the predicted value. In *Robocentric Map Joining* we delay the composition until the map and the motion have been refined using new observations of the environment. This results in a better linearization point for the composition.

More elegant improvements over the EKF has been studied to overcome the two sources of nonlinearity. Higher order approximations has been developed to reduce the effect of real nonlinearity, e.g.: second order extended Kalman filter, sigma points Kalman filters (including the well known *unscented* Kalman

filter), iterated extended Kalman filter, etc. Those algorithms can also be used in the joint state and parameter estimation [van der Merwe 04]. In the last few years, some works have been reported which propose alternative linearization techniques in certain parts of the SLAM algorithms [Chong 99, Paskin 03, Andrade-Cetto 05].

In this chapter, we will also present a novel SLAM algorihm fully based on the unscented transformation as the linearization technique [Martinez-Cantin 05], but still following under the Gaussian SLAM paradigm. The *unscented Kalman filter* or UKF was developed by Julier and Uhlmann, see [Julier 04] and references therein. The UKF increases the accuracy of the state estimation with a comparable computational cost[2]. At time step t, the unscented filter estimates the first two moments of the underlying pdf (i.e. mean and covariance) by a linear weighted regression of evaluations of the true non-linear models at the so-called *sigma-points*. As proved in the literature, the mean and covariance of a n-dimensional Gaussian distribution propagated through a nonlinear function can be precisely and accurately approximated by a set of $\mathcal{O}(n)$ carefully selected sigma-points. UKF and other sigma-point filters use Gaussian Quadrature integration to solve the bayesian recursion integrals [Ito 00]. In this case, the Gaussian distribution is approximated with a discrete distribution with the same first and second moments. Then the discrete distribution can be easily propagated through the nonlinear function, preserving the distribution parameters. The first use of the unscented transformation in SLAM was reported in [Chong 99] where preliminary work on a small-scale indoor environment was considered. More recently, Andrade-Cetto et al., [Andrade-Cetto 05] describe the application of the *unscented transformation* to the vehicle movement model by assuming linearity both in the prediction of the map features and the update of the full state vector. In addition, Paskin [Paskin 03] uses the unscented transform to preprocess the feature observations, but the rest of the algorithm use EKF-type linearizations.

It is important to note that the UKF and other sigma points methods are not actual sampling methods, like Monte Carlo methods [Doucet 01]. Sigma point filters are still parametric methods to approximate the posterior or filtering distribution, while particle filters use Monte Carlo integration and the actual samples as the approximated distribution. Monte Carlo based approaches for SLAM [Doucet 00, Montemerlo 03b] will be discussed in chapter 4.

In practice, model nonlinearity, either real or induced, comes from three factors:

[2]For a n-dimensional state estimation problem, naive UKF is $\mathcal{O}(n^3)$, but square root UKF [van der Merwe 01] can achieve $\mathcal{O}(n^2)$ updates

1. **Robot heading**: The dynamic system of a robot equipped with a perfect compass and simple motion constrains like differential drive or constant velocity can be represented as a linear function. For 3D motion, the compass should provide measures for the 3 angles (roll, pitch and yaw).

2. **Sensor bearing**: Assuming an on-board range and/or bearing sensor, the model for feature initialization is linear if the sensor has perfect bearing information.

3. **Parallax**: As shown in [Civera 08] the linearity of the observation equation with an on-board sensor is proportional to the parallax, that is, the change of angular position of two observations of the same feature.

Those factors are related with the three models in feature based SLAM, which are, robot motion, feature initialization and feature update.

Cameras are good bearing sensors. Consequently, it is not surprising that even low resolution cameras mapping distant objects can be used as an accurate compass [Montiel 06]. Using cameras, inverse depth representation of the features provide an almost linear initialization. Also, the update step is linear provided that either low parallax or good bearing information is obtained [Civera 08]. Thus, for environments where distant objects are observed, quasi-linear performance can be obtained for Gaussian SLAM using cameras. But the algorithms presented in this thesis are intended to be more general. Therefore, they have to be applicable in different sensors and platforms.

From a practical point of view, in SLAM, one of the most significant factors that jeopardize the results of any SLAM algorithm are the mismatches between observations and map features due to inconsistent estimation of location uncertainty. The increased accuracy in the computation of the mean and covariance of the state distribution suggests a reduction in the ambiguity of data association, hence, low-complexity validation gate approaches could reliably and robustly be utilized.

3.2 Gaussian state-space SLAM

In the Gaussian state-space formulation of SLAM, the vehicle R and a set of environment features $\mathcal{F} = \{F_1, \ldots, F_n\}$ are represented by a normally distributed state vector $\mathbf{x}^B$ with estimated mean $\hat{\mathbf{x}}^B$ and estimated error covariance $\mathbf{P}^B$:

$$\hat{\mathbf{x}}^B = \begin{bmatrix} \hat{\mathbf{x}}_R^B \\ \hat{\mathbf{x}}_{\mathcal{F}}^B \end{bmatrix} \; ; \; \mathbf{P}^B = \begin{bmatrix} \mathbf{P}_R^B & \mathbf{P}_{R\mathcal{F}}^B \\ \mathbf{P}_{\mathcal{F}R}^B & \mathbf{P}_{\mathcal{F}}^B \end{bmatrix} \tag{3.1}$$

where $\hat{\mathbf{x}}_R^B$ is the estimated location of the vehicle with respect to (w.r.t.) a base reference frame B, $\hat{\mathbf{x}}_{\mathcal{F}}^B$ is the estimated location of the features also w.r.t. B, $\mathbf{P}_R^B$ is the estimated error covariance of the location of R, $\mathbf{P}_{\mathcal{F}}^B$ is the estimated error covariance of the location of the features, and finally, $\mathbf{P}_{R\mathcal{F}}^B$ represents the cross-covariance between the different elements of the state vector. Additionally, it is generally assumed that the underlaying probability density function is Gaussian, hence, at time step k, $\mathbf{x}_k^B \sim \mathcal{N}(\hat{\mathbf{x}}_k^B, \mathbf{P}_k^B)$.

When the vehicle moves from position at step $t-1$ to position at step t, the stochastic state vector changes according to the nonlinear motion equation:

$$\mathbf{x}_t^B = \mathbf{f}_t(\mathbf{x}_{t-1}^B, \mathbf{x}_{R_t}^{R_{t-1}}) \tag{3.2}$$

where the uncertain relative motion $\mathbf{x}_{R_t}^{R_{t-1}}$ is estimated by odometry and assumed to be white Gaussian.

From a Bayesian point of view, suppose that the stochastic map $\mathbf{x}_{t-1}^B \sim \mathcal{N}(\hat{\mathbf{x}}_{t-1}^B, \mathbf{P}_{t-1}^B)$ is available at time $t-1$, then, the predicted stochastic map at time t results from:

$$\begin{aligned} \hat{\mathbf{x}}_{t|t-1}^B &= \mathbb{E}[\mathbf{f}_t(\mathbf{x}_{t-1}^B, \mathbf{x}_{R_t}^{R_{t-1}})] \\ \mathbf{P}_{t|t-1} &= \mathbb{E}[(\mathbf{x}_t^B - \hat{\mathbf{x}}_{t|t-1}^B)(\mathbf{x}_t^B - \hat{\mathbf{x}}_{t|t-1}^B)^T] \end{aligned} \tag{3.3}$$

On-board sensors provide, at time t, the observation $\mathbf{y}_t$ related to the state vector $\mathbf{x}_t^B$ by the nonlinear measurement equation:

$$\mathbf{y}_t = \mathbf{h}_t(\mathbf{x}_t^B, \mathbf{x}_{\mathcal{E}_t}^{R_t}) \tag{3.4}$$

where $\mathbf{x}_{\mathcal{E}_t}^{R_t}$ represents the set of white Gaussian gathered observations w.r.t. the reference frame R_t. This new information about the state vector, can be incorporated into the state by using the update equations of a linear (in the measurements) estimator:

$$\begin{aligned} \hat{\mathbf{x}}_t^B &= \hat{\mathbf{x}}_{t|t-1}^B + \mathbf{P}_{\mathbf{xy}}\mathbf{P}_{\mathbf{yy}}^{-1}(\mathbf{y}_t - \hat{\mathbf{y}}_t) \\ \mathbf{P}_t^B &= \mathbf{P}_{t|t-1}^B - \mathbf{P}_{\mathbf{xy}}\mathbf{P}_{\mathbf{yy}}^{-1}\mathbf{P}_{\mathbf{yx}} \end{aligned} \tag{3.5}$$

with:

$$\mathbf{P}_{\mathbf{yy}} = \mathbb{E}[(\mathbf{y}_t - \hat{\mathbf{y}}_t)(\mathbf{y}_t - \hat{\mathbf{y}}_t)^T] \tag{3.6}$$

$$\mathbf{P}_{\mathbf{xy}} = \mathbb{E}[(\mathbf{x}_t^B - \hat{\mathbf{x}}_{t|t-1}^B)(\mathbf{y}_t - \hat{\mathbf{y}}_t)^T] \tag{3.7}$$

If a reference external to the vehicle is used as base reference, the vehicle location must be initialized with the corresponding nonzero uncertainty. A common misconception is that this nonzero initial level of uncertainty in the vehicle location may improve map consistency. In contrast, our experiments will show that this quickly results in optimistic covariance values due to linearization errors. For this reason, we use the vehicle location before the first observation (at step $t = 0$) as the base reference ($B = R_0$). Thus, the map can be initialized with zero covariance for the vehicle location: $\hat{\mathbf{x}}_0^B = (0,\, 0,\, 0)^T$, $\mathbf{P}_0^B = \mathbf{0}$. Our results show that this improves the consistency of the EKF-SLAM algorithm.

3.3 The Curse of Dimensionality in Gaussian SLAM

Intuitively, uncertainty can be seen as the volume of the state space that we are interested in. The *curse of dimensionality* is a term to describe the problem caused by the exponential increase in volume associated with adding extra dimensions to a mathematical space. Therefore, adding new dimensions implies and exponential growth of the uncertainty. In SLAM, there are two sources of dimensionality addition: temporal and spatial.

Every time step, the Markov chain is increased with the corresponding new states and observations. However, the mean and the covariance of a Gaussian are sufficient statistics to represent the current state. Therefore, we can apply the Markov property and predict the future state based on the marginal distribution. In contrast, some SLAM algorithms use incremental smoothing, with nonlinear optimization to improve the overall consistency or to exploit the sparsity of the problem. In that case, the solution depends on the whole posterior distribution, suffering from the temporal curse of dimensionality.

On the other hand, it is impossible to known a priori the number of landmarks that a robot is going to see while exploring an unknown environment. Then, every feature-based SLAM algorithm needs to consider the addition of previously unobserved landmarks. Every feature added to the state vector increase the dimensionality of the problem, consequently, the uncertainty grows exponentially and so does the induced nonlinearity.

It is important to remark that the spatial dimension only grows when the robot is exploring new areas. In contrast, the temporal dimension is always increasing, no matter if the robot is exploring or revisiting known areas.

3.4 Linearizations in the Classical EKF-SLAM Algorithm

The classical EKF-based SLAM approach computes the estimated covariance matrices of equations (3.3) and (3.6) by first-order *analytical* linearization of the motion and measurement models respectively.

3.4.1 The prediction step

When the vehicle moves from position at time step $t-1$ to position at time step t, its location is predicted as follows:

$$\begin{aligned} \mathbf{x}^B_{R_{t|t-1}} &= \mathbf{f}_t(\mathbf{x}^B_{R_{t-1}}, \mathbf{x}^{R_{t-1}}_{R_t}) \\ &= \mathbf{x}^B_{R_{t-1}} \oplus \mathbf{x}^{R_{t-1}}_{R_t} \end{aligned} \tag{3.8}$$

where the uncertain displacement $\mathbf{x}^{R_{t-1}}_{R_t}$ is estimated by odometry and assumed to be corrupted by zero mean white Gaussian noise, $\mathbf{v}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_t)$. Note that due to the transformation composition $\oplus$, a nonlinear prediction model is formulated. Thus, a *first* linearization, around the estimated values $\hat{\mathbf{x}}^B_{R_{t-1}}$ and $\hat{\mathbf{x}}^{R_{t-1}}_{R_t}$ using the appropriate Jacobians (see appendix A), is required:

$$\begin{aligned} \hat{\mathbf{x}}^B_{t|t-1} &= \begin{bmatrix} \hat{\mathbf{x}}^B_{R_{t-1}} \oplus \hat{\mathbf{x}}^{R_{t-1}}_{R_t} \\ \hat{\mathbf{x}}^B_{F_{1,t-1}} \\ \vdots \\ \hat{\mathbf{x}}^B_{F_{n,t-1}} \end{bmatrix} \\ \mathbf{P}^B_{t|t-1} &\simeq \mathbf{J}_1 \mathbf{P}^B_{t-1} \mathbf{J}_1^T + \mathbf{J}_2 \mathbf{Q}_k \mathbf{J}_2^T \end{aligned} \tag{3.9}$$

where

$$\mathbf{J}_1 = \begin{bmatrix} \mathbf{J}_{1\oplus}\left\{\hat{\mathbf{x}}^B_{R_{t-1}}, \hat{\mathbf{x}}^{R_{t-1}}_{R_t}\right\} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & & \vdots \\ \vdots & & \ddots & \\ \mathbf{0} & \cdots & & \mathbf{I} \end{bmatrix}, \quad \mathbf{J}_2 = \begin{bmatrix} \mathbf{J}_{2\oplus}\left\{\hat{\mathbf{x}}^B_{R_{t-1}}, \hat{\mathbf{x}}^{R_{t-1}}_{R_t}\right\} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix}$$

and $\mathbf{J}_{1\oplus}$ and $\mathbf{J}_{2\oplus}$ being the Jacobians of transformation composition (appendix A).

3.4.2 The update step

At step t an onboard sensor obtains a partial measurement $\mathbf{y}_t$ of the environment features and related to the state by a nonlinear function $\mathbf{h}_t$:

$$\mathbf{y}_t = \mathbf{h}_t(\mathbf{x}_t^B, \mathbf{x}_{\mathcal{E}_t}^{R_t}) \tag{3.10}$$

where $\mathbf{x}_{\mathcal{E}_t}^{R_t}$ represents the set of uncertain observations with respect to R_t, and corrupted by zero mean white Gaussian noise, $\mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_t)$.

A *second* linearization, this time around the current map prediction $\hat{\mathbf{x}}_{t|t-1}^B$, yields:

$$\begin{aligned}
\mathbf{y}_t &\simeq \mathbf{h}_t(\hat{\mathbf{x}}_{t|t-1}^B, \hat{\mathbf{x}}_{\mathcal{E}_t}^{R_t}) + \mathbf{H}_t(\mathbf{x}_t^B - \hat{\mathbf{x}}_{t|t-1}^B) + \mathbf{G}_t(\mathbf{x}_{\mathcal{E}_t}^{R_t} - \hat{\mathbf{x}}_{\mathcal{E}_t}^{R_t}) \\
\mathbf{H}_t &= \left.\frac{\partial \mathbf{h}_t}{\partial \mathbf{x}_t^B}\right|_{(\hat{\mathbf{x}}_{t|t-1}^B, \hat{\mathbf{x}}_{\mathcal{E}_t}^{R_t})} = \begin{bmatrix} \mathbf{H}_{R_t} & \mathbf{0} & \cdots & \mathbf{H}_{F_t} \cdots & \mathbf{0} \end{bmatrix} \\
\mathbf{H}_{R_t} &= \left.\frac{\partial \mathbf{h}_t}{\partial \mathbf{x}_{R_t}^B}\right|_{(\hat{\mathbf{x}}_{t|t-1}^B, \hat{\mathbf{x}}_{\mathcal{E}_t}^{R_t})} ; \quad \mathbf{H}_{F_t} = \left.\frac{\partial \mathbf{h}_t}{\partial \mathbf{x}_{F_t}^B}\right|_{(\hat{\mathbf{x}}_{t|t-1}^B, \hat{\mathbf{x}}_{\mathcal{E}_t}^{R_t})} \\
\mathbf{G}_t &= \left.\frac{\partial \mathbf{h}_t}{\partial \mathbf{x}_{\mathcal{E}_t}^{R_t}}\right|_{(\hat{\mathbf{x}}_{t|t-1}^B, \hat{\mathbf{x}}_{\mathcal{E}_t}^{R_t})}
\end{aligned} \tag{3.11}$$

Measurement $\mathbf{y}_t$ is used to obtain a new estimation of the state using the standard EKF update equations:

$$\begin{aligned}
\hat{\mathbf{x}}_t^B &= \hat{\mathbf{x}}_{t|t-1}^B + \mathbf{K}_t \nu_t \\
\mathbf{P}_t^B &= (\mathbf{I} - \mathbf{K}_t \mathbf{H}_t)\mathbf{P}_{t|t-1}^B \\
\mathbf{K}_t &= \mathbf{P}_{t|t-1}^B \mathbf{H}_t^T (\mathbf{H}_t \mathbf{P}_{t|t-1}^B \mathbf{H}_t^T + \mathbf{G}_t \mathbf{R}_t \mathbf{G}_t^T)^{-1}
\end{aligned} \tag{3.12}$$

where $\nu_t = \mathbf{y}_t - \mathbf{h}_t(\hat{\mathbf{x}}_{t|t-1}^B, \hat{\mathbf{x}}_{\mathcal{E}_t}^{R_t})$ is the *innovation* of the filter, with covariance matrix $\mathbf{S}_t = \mathbf{H}_t \mathbf{P}_{t|t-1}^B \mathbf{H}_t^T + \mathbf{G}_t \mathbf{R}_t \mathbf{G}_t^T$, that is, $p(\mathbf{y}_t | \mathbf{y}_{1:t-1}) = \mathcal{N}(\nu_t, \mathbf{S}_t)$.

3.5 The Inconsistency of EKF-SLAM

Let $\hat{\mathbf{x}}_t^B$ and $\mathbf{P}_t^B$ be the first two moments of the SLAM state estimated at time t. The state estimator is called *consistent* [Bar-Shalom 01] if its state

estimation error $\mathbf{x}_t^B - \hat{\mathbf{x}}_t^B$ is *unbiased*, i.e. $\mathbb{E}\left[\mathbf{x}_t^B - \hat{\mathbf{x}}_t^B\right] = \mathbf{0}$ and the actual Mean Square Error matches the filter calculated covariances:

$$\mathbb{E}\left[\left(\mathbf{x}_t^B - \hat{\mathbf{x}}_t^B\right)\left(\mathbf{x}_t^B - \hat{\mathbf{x}}_t^B\right)^T\right] = \mathbf{P}_t^B \tag{3.13}$$

Whenever ground-truth for the state variables is available, a statistical test for filter consistency can be carried out on the normalized estimation error squared (NEES):

$$\text{NEES} = \left(\mathbf{x}_t^B - \hat{\mathbf{x}}_t^B\right)^T \left(\mathbf{P}_t^B\right)^{-1} \left(\mathbf{x}_t^B - \hat{\mathbf{x}}_t^B\right) \leq \chi^2_{r,1-\alpha} \tag{3.14}$$

where $\chi^2_{r,1-\alpha}$ is a threshold obtained from the χ^2 distribution with $r = dim(\mathbf{x}_t^B)$ degrees of freedom, and α the desired significance level (usually 0.05).

Unfortunately, ground truth for the state variables is not directly available except in some controlled simulation experiments. However, a statistical test for real-time consistency can still be carried out, in this case, on the normalized innovation squared (NIS):

$$\text{NIS} = \nu_t^T \mathbf{S}_t^{-1} \nu_t \leq \chi^2_{r,1-\alpha} \tag{3.15}$$

where $r = dim(\nu_t)$.

In practice, this test is used to solve the lack of generative model. It can decide wether an observation corresponds to a novel feature or an existing one. In case of ambiguous data associations, the method selects the one with lower NIS. Thus, one of the most critical factors that jeopardize the consistency of any SLAM algorithm are the incorrect data associations between observations and map features.

3.5.1 Empirical proof of EKF-SLAM inconsistency

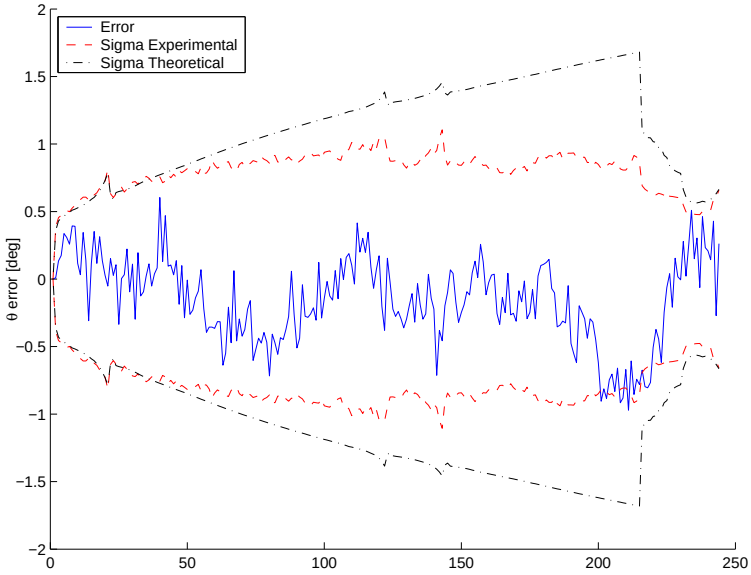
To isolate the effects of linearization errors on the consistency of the EKF-based approach to SLAM, we have designed a simulated experiment with known data association. The vehicle travels along a rectangular-shaped trajectory of 100 × 20 meters, i.e. a 240-meter loop trajectory, moving 1m per step. The map of the navigation environment is composed of 2-D point features, located at both sides of the vehicle trajectory with a feature density of 0.5 feature/m. The vehicle is equipped with a range-bearing sensor with a maximum range of 15 meters and a 180 degrees frontal field-of-view. Gaussian-distributed synthetic errors were generated for both the sensor measurements (standard deviation of 5 cm per m in range and 0.5 deg in orientation) and for

the odometry model of the vehicle (standard deviations of 0.2 m per m in displacement and 0.5 deg in orientation). We have run a Monte Carlo simulation with 20 replications.

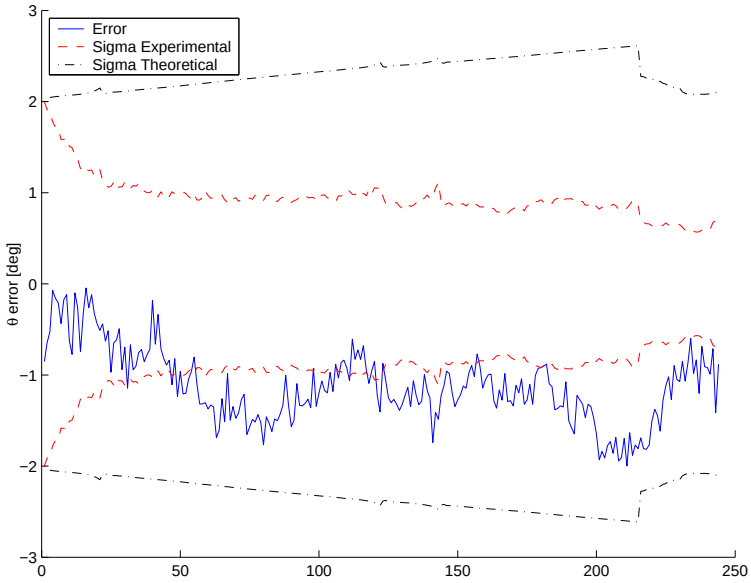
Figure 3.3, top, shows the evolution of angular error and uncertainty (2σ bounds) in the vehicle location along the trajectory for a representative replication of the experiment. For this SLAM simulation, the initial vehicle location is used as base reference, allowing to set the initial vehicle uncertainty to zero. The theoretical uncertainty level was obtained by simulating the same trajectory linearizing around ground truth (simulated with noise = 0), so that there are no linearization errors. We can see that, while the theoretical angular uncertainty increases until loop closing, the uncertainty computed by the EKF saturates reaching a maximum level (around 0.5 deg in this case). This results in the vehicle location estimation failing the consistency check of equation (3.14) after only 100m. Additionally, the average of the 20 replications resulted in biased estimation for frontal, lateral and angular errors.

From the experimental experience gained from EKF-SLAM mapping an important conclusion can be derived: The consistency of the EKF-SLAM algorithm greatly depends on the level of uncertainty of the state vector, the higher the uncertainty, especially heading uncertainty, the worst the consistency of the estimates.

It is common practice to build a map relative to a fixed base reference, different from the initial vehicle location. This normally requires to assign an initial level of uncertainty to the vehicle estimated location. As it is argued in [Dissanayake 01], the vehicle uncertainty should always be above this initial level. Surprisingly, our simulations shows that when a non-zero initial uncertainty is used (figure 3.3, bottom), the estimated vehicle uncertainty rapidly drops below its initial value (1 deg) making the estimation inconsistent after only 50 EKF update steps. This corroborates the results of [Julier 01], but also shows that the problem arises in practice earlier than they suggested. It is important to note that this result is only due to nonlinearities. It has been proved that the relative entropy of a wrongly initialized filter in a Markov process with respect to the optimal filter is a supermartingale [Clark 99], which means that the expectation of the relative entropy can only decrease with time. In other words, the filtering distribution of both filters should became similar, which is not the case here.



(a)



(b)

Figure 3.3: Angular error and 2σ uncertainty bounds of the vehicle estimated location for the cases of zero (a) and nonzero (b) initial uncertainty.

3.6 Improving the consistency of EKF-SLAM

Previous sections have shown that consistency is a matter of the problem structure, not only the algorithm. In this section we will show how the structure of the SLAM can be exploited to improve the consistency of the naive EKF-SLAM algorithm.

3.6.1 Robocentric Mapping

In the robocentric filter, we formulate the EKF-SLAM problem using the reference frame attached to the vehicle R as base reference of the stochastic map. Thus, the environmental information $\{R, B, F_1, \ldots, F_n\}$ is represented by a stochastic state vector $\mathbf{x}^R$ with estimated mean $\hat{\mathbf{x}}^R$ and estimated error covariance $\mathbf{P}^R$:

$$\hat{\mathbf{x}}^R = \begin{bmatrix} \hat{\mathbf{x}}^R_B \\ \hat{\mathbf{x}}^R_{F_1} \\ \vdots \\ \hat{\mathbf{x}}^R_{F_n} \end{bmatrix} ; \; \mathbf{P}^R = \begin{bmatrix} \mathbf{P}^R_B & \cdots & \mathbf{P}^R_{BF_n} \\ \vdots & \ddots & \vdots \\ \mathbf{P}^R_{F_nB} & \cdots & \mathbf{P}^R_{F_n} \end{bmatrix} \tag{3.16}$$

where the world reference frame B has been included as a non-observable feature in the stochastic state vector. This has the purpose of allowing to recover the equivalent absolute map if desired, but it is not necesary for the filtering algorithm, neither for navigation. With the purpose of avoiding the inconsistency problem related to non-zero initial uncertainty described above, we take the initial vehicle location as base reference $B = R_0$, and thus at step $t = 0$ the map is initialized with perfect knowledge of the world location: $\hat{\mathbf{x}}^R_0 = (0, 0, 0)^T$ and $\mathbf{P}^R_0 = \mathbf{0}$.

In robocentric mapping, each filter iteration includes three steps: prediction, update and composition, which are detailed next.

The prediction step

After the vehicle changes its location from step $t-1$ to step t, the complete structure of the stochastic map should be affected by the process noise associated with the displacement $\mathbf{x}^{R_{t-1}}_{R_t}$ as estimated by odometry, and with covariance matrix $\mathbf{Q}_t$. Thus, the estimated location of a given map feature F should be updated as:

$$\mathbf{x}^{R_t}_{F_{t|t-1}} = \ominus \mathbf{x}^{R_{t-1}}_{R_t} \oplus \mathbf{x}^{R_{t-1}}_{F_{t-1}} \tag{3.17}$$

and therefore, its estimated covariance would be computed from the corresponding linearization around the estimated values $\hat{\mathbf{x}}_{R_t}^{R_{t-1}}$ and $\hat{\mathbf{x}}_{F_{t-1}}^{R_{t-1}}$. As odometry is the least precise component in the system, this linearization can introduce significant errors. Instead, we propose to delay the composition in equation (3.17) until the estimated vehicle motion has been improved by the update step of the EKF algorithm. Conceptually, this idea resembles the optimal proposal approximation in particle filters, where the last observation is *somehow* included in the proposal to reduce the approximation error.

Therefore, in the prediction step the vehicle motion $\hat{\mathbf{x}}_{R_t}^{R_{t-1}}$ obtained by odometry is simply added, as an independent feature, to the previously available stochastic map $\mathbf{x}_{t-1}^{R_{t-1}}$:

$$\hat{\mathbf{x}}_{t|t-1}^{R_{t-1}} = \begin{bmatrix} \hat{\mathbf{x}}_{t-1}^{R_{t-1}} \\ \hat{\mathbf{x}}_{R_t}^{R_{t-1}} \end{bmatrix}; \; \mathbf{P}_{t|t-1}^{R_{t-1}} = \begin{bmatrix} \mathbf{P}_{t-1}^{R_{t-1}} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_t \end{bmatrix} \tag{3.18}$$

The update step

Now, linearization of the measurement equation around the estimated values of both the stochastic state vector and the partial measurement $\mathbf{y}_t$ yields:

$$\begin{aligned} \mathbf{y}_t &\simeq \mathbf{h}_t(\hat{\mathbf{x}}_{t|t-1}^{R_{t-1}}) + \mathbf{H}_t(\mathbf{x}_t^{R_{t-1}} - \hat{\mathbf{x}}_{t|t-1}^{R_{t-1}}) \\ \mathbf{H}_t &= \left.\frac{\partial \mathbf{h}_t}{\partial \mathbf{x}_t^{R_{t-1}}}\right|_{\hat{\mathbf{x}}_{t|t-1}^{R_{t-1}}} = [\mathbf{0} \; \cdots \; \mathbf{0} \; \mathbf{H}_{F_t} \; \mathbf{0} \; \cdots \; \mathbf{0} \; \mathbf{H}_{R_t}] \end{aligned} \tag{3.19}$$

where:

$$\mathbf{H}_{F_t} = \left.\frac{\partial \mathbf{h}_t}{\partial \mathbf{x}_{F_t}^{R_{t-1}}}\right|_{\hat{\mathbf{x}}_{t|t-1}^{R_{t-1}}}; \; \mathbf{H}_{R_t} = \left.\frac{\partial \mathbf{h}_t}{\partial \mathbf{x}_{R_t}^{R_{t-1}}}\right|_{\hat{\mathbf{x}}_{t|t-1}^{R_{t-1}}}$$

Equations which are subsequently used to obtain a new estimation of the stochastic state vector $\hat{\mathbf{x}}_{t|t}^{R_{t-1}}$ and its covariance matrix $\mathbf{P}_{t|t}^{R_{t-1}}$, using the previously described EKF update equations. Note that, because the relative displacement of the vehicle from time $t-1$ to time t was included as a feature of the stochastic state vector, it is also refined during the application of the update equations.

The use of a robot centered representation greatly influences the internal structure of the measurement equation (3.19) is comparison with the measurement equation (3.11) obtained by using an absolute representation of the stochastic map. Precisely, and except for loop closing, the uncertainty of the

filter innovation is greatly reduced down to the level of the observation uncertainty, thus, improving the accuracy of the linearization.

The composition step

As a final step in the robocentric mapping algorithm, the stochastic state vector of the robocentric map is obtained by affecting each estimated location by the improved vehicle motion:

$$\hat{\mathbf{x}}_t^{R_t} = \begin{bmatrix} \ominus\hat{\mathbf{x}}_{R_t}^{R_{t-1}} \oplus \hat{\mathbf{x}}_B^{R_{t-1}} \\ \ominus\hat{\mathbf{x}}_{R_t}^{R_{t-1}} \oplus \hat{\mathbf{x}}_{F_1}^{R_{t-1}} \\ \vdots \\ \ominus\hat{\mathbf{x}}_{R_t}^{R_{t-1}} \oplus \hat{\mathbf{x}}_{F_n}^{R_{t-1}} \end{bmatrix} \tag{3.20}$$

with corresponding covariance matrix:

$$\begin{aligned} \mathbf{P}_t^{R_t} &\simeq \begin{bmatrix} \mathbf{J}_2 & \mathbf{J}_1 \end{bmatrix} \mathbf{P}_{t|t}^{R_{t-1}} \begin{bmatrix} \mathbf{J}_2^T \\ \mathbf{J}_1^T \end{bmatrix} \\ \mathbf{J}_1 &= \begin{bmatrix} \mathbf{J}_{1\oplus}\{\ominus\hat{\mathbf{x}}_{R_t}^{R_{t-1}}, \hat{\mathbf{x}}_B^{R_{t-1}}\}\mathbf{J}_{\ominus}\{\hat{\mathbf{x}}_{R_t}^{R_{t-1}}\} \\ \vdots \\ \mathbf{J}_{1\oplus}\{\ominus\hat{\mathbf{x}}_{R_t}^{R_{t-1}}, \hat{\mathbf{x}}_{F_n}^{R_{t-1}}\}\mathbf{J}_{\ominus}\{\hat{\mathbf{x}}_{R_t}^{R_{t-1}}\} \end{bmatrix} \\ \mathbf{J}_2 &= \begin{bmatrix} \mathbf{J}_{2\oplus}\{\ominus\hat{\mathbf{x}}_{R_t}^{R_{t-1}}, \hat{\mathbf{x}}_B^{R_{t-1}}\} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{J}_{2\oplus}\{\ominus\hat{\mathbf{x}}_{R_t}^{R_{t-1}}, \hat{\mathbf{x}}_{F_n}^{R_{t-1}}\} \end{bmatrix} \end{aligned} \tag{3.21}$$

The computational cost of the update steps in both absolute and robocentric mapping requires updating the covariance matrix of the estimation and is thus $\mathcal{O}(n^2)$, where n is the number of features in the map. The prediction step in absolute mapping requires updating the correlations between the vehicle and the features and is thus $\mathcal{O}(n)$, while in robocentric mapping it only requires stacking, $\mathcal{O}(1)$. In contrast, robocentric mapping includes an additional composition step in which the full covariance matrix is updated, again with a computational cost of $\mathcal{O}(n^2)$.

3.6.2 Robocentric Map Joining

In [Tardós 02] Tardós et al. proposed a map building technique in which, instead of building one global map from the beginning of the exploration task,

a sequence of local maps of limited size is built, and later joined together, to obtain the global map. Here we show that, not only is map joining computationally more efficient than building one global map from the beginning, as it is shown in [Tardós 02], but it also allows to attain better consistency in the stochastic map.

Robocentric map joining is carried out as follows: given two consecutive robocentric local maps:

$$\mathcal{M}_{\mathcal{F}}^{R_l} = (\hat{\mathbf{x}}_{\mathcal{F}}^{R_l}, \mathbf{P}_{\mathcal{F}}^{R_l}) \;\; ; \;\; \mathcal{F} = \{R_l, B_l, F_1, \ldots, F_m\}$$
$$\mathcal{M}_{\mathcal{E}}^{R_{l-1}} = (\hat{\mathbf{x}}_{\mathcal{E}}^{R_{l-1}}, \mathbf{P}_{\mathcal{E}}^{R_{l-1}}) \;\; ; \;\; \mathcal{E} = \{R_{l-1}, B_{l-1}, E_1, \ldots, E_n\}$$

Because, there exists a link between the two maps $B_l \equiv R_{l-1}$ a full stochastic map can be obtained by map joining:

$$\mathcal{M}_{\mathcal{F}+E}^{R_l} = (\hat{\mathbf{x}}_{\mathcal{F}+\mathcal{E}}^{R_l}, \mathbf{P}_{\mathcal{F}+\mathcal{E}}^{R_l})$$

which contains the estimations of the features from both maps, relative to the reference frame R_l of the current robocentric local map. We proceed as follows:

Stacking together the local maps

Because the robocentric local maps $\mathcal{M}_{\mathcal{F}}^{R_l}$ and $\mathcal{M}_{\mathcal{E}}^{R_{l-1}}$ are built using independent information, they are uncorrelated [Tardós 02]. Thus, we form a stacked state vector:

$$\hat{\mathbf{x}}_{\mathcal{F}+\mathcal{E}} = \begin{bmatrix} \hat{\mathbf{x}}_{\mathcal{F}}^{R_l} \\ \hat{\mathbf{x}}_{\mathcal{E}}^{R_{l-1}} \end{bmatrix} \;\; ; \;\; \mathbf{P}_{\mathcal{F}+\mathcal{E}} = \begin{bmatrix} \mathbf{P}_{\mathcal{F}}^{R_l} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_{\mathcal{E}}^{R_{l-1}} \end{bmatrix} \tag{3.22}$$

which stores all the available information about the previous uncorrelated local maps.

The update step

Data association is carried out to match the features of the local map $\mathcal{M}_{\mathcal{F}}^{R_l}$ with those of the local map $\mathcal{M}_{\mathcal{E}}^{R_{l-1}}$. We use the Joint Compatibility test [Neira 01], which obtains the largest set of pairings which are jointly compatible, a consensus criteria that reduces the possibility of accepting a spurious pairing. Let F_i and E_{j_i} be two matched features, thus, a nonlinear measurement equation of the form:

$$\mathbf{z}_{ij_i} = \mathbf{h}_{ij_i}(\mathbf{x}_{\mathcal{F}+\mathcal{E}}) = \ominus \mathbf{x}_{F_i}^{R_l} \oplus \mathbf{x}_{B_l}^{R_l} \oplus \mathbf{x}_{E_{j_i}}^{R_{l-1}} = \mathbf{0} \tag{3.23}$$

constraints their relative location vector, where $B_l \equiv R_{l-1}$ as discussed above. Linearization of equation (3.23) around the best available estimation $\hat{\mathbf{x}}_{\mathcal{F}+\mathcal{E}}$ gives:

$$\mathbf{z}_{ij_i} \simeq \mathbf{h}_{ij_i}(\hat{\mathbf{x}}_{\mathcal{F}+\mathcal{E}}) + \mathbf{H}_{ij_i}(\mathbf{x}_{\mathcal{F}+\mathcal{E}} - \hat{\mathbf{x}}_{\mathcal{F}+\mathcal{E}}) \tag{3.24}$$

where, the linearization coefficient results from:

$$\mathbf{H}_{ij_i} = \left.\frac{\partial \mathbf{h}_{ij_i}}{\partial \mathbf{x}_{\mathcal{F}+\mathcal{E}}}\right|_{\hat{\mathbf{x}}_{\mathcal{F}+\mathcal{E}}} = \left[\begin{array}{cccccccccc} \mathbf{H}_{B_l} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{H}_{F_i} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{H}_{E_{j_i}} & \mathbf{0} & \dots & \mathbf{0} \end{array}\right] \tag{3.25}$$

and,

$$\mathbf{H}_{B_l} = \left.\frac{\partial \mathbf{h}_{ij_i}}{\partial \mathbf{x}_{B_l}^{R_l}}\right|_{\hat{\mathbf{x}}_{\mathcal{F}+\mathcal{E}}} \;;\; \mathbf{H}_{F_i} = \left.\frac{\partial \mathbf{h}_{ij_i}}{\partial \mathbf{x}_{F_i}^{R_l}}\right|_{\hat{\mathbf{x}}_{\mathcal{F}+\mathcal{E}}} \;;\; \mathbf{H}_{E_{j_i}} = \left.\frac{\partial \mathbf{h}_{ij_i}}{\partial \mathbf{x}_{E_{j_i}}^{R_{l-1}}}\right|_{\hat{\mathbf{x}}_{\mathcal{F}+\mathcal{E}}} \tag{3.26}$$

The update of the stacked state vector by using the EKF equations, would therefore improve not only the structure of both local maps but also the link between them. This strategy increases the accuracy of the map joining technique over the direct change of reference between local map proposed in [Tardós 02]. After updating the map, matched features of the local map $\mathcal{M}_{\mathcal{E}}^{R_{l-1}}$ are removed from the stacked state vector and only a subset $\mathcal{E}^* \subset \mathcal{E}$ of features remains.

The composition step

Given that the features from the current local map are expressed relative to reference R_l and features from the previous local map are expressed relative to reference R_{l-1}, to form the full stochastic map $\mathcal{M}_{\mathcal{F}+\mathcal{E}^*}^{R_l}$ we only need to transform the features of the previous map to the reference R_l using their common link. Thus,

$$\mathbf{x}_{\mathcal{F}+\mathcal{E}^*}^{R_l} = \left[\begin{array}{c} \mathbf{x}_{\mathcal{F}}^{R_l} \\ \mathbf{x}_{\mathcal{E}^*}^{R_l} \end{array}\right] = \left[\begin{array}{c} \mathbf{x}_{\mathcal{F}}^{R_l} \\ \mathbf{x}_{R_{l-1}}^{R_l} \oplus \mathbf{x}_{\mathcal{E}^*}^{R_{l-1}} \end{array}\right] \tag{3.27}$$

which directly provides the estimation $\hat{\mathbf{x}}_{\mathcal{F}+\mathcal{E}^*}^{R_l}$ of the full stochastic map. Its covariance matrix $\mathbf{P}_{\mathcal{F}+\mathcal{E}^*}^{R_l}$ derives from the linearization of equation (3.27) as:

$$\mathbf{P}_{\mathcal{F}+\mathcal{E}^*}^{R_l} = \left[\begin{array}{cc} \mathbf{P}_{\mathcal{F}}^{R_l} & \mathbf{P}_{\mathcal{F}\mathcal{E}^*}^{R_l} \\ \mathbf{P}_{\mathcal{E}^*\mathcal{F}}^{R_l} & \mathbf{P}_{\mathcal{E}^*}^{R_l} \end{array}\right] \simeq \left[\begin{array}{cc} \mathbf{I} & \mathbf{0} \\ \mathbf{J}_1 & \mathbf{J}_2 \end{array}\right] \left[\begin{array}{cc} \mathbf{P}_{\mathcal{F}}^{R_l} & \mathbf{P}_{\mathcal{F}\mathcal{E}^*} \\ \mathbf{P}_{\mathcal{E}^*\mathcal{F}} & \mathbf{P}_{\mathcal{E}^*}^{R_{l-1}} \end{array}\right] \left[\begin{array}{cc} \mathbf{I} & \mathbf{J}_1^T \\ \mathbf{0} & \mathbf{J}_2^T \end{array}\right]$$

where the matrices of Jacobians (appendix A) are:

$$\mathbf{J}_1 = \begin{bmatrix} \mathbf{J}_{1\oplus}\{\hat{\mathbf{x}}^{R_l}_{R_{l-1}}, \hat{\mathbf{x}}^{R_{l-1}}_{B_{l-1}}\} & \dots & \mathbf{0} \\ \vdots & & \vdots \\ \mathbf{J}_{1\oplus}\{\hat{\mathbf{x}}^{R_l}_{R_{l-1}}, \hat{\mathbf{x}}^{R_{l-1}}_{E_n}\} & \dots & \mathbf{0} \end{bmatrix}$$

and

$$\mathbf{J}_2 = \begin{bmatrix} \mathbf{J}_{2\oplus}\{\hat{\mathbf{x}}^{R_l}_{R_{l-1}}, \hat{\mathbf{x}}^{R_{l-1}}_{B_{l-1}}\} & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \mathbf{J}_{2\oplus}\{\hat{\mathbf{x}}^{R_l}_{R_{l-1}}, \hat{\mathbf{x}}^{R_{l-1}}_{E_n}\} \end{bmatrix}$$

As in absolute map joining [Tardós 02], the computational cost of robocentric map joining requires updating the full stochastic map covariance at each join and thus is $\mathcal{O}(n^2)$. Note however that most of the updates take place in a local map of bounded size (with a $\mathcal{O}(1)$ cost), and thus you can expect robocentric map joining to cut processing time by a large constant factor.

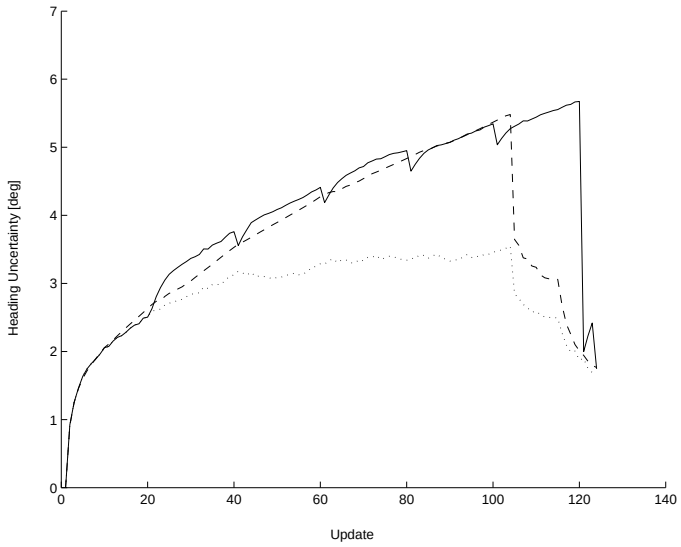
3.6.3 Experiments

In this section, we carry out a series of simulated and indoor/outdoor experiments to validate the robocentric map joining algorithm.

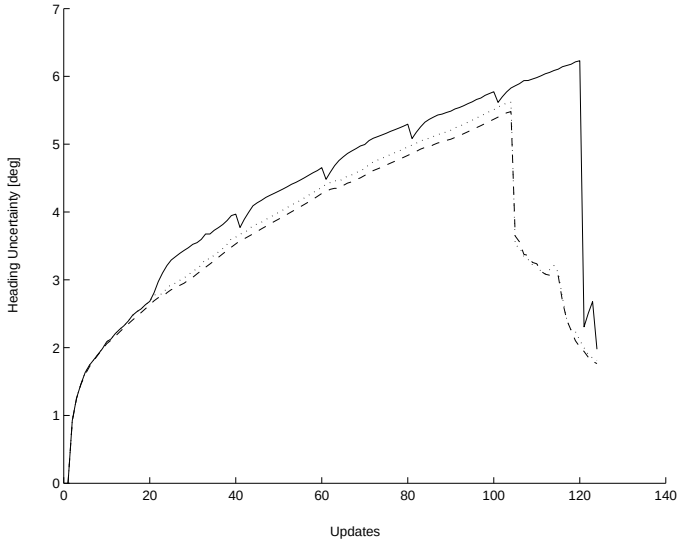
Simulation

In our controlled simulation environment we have compared the performance of the different algorithms presented in the previous discussion. Two main categories of experiments have been conducted, namely, those with an absolute representation, and those with a robot centered representation.

Figure 3.4(a) compares the vehicle heading uncertainty computed using an absolute map representation by an ideal error-free EKF (simulated with noise = 0), the standard EKF and the map joining algorithm proposed in [Tardós 02]. Note that all mapping algorithms in error-free simulations should produce identical correct results. In this case, the heading uncertainty computed by the standard EKF presents a saturation effect which makes the algorithm more and more optimistic as the number of updates increases, and thus it undermines consistency. The map joining algorithm performs much better but it is still slightly optimistic in the second-half of the trajectory.



(a) Absolute mapping



(b) Robocentric mapping

Figure 3.4: Vehicle heading uncertainty (1σ) computed using absolute and robocentric mapping by: the ideal error-free EKF (dashed line), the standard EKF (dotted line) and the map joining algorithm (solid line).

Similarly, figure 3.4(b) compares the vehicle heading uncertainty, using a robot centered representation[3], of the ideal error-free EKF against the robocentric mapping approach reported in [Castellanos 04] and the new robocentric map joining algorithm. As observed from the figure, both algorithms obtain a non-optimistic estimation for the vehicle heading uncertainty along the vehicle trajectory, which makes loop closing detection possible. In this experiment, robocentric map joining behaves slightly pessimistically as compared to the basic robocentric approach. It provides a more efficient solution in terms of computing time. Finally, we can see that in these robot centered approaches, saturation effects have disappeared.

Indoor/outdoor mapping experiment

To validate the new robocentric mapping algorithm, we have conducted an experiment in one of the buildings at our campus using a robotized wheelchair equipped with a SICK laser scanner. The vehicle was hand-driven along a mixed indoor/outdoor path of about 250 m, at a mean speed of 0.45 m/s. The scans were processed to obtain line features using the ASSC algorithm introduced in chapter 2.

Figure 3.5 shows the map obtained along the commanded trajectory by the classical EKF-SLAM algorithm using an absolute representation. The saturation effect in the map uncertainty makes the result inconsistent as observed in the top-left part of the figure, where clearly the loop could not be closed by simple data association strategies. We processed the same data dividing the full map into 50 robocentric maps (each for a trajectory of 5 m with approximate 10 line features) and using robocentic map joining to compute the full stochastic map. As shown in figure 3.6, this change of representation, from absolute to robot centered using map joining performs adequately in this case. Due to the increased accuracy in linearization and the reduced level of uncertainty of this local representation, the mapping of the 250m trajectory was accurately performed, closing the loop when the vehicle homed.

Finally, figure 3.7 depicts the evolution of the vehicle angular uncertainty along the 250m trajectory. Again, the EKF-SLAM estimated uncertainty presents the previously discussed saturation effects driving the solution of the mapping algorithm out of consistency. The Map Joining algorithm which also computes the estimated location of features in an absolute representation presents a similar, although clearly improved, optimistic behavior. As

[3]For ease of comparison, the robot centered solutions have been transformed back to the absolute representation.

Figure 3.5: Classical EKF-SLAM algorithm with an absolute representation. Observe that the vehicle location uncertainty (extremely small ellipses) is incompatible with the real error, especially in the top left part of the figure, where clearly, multiple hypotheses for the same feature appear.

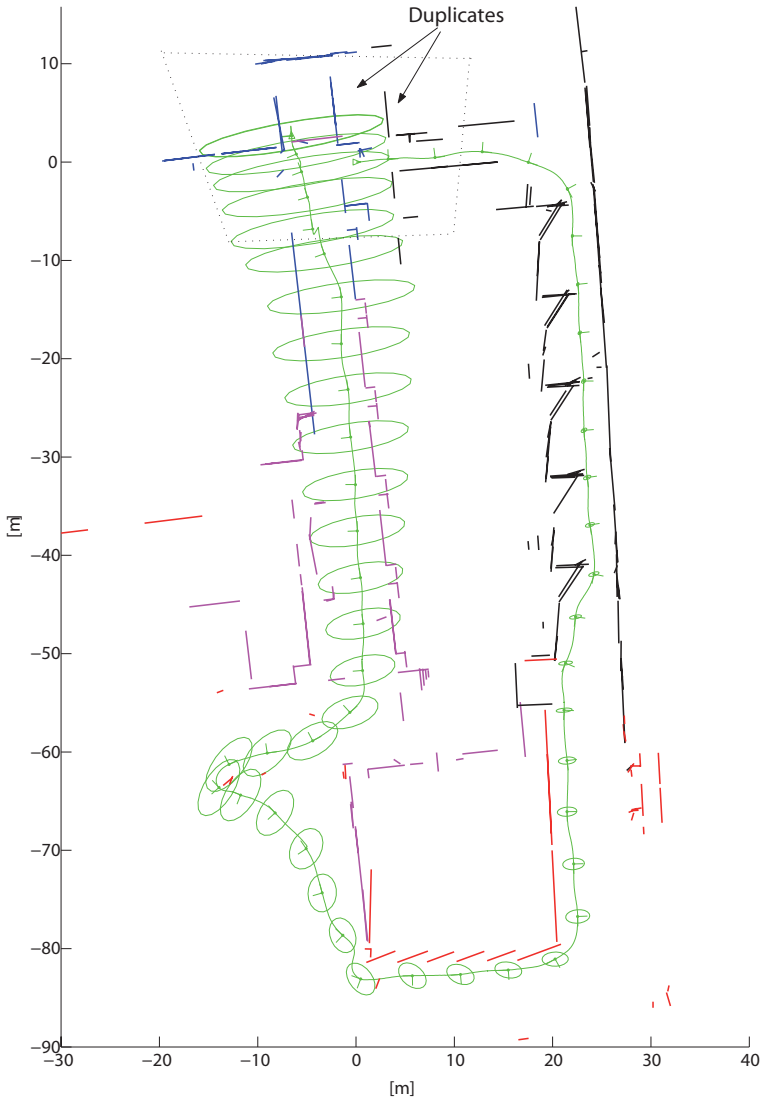


Figure 3.6: Robocentric Map Joining before loop closing. In this case, the vehicle location uncertainty is consistent with the real error, as observed in the top left part of the figure (Note that results have been transformed back to the absolute representation).

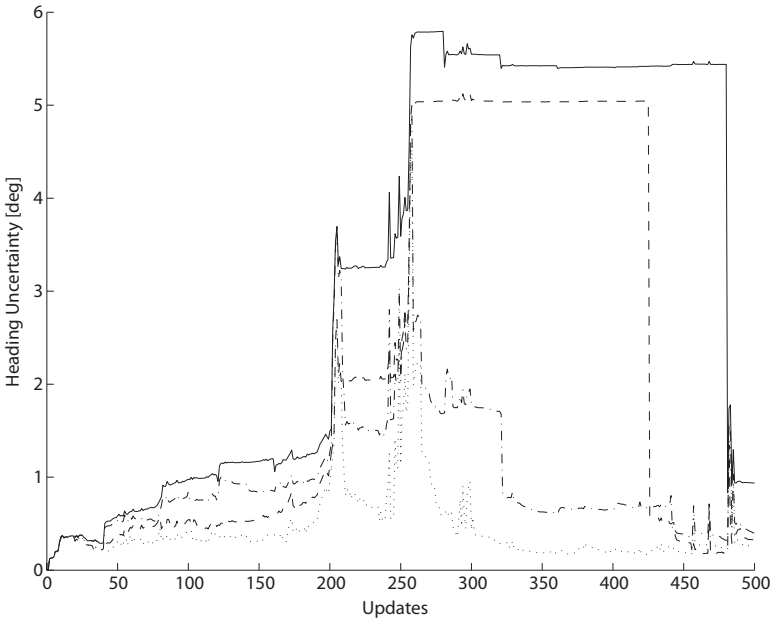
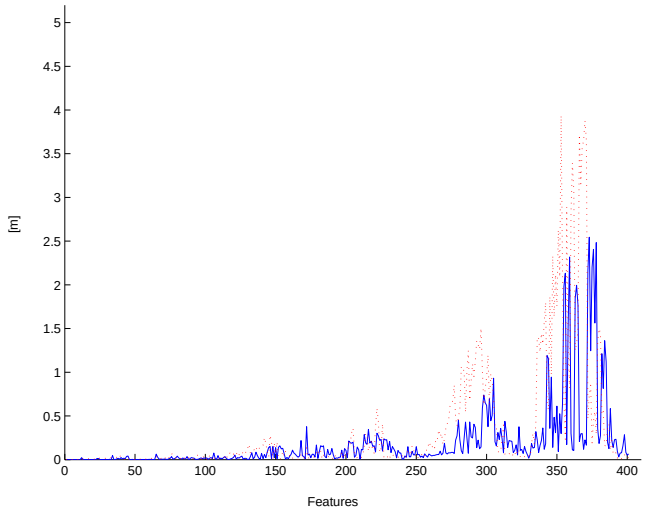


Figure 3.7: Experimental vehicle heading uncertainty: EKF-SLAM algorithm (dotted line), Map Joining algorithm (dash-dot), Robocentric (dashed) and Robocentric Map Joining (solid).

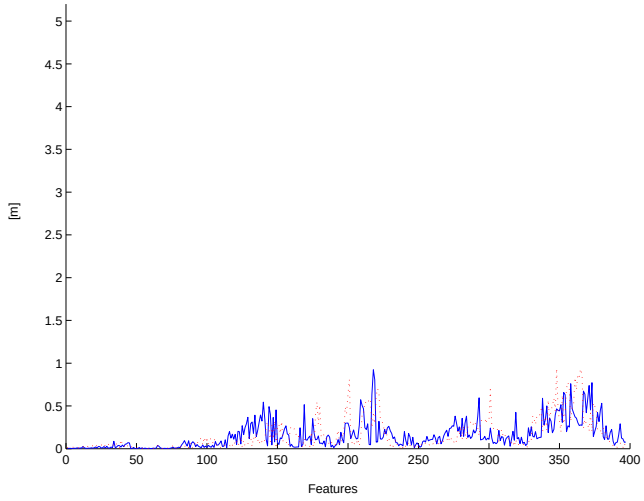
observed in simulation, and now in a real experiment, the robot centered representation surpasses these inconsistency problems, at least at the scale of the reported experiments. Although the Robocentric mapping algorithm performed consistently, the Robocentric Map Joining algorithm provided the more efficient solution from the computational time point-of-view with an accurate estimation of the uncertainty.

Long term outdoor mapping experiment

For large-scale outdoor testing experiments, a benchmark dataset has been used [Guivant 01]. It had been captured using a truck (Ackerman vehicle) driving through Victoria Park, in Sydney, Australia. On-board sensors provide 2D range scans and odometry (speed and steering). The quality of the odometry measurements is very poor, however, the dataset includes a feature extractor to detect cylindrical landmarks (trees) with high accuracy, but with a significant level of spurious landmarks. The whole trajectory is about 3.5 km with several short loops and two big loops which are specially critic owing

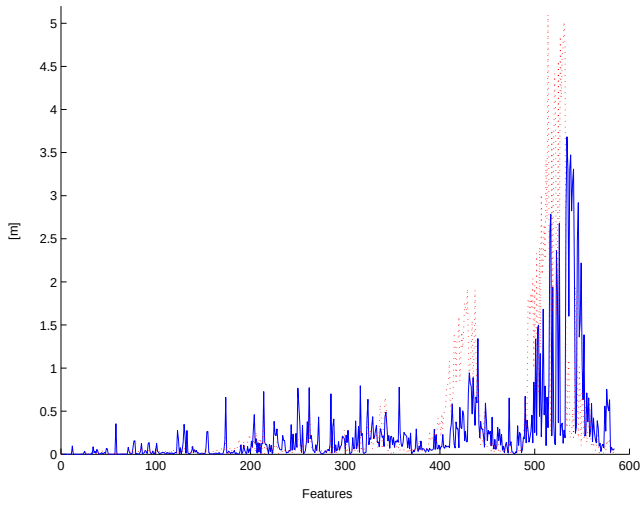


(a) Absolute Global Map

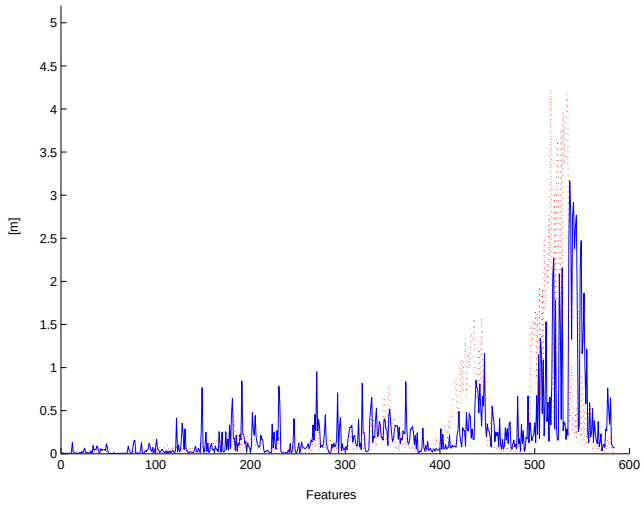


(b) Robocentric Global Map

Figure 3.8: Landmarks uncertainty level at final step of the Victoria Park experiment (σ_x solid, σ_y dotted). Robocentric representation gives lower final uncertainty levels although is able to close the same loops (same consistency, more accuracy).



(a) Absolute Local Maps



(b) Robocentric Local Maps

Figure 3.9: Landmarks uncertainty level at final step of the Victoria Park experiment (σ_x solid, σ_y dotted). Local map sequencing approach computes higher uncertainty levels because not all posible data association has been done, but this effect is reduced using both robocentric and local representations

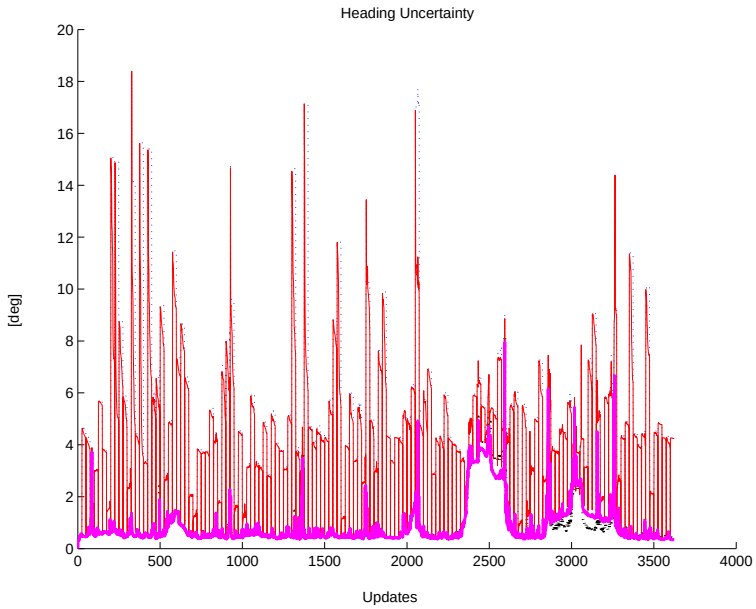


Figure 3.10: Heading uncertainty (1σ) using different combinations of robocentric representation and local maps. Solid line is absolute representation and dotted line corresponds to robocentric representation. Upper lines shows uncertainty level for local maps based algorithms. Downwards peaks represents map-joining phases

to the low level of reliable features in these zones.

As previously commented, Joint Compatibility [Neira 01] data association algorithm has been used. In practice, this algorithm guarantees a low number of false positives correspondences, which can lead to filter divergence. However, the number of false negatives increases.

Figures 3.8 and 3.9 show the final uncertainty (σ_x, σ_y) of mapped landmarks. Since the number of landmarks detected is always the same, the difference in the final number of mapped landmarks comes from failures or restrictions during data association step. As we can see, the number of landmarks of the final map built using local maps is bigger, because data association during map-joining is more delicate and, as previously addressed, Joint Compatibility tends to be restrictive (high number of false negatives). In consequence, non-matched observations appears as new landmarks. Furthermore, non-matched landmarks means information loss during update. Hence, final uncertainty is larger.

Using either a global or local framework, both absolute and robocentric

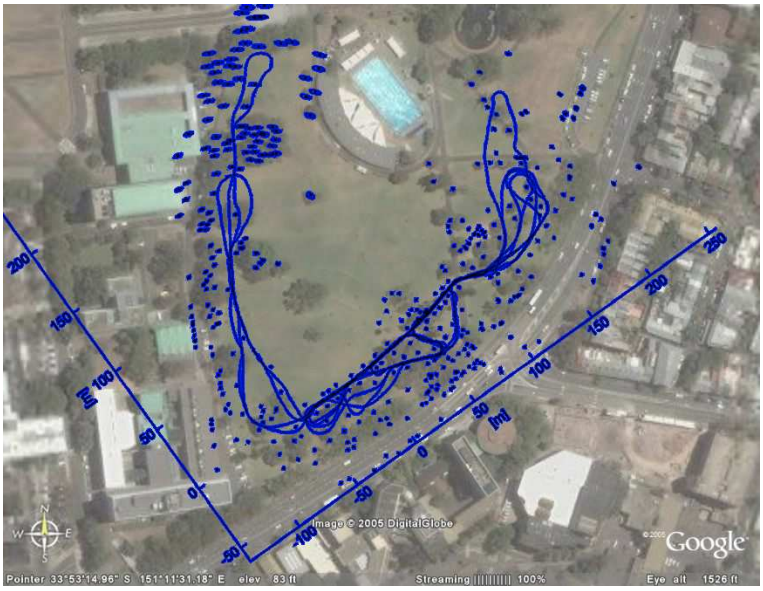


Figure 3.11: Trajectory and final feature map superimposed with a satellite image

representation have similar number of landmarks. Nevertheless, robocentric approach provides lower levels of uncertainty, but still allowing loop closure. Ergo, higher accuracy level is obtained preserving consistence and convergence.

As commented before, a classical approach to fight EKF inconsistence is the so called *injecting stabilizing noise process* [Maybeck 79], which consist in tuning the matrix covariance increasing observation noise in order to assure a pessimistic covariance. Usually, this process can lead to false positive data association and inaccurate maps. However, robocentric approach could provide high accuracy despite injected noise.

Finally, heading uncertainty evolution is shown on figure 3.10. On the one hand, global map algorithms have lower uncertainty levels that are reached only during map joining phase in local maps building. This validates the results obtained during simulation experiments. On the other hand, robocentric representation have similar behavior compared to absolute representation because measurements are very accurate in this dataset.

Figure 3.11 shows the final map plus the estimated trajectory compared with a satellite photography[4].

[4]Courtesy of Google Earth http://earth.google.com

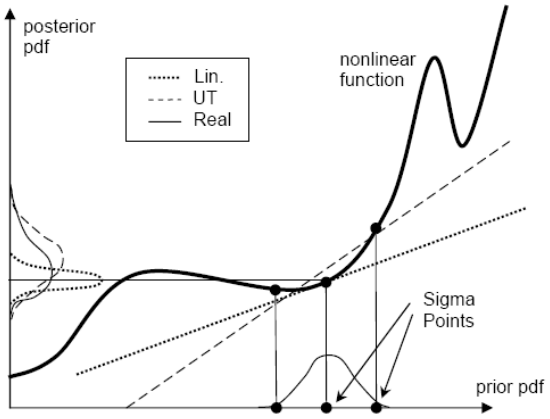


Figure 3.12: Propagation of a normal distribution through a nonlinear function. The first order Taylor expansion (dotted) only uses information of the function at the mean point to compute the linear approximation, while the UT (dashed) approaches the function with a linear regression of several sigma points. The actual distribution is the solid one. (Adapted from [van der Merwe 04])

3.7 Unscented Filtering

The core of unscented filtering is the so-called *unscented transformation* (UT) [Julier 00]. Approximating the prior distribution by a proper discrete function provides a minimal set of samples, named *sigma points*, which capture the moments of the underlying density function. The moments of the posterior distribution are then obtained by means of weighted linear regression of evaluations of the nonlinear function at the selected samples points. Figure 3.12 compares the propagation of a normal distribution through a nonlinear function obtained by analytical linearization and by the unscented transformation (i.e. statistical linearization).

The *Scaled Unscented Transform* (SUT) has been proposed [Julier 02, van der Merwe 04] which generalizes the standard UT, allowing an extra degree of freedom of the sigma-points. This can be used to give more importance to modes or tails of the distribution, depending on the application. Van der Merwe [van der Merwe 04] proved that the SUT is part of the more general *sigma point filters*, which achieve similar performance results. Other sigma point methods are the *central difference filter* (CDF) [Ito 00] and the *divided difference filter* (DDF) [Nørgaard 00]. Improved performance can be achieved exploiting the Gaussian quadrature integration in the *Gauss-Hermite filter*

[Ito 00]. However, the computational cost of this filter is exponential, which becomes intractable for SLAM. For an N-dimensional state vector, the symmetric set of $2N+1$ sigma-points that characterize the SUT is given by

$$\begin{aligned} \mathcal{X}^{(0)} &= \hat{\mathbf{x}} \\ \mathcal{X}^{(j)} &= \hat{\mathbf{x}} + \left(\sqrt{(N+\lambda)\,\mathbf{P}}\right)_j \quad , \; j = 1, \ldots, N \\ \mathcal{X}^{(j)} &= \hat{\mathbf{x}} - \left(\sqrt{(N+\lambda)\,\mathbf{P}}\right)_j \quad , \; j = N+1, \ldots, 2N \end{aligned} \tag{3.28}$$

where $\hat{\mathbf{x}}$ and $\mathbf{P}$ are, respectively, the mean and covariance of the sampled distribution. For efficient computation of the matrix square root, a Cholesky decomposition $\mathbf{P} = \mathbf{S}\mathbf{S}^T$ is used. The parameter λ controls the scaling of the sigma-points. If we use the SUT, $\lambda = \alpha^2(N+\kappa) - N$. Then, for a Gaussian prior $N + \kappa = 3$ [Julier 02] and therefore, the numerical behavior of the SUT is the same as the *Central Difference Filter* with a step $h = \sqrt{3}$ and the *Gauss-Hermite Filter* with 3 points [Ito 00]. For a complete explanation of the parameters of the SUT refer to [van der Merwe 04].

Let $\mathbf{x}_{t-1}$ be the state vector at time $t-1$, and let $\{\mathcal{X}_{t-1}^{(0)}, \ldots, \mathcal{X}_{t-1}^{(2N)}\}$ be the set of sigma-points computed from its distribution. Also, let the state evolution be characterized by the nonlinear function

$$\mathbf{x}_t = \mathbf{f}_t(\mathbf{x}_{t-1}) \tag{3.29}$$

Hence, the set of sigma-points are transformed by $2N+1$ evaluations of the nonlinear function at the sigma-points:

$$\mathcal{X}_t^{(j)} = \mathbf{f}_t(\mathcal{X}_{t-1}^{(j)}), \quad j = 0, \ldots, 2N \tag{3.30}$$

Then, the first two moments of the density function of $\mathbf{x}_t$ are computed by a weighted linear regression of the transformed sigma-points

$$\hat{\mathbf{x}}_t = \sum_{j=0}^{2N} \omega_m^{(j)} \mathcal{X}_k^{(j)} \tag{3.31}$$

$$\mathbf{P}_t = \sum_{j=0}^{2N} \omega_c^{(j)} (\mathcal{X}_t^{(j)} - \hat{\mathbf{x}}_t)(\mathcal{X}_t^{(j)} - \hat{\mathbf{x}}_k)^T \tag{3.32}$$

where the weights are given by

$$\begin{aligned}\omega_c^{(0)} &= \frac{\lambda}{N+\lambda} + (1-\alpha^2+\beta) \\ \omega_m^{(0)} &= \frac{\lambda}{N+\lambda} \\ \omega_m^{(j)} &= \omega_c^{(j)} = \frac{1}{2(N+\lambda)}, \quad j = 1,\ldots,2N\end{aligned} \tag{3.33}$$

Note that the weights for the computation of the mean (m) and the covariance (c) are different in the 0-th component to compensate for scaling.

3.7.1 Unscented SLAM

Let $\mathbf{x}_{t-1}^B \sim \mathcal{N}(\hat{\mathbf{x}}_{t-1}^B, \mathbf{P}_{t-1}^B)$ be the stochastic map available at time $t-1$. Let $\mathbf{x}_{R_t}^{R_{t-1}} \sim \mathcal{N}(\hat{\mathbf{x}}_{R_t}^{R_{t-1}}, \mathbf{P}_{R_t}^{R_{t-1}})$ be the vehicle motion from time step $t-1$ to time step t as estimated by odometry, and finally let $\mathcal{E} = \{E_1,\ldots,E_m\}$ be the set of observations gathered by on-board sensors at time t, with a joint-Gaussian distribution with covariance matrix $\mathbf{P}_{\mathcal{E}_t}^{R_t}$. Assuming independence between the prior state estimation, the displacement and the set of available measurements, we define an augmented stochastic state vector with mean

$$\hat{\mathbf{x}}_{a,t-1}^B = \begin{bmatrix} \hat{\mathbf{x}}_{R_{t-1}}^B \\ \hat{\mathbf{x}}_{\mathcal{F}_{t-1}}^B \\ \hat{\mathbf{x}}_{\mathcal{E}_t}^{R_t} \\ \hat{\mathbf{x}}_{R_t}^{R_{t-1}} \end{bmatrix} \tag{3.34}$$

and a block-diagonal covariance matrix

$$\mathbf{P}_{a,t-1}^B = \begin{bmatrix} \mathbf{P}_{R_{t-1}}^B & \mathbf{P}_{R\mathcal{F}_{t-1}}^B & \mathbf{0} & \mathbf{0} \\ \mathbf{P}_{\mathcal{F}R_{t-1}}^B & \mathbf{P}_{\mathcal{F}_{t-1}}^B & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{P}_{\mathcal{E}_t}^{R_t} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{P}_{R_t}^{R_{t-1}} \end{bmatrix} \tag{3.35}$$

By using the deterministic sampling algorithm described in section 3.7, we obtain a small-size set of sigma-points which accurately represents the first two moments of the previous distribution: $\{\mathcal{X}_{a,t-1}^{(0)},\ldots,\mathcal{X}_{a,t-1}^{(2N)}\}$.

The set of sigma-points at time $t-1$ is propagated forward in time through the nonlinear state equation

$$\mathbf{x}_{a,t|t-1}^B = \mathbf{f}_t(\mathbf{x}_{a,t-1}^B) \tag{3.36}$$

with

$$\begin{aligned}
\mathbf{x}^B_{R_t} &= \mathbf{x}^B_{R_{t-1}} \oplus \mathbf{x}^{R_{t-1}}_{R_t} \\
\mathbf{x}^B_{\mathcal{F}_t} &= \mathbf{x}^B_{\mathcal{F}_{t-1}} \\
\mathbf{x}^B_{E_{1,t}} &= \mathbf{x}^B_{R_{t-1}} \oplus \mathbf{x}^{R_{t-1}}_{R_t} \oplus \mathbf{x}^{R_t}_{E_{1,t}} \\
&\vdots \\
\mathbf{x}^B_{E_{m,t}} &= \mathbf{x}^B_{R_{t-1}} \oplus \mathbf{x}^{R_{t-1}}_{R_t} \oplus \mathbf{x}^{R_t}_{E_{m,t}}
\end{aligned} \tag{3.37}$$

where $\oplus$ represents the composition of location vectors (appendix A). The predicted mean at time t of the augmented state vector and its estimated error covariance matrix are then computed from a linear weighted regression of the transformed sigma-points $\{\mathcal{X}^{(0)}_{a,t|t-1}, \ldots, \mathcal{X}^{(2N)}_{a,t|t-1}\}$:

$$\hat{\mathbf{x}}^B_{a,t|t-1} = \sum_{j=0}^{2N} \omega_m^{(j)} \mathcal{X}^{(j)}_{a,t|t-1} \tag{3.38}$$

and,

$$\mathbf{P}^B_{a,t|t-1} = \sum_{j=0}^{2N} \omega_c^{(j)} (\mathcal{X}^{(j)}_{a,t|t-1} - \hat{\mathbf{x}}^B_{a,t|t-1})(\mathcal{X}^{(j)}_{a,t|t-1} - \hat{\mathbf{x}}^B_{a,t|t-1})^T \tag{3.39}$$

Data association provides the observation $\mathbf{y}_t$ statistically compatible and related to the augmented state vector by a nonlinear function $\mathbf{h}_t$:

$$\mathbf{y}_t = \mathbf{h}_t(\mathbf{x}^B_{a,t|t-1}) \tag{3.40}$$

Hence, the update of the estimated mean and estimated error covariance at time t follows from:

$$\hat{\mathbf{x}}^B_{a,t} = \hat{\mathbf{x}}^B_{a,t|t-1} + \mathbf{P}_{\mathbf{x}\nu}\mathbf{S}_t^{-1}(\mathbf{y}_t - \hat{\mathbf{y}}_t) \tag{3.41}$$

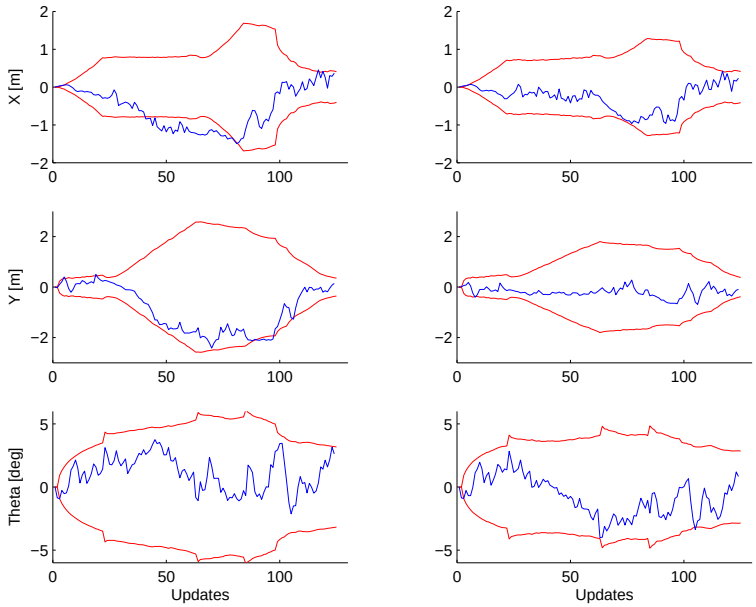
$$\mathbf{P}^B_{a,t} = \mathbf{P}^B_{a,t|t-1} - \mathbf{P}_{\mathbf{x}\nu}\mathbf{S}_t^{-1}\mathbf{P}_{\nu\mathbf{x}} \tag{3.42}$$

where

$$\hat{\mathbf{y}}_t = \sum_{j=0}^{2N} \omega_m^{(j)} \mathcal{Y}_t^{(j)} \tag{3.43}$$

with

$$\mathcal{Y}_t^{(j)} = \mathbf{h}_t(\mathcal{X}^{(j)}_{a,t|t-1}) \tag{3.44}$$



(a) (b)

Figure 3.13: Estimated errors and 2σ bounds for the (x, y, θ) components of the vehicle estimated location: (a) EKF-based SLAM and (b) Unscented SLAM.

and

$$\mathbf{S}_t = \sum_{j=0}^{2N} \omega_c^{(j)} (\mathcal{Y}_t^{(j)} - \hat{\mathbf{y}}_t)(\mathcal{Y}_t^{(j)} - \hat{\mathbf{y}}_t)^T \tag{3.45}$$

$$\mathbf{P}_{\mathbf{x}\nu} = \sum_{j=0}^{2N} \omega_c^{(j)} (\mathcal{X}_{a,t|t-1}^{(j)} - \hat{\mathbf{x}}_{a,t|t-1}^{B})(\mathcal{Y}_t^{(j)} - \hat{\mathbf{y}}_t)^T \tag{3.46}$$

As clearly stated in the previous equations, the expectations of the stochastic vectors involved in the Bayesian solution to the non-linear SLAM problem, given by equations (3.3)-(3.7) are approximated by weighted evaluations of the model and measurement functions at the deterministically selected sigma-points.

3.7.2 Experiments

This section presents both simulation and real experimental results to demonstrate the applicability of the unscented filter to the SLAM problem.

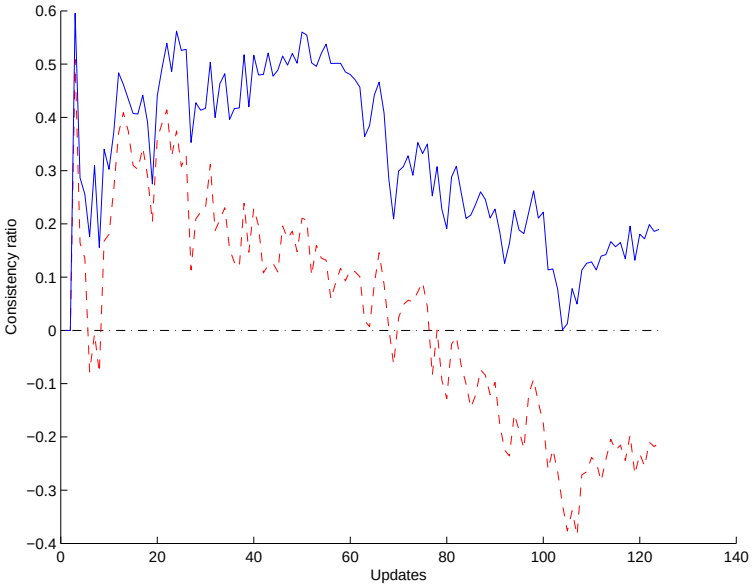


Figure 3.14: Consistency ratio for EKF-based SLAM (dashed) and Unscented SLAM (solid) for the 120 m simulated loop trajectory.

Simulation Results

A first experiment has been designed to isolate the effects of linearization errors on the consistency of the SLAM solution, based on our simulated environment. Gaussian-distributed synthetic errors were generated for both the sensor measurements and for the odometry model of the vehicle. Additionally, known data association is considered.

The evolution of the errors in the components of the vehicle estimated location are plotted in figure 3.13. In general, the unscented filter provides lower estimated errors than the EKF-based approach with slightly more tight uncertainty bounds, which would result in lower ambiguity of data association in a real application. Figure 3.14 plots the consistency ratio, $1-\text{NEES}/\chi^2_{r,1-\alpha}$, for both the EKF-based SLAM and the Unscented SLAM. Clearly, as previously reported, the EKF-based approach becomes inconsistent after only a few update steps. Nevertheless, the Unscented filter remains consistent (up to 5% statistical error) during the complete vehicle trajectory. Furthermore, figure 3.15 illustrates that even after a 400-m loop trajectory, the Unscented SLAM remains consistent.

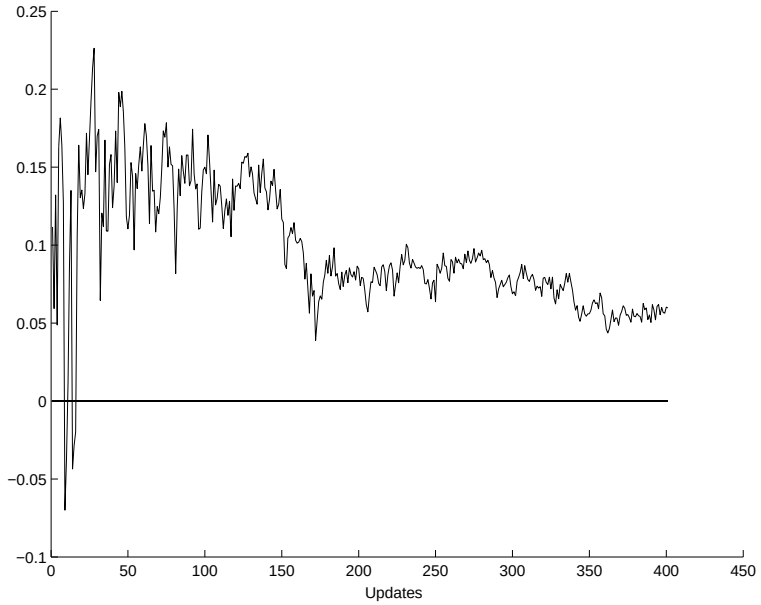


Figure 3.15: Consistency ratio for Unscented SLAM in a 400-m long loop. At the end, the estimator remains consistent, so the loop could be closed with simple data association strategies.

Long term outdoor mapping experiment

We have also tested the Unscented SLAM with the large outdoor dataset [Guivant 01].

Figure 3.16 describes the Unscented SLAM solution, with the complete vehicle trajectory and the detected environmental features with the estimated location uncertainty. Our approach keeps a high level of accuracy during all the path enough to do the data association using only a simple nearest neighbor algorithm. For example, the heading of the robot is bounded by a standard deviation of 0.5 deg during all the trajectory, except in a reduced number of steps. Furthermore, the heading deviation is lower than 0.25 deg most of time (figure 3.17).

The evolution of the consistency ratio $1-\text{NIS}/\chi^2_{r,1-\alpha}$ along the complete trajectory is presented in figure 3.18. The number of inconsistent updates, from the NIS view-point, is roughly 7%, sufficiently close to the theoretical 5%. Furthermore, some of these inconsistent steps are produced by spurious observations. So, we can conclude that the Unscented SLAM is statistically

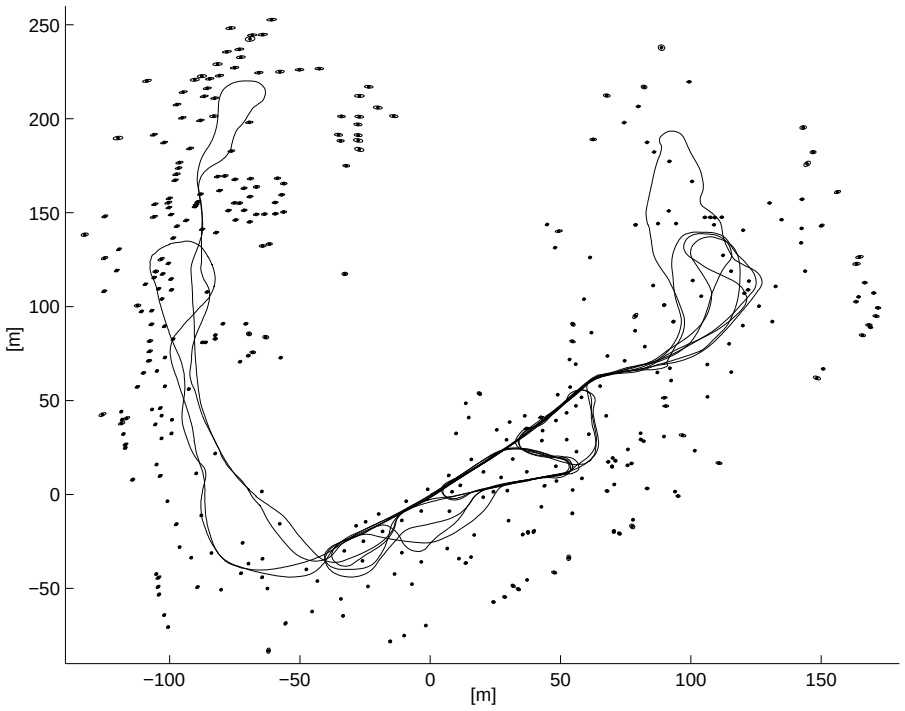


Figure 3.16: Outdoor SLAM: Final 2D-point feature based map and vehicle estimated trajectory. Observe that the given trajectory has not been corrected backwards in time.

consistent in a 3.5 km outdoor trajectory.

3.8 Conclusion

In this chapter we have shown that in the standard extended Kalman Filter approach to SLAM, linearization errors produce inconsistency problems that show up long before computational problems arise. We follow a precise definition of filter consistency that considers both the accuracy of the estimation and of its covariances.

We have proposed the Robocentric Map Joining algorithm which improves consistency of the mapping scheme by: (1) bounding the uncertainty along the exploration trajectory using a sequence of local maps, and (2) improving linearization of the model equations due to the reduced level of uncertainty provided by the robot centered representation. As described both in simulation

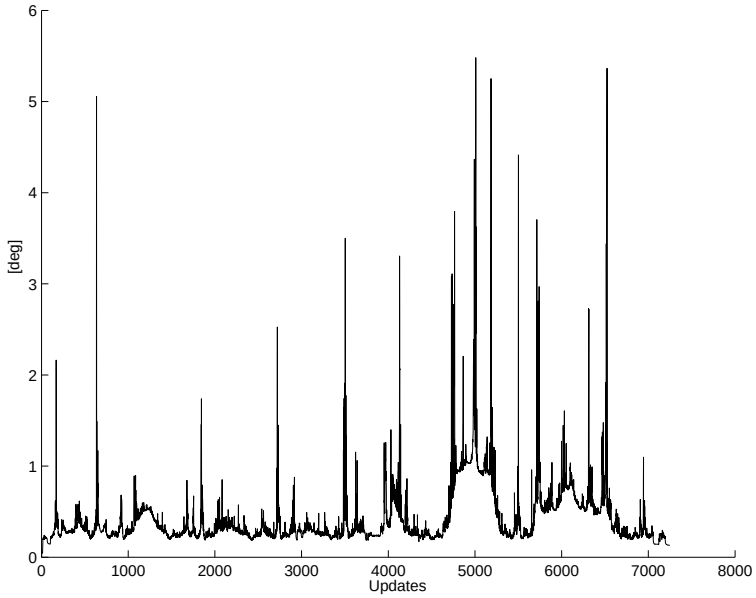


Figure 3.17: 1σ bound for the estimated heading error of the vehicle along the trajectory as computed by the Unscented filter. The maximum standard deviation in the complete trajectory (around 3.5km) is lower than 0.5deg

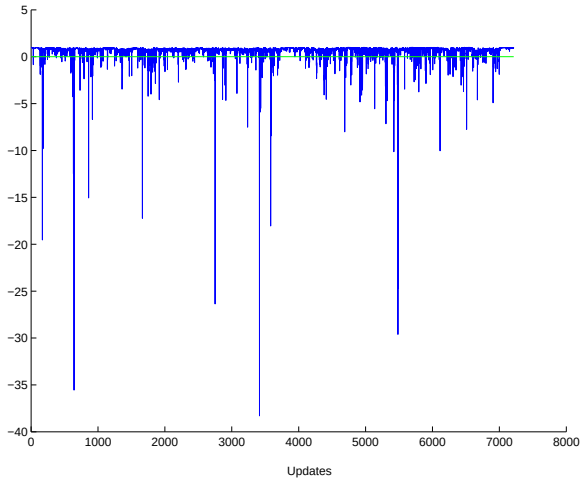


Figure 3.18: Consistency ratio of complete evolution of the state vector of the Unscented SLAM. Only the 7% of the poses are inconsistent which is similar to the significant level (5%)

and in real mixed indoor/outdoor experiments, the combination of local map joining and robocentric mapping allows to apply the EKF-based solution to SLAM at a larger scale.

We have also presented the Unscented SLAM, which represents a feasible framework to the solution of the simultaneous localization and mapping problem. Accuracy of state estimation has been increased (lower errors, tighter uncertainty bounds) over the classical EKF-based approach. We emphasized the improvement in the consistency of the sequential algorithm achieved by avoiding analytical linearization of the model equations. A normalized innovation-based consistency checking has been presented to support the applicability of the approach to large-scale missions.

It is however likely that nonlinearity problems will arise again as larger environments are tackled. We feel that to overcome these limitations it is important to investigate the use of alternative formulations to SLAM, nonlinear and non-Gaussian methods. Making these methods computationally efficient to be used in real time is the next important challenge in SLAM.

Chapter 4

Filtering and learning in Sequential Monte Carlo SLAM

4.1 Introduction

We have seen in the previous chapter that the EKF linearization can lead to filter divergence. In addition, we have proposed techniques to reduce the linearization effect, using higher order approximations with unscented transformations and geometric manipulations. Though these techniques result in some improvement, the inherent step by step approximations remains, causing accumulation of errors and irreversible filter divergence.

The introduction of sequential Monte Carlo (SMC) methods, also called, particle filters (PFs) gave researchers the power and flexibility to handle nonlinearity and non-Gaussian distributions routinely [Fox 01]. Moreover, it enabled researchers to exploit conditional independence, using the Rao-Blackwellized particle filtering (RBPF) variance reduction technique, to obtain more efficient Monte Carlo schemes. RBPFs were applied to dynamic maps [Doucet 00] and subsequently to static maps with the celebrated FastSLAM algorithm [Montemerlo 03b]. Also, RBPF presents extra advantages in terms of computational complexity, joint feature-based and grid mapping [Sim 06, Wurm 07] and ambiguous data association [Montemerlo 03a].

The application of RBPF to dynamic maps is only sensible inasmuch as one has a good model to describe the evolution of the *dynamic* map. Recently, the application of RBPF to static maps has come into question. It has become

popular knowledge that the approach of FastSLAM in robotics can diverge [Bailey 06b]. In loose terms, learning static variables (the map) by conditioning on increasing histories of the state variables results in an accumulation of Monte Carlo errors and an explosion of variance. Heuristic approaches to ameliorate the situation [Stachniss 05b, Elinas 06] have been proposed, but these do no solve the fundamental problem at hand. We note that the problem of PF divergence resulting from learning fixed variables by conditioning on increasing paths was already described in as early as 1999 [Andrieu 99].

This lack of statistical consistency of the most popular SLAM methods has naturally created some justified concern. In this chapter, we analize the FastSLAM algorithm and similar approaches. Then we present Marginal-SLAM, a novel approach with two key ingredients to avoid limitations of previous algorithms.

The first ingredient is to consider SLAM as a robot localization problem with unknown observation model parameters. Thus, the core is to *treat static maps as parameters*, which by necessity are learned using maximum likelihood (ML) or maximum a posteriori (MAP) inference[1]. The idea of treating maps as parameters is not new. It has been central to the incremental ML method [Thrun 93]. However, this method resorts to an ML estimate of the state and hence fails to manage the uncertainty in the robot state properly, as elaborated in [Thrun 02]. A variation based on learning the distribution of the robot states and using an ML estimator of the map as a function of the existing and *growing state trajectories* was proposed in [Thrun 01]. This approach unfortunately suffers from the same consistency problems as FastSLAM.

For full SLAM, various EM[2] approaches using forward-backward state smoothing and map estimation in the M step have also been proposed; see [Thrun 02] for a survey. However, this approach only applies when learning the map off-line. Moreover, it is only very recently that particle smoothing has become feasible [Klaas 06]. Although improved smoothing techniques such as the two-filter smoother, which do better than forward-backward smoothers in the PF context, are very inefficient for SLAM. Some incremental ML methods have appeared in the literature but the incremental stage introduces some extra approximations that again leads to increasing errors [Kaess 07, Olson 07, Frese 05].

Artificial dynamic models, MCMC[3] moves, sufficient statistics computa-

[1]The reason is because there is no full-Bayesian algorithm for joint parameter and state estimation to compute the joint posterior distribution for general models.

[2]Expectation-Maximization

[3]Markov Chain Monte Carlo

tion, fixed lag smoothing or block sampling have also been suggested to improve the consistency of joint parameter and state estimation using particles [Olsson 08, Beevers 07, Fearnhead 02, Liu 01, Storvik 02, Doucet 06]. However, similarly to improved linearization techniques in Gaussian SLAM, the accumulated error remains, although the performance in some applications is better. For SLAM, where the observability of the system is reduced to a small set of features, artificial dynamics are specially bad, because most of the map is forgotten before revisiting.

The second key ingredient in Marginal-SLAM is being able to compute the filter derivative for the recursive ML. Furthermore, the estimation of the filter derivative has to overcome the curse of dimensionality that appears by conditioning on increasing trajectories. It has been proved that filter derivatives also presents *exponential forgetting* when the system is ergodic, for example, if it is dynamic [Tadic 05]. However, this has only become possible in practice very recently following new advances in particle simulation [Klaas 05, Poyadjis 05a, Coquelin 07], which will be presented in section 4.3.2. In contrast to other approaches that try to learn the parameters as a function of the full trayectory, we use the marginal particle filter (MPF), which actually provides samples from the filtering distribution $p(\mathbf{x}_t|\,\mathbf{y}_{1:t})$, instead of the full posterior distribution $p(\mathbf{x}_{1:t}|\mathbf{y}_{1:t})$.

Static parameter estimation in nonlinear, non-Gaussian dynamic models is a formidable challenge, so it comes as no surprise that probabilistic SLAM has proved to be so demanding. We remark that we adopt recursive ML estimates of the map because the problem of doing efficient full-Bayesian recursive parameter estimation has not yet been solved.

Our aim in this chapter is to compute sequentially in time the *filtering distribution* $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ and point estimates of the map θ. We use θ to describe the map only. However, θ could be used to include other parameters in the transition and measurement models, as well as, to include data association variables [Thrun 02]. For simplicity of presentation, we will assume that the associations are given.

4.2 Introduction to Sequential Monte Carlo

Let $\{\mathbf{x}_t^{(i)}\}_{i=1}^N$ be a set of samples (or *particles*) from $p(\mathbf{x}_t|\mathbf{y}_{1:t})$. Given those samples, we can approximate the filtering distribution with the Monte Carlo

estimate:

$$\widehat{p}(d\mathbf{x}_t|\mathbf{y}_{1:t}) = \frac{1}{N}\sum_{i=1}^{N}\delta_{\mathbf{x}_t^{(i)}}(d\mathbf{x}_t)$$

where $\delta_{\mathbf{x}_t^{(i)}}(d\mathbf{x}_t)$ denotes the delta Dirac function. Following the perfect Monte Carlo theorem, this estimate converges almost surely to the true expectation as N goes to infinity [Doucet 01]. Unfortunately, one cannot easily sample from the marginal distribution $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ directly. Instead, we draw particles from $p(\mathbf{x}_{1:t}|\mathbf{y}_{1:t})$ and samples $\mathbf{x}_{1:t-1}$ are ignored. This is a valid way to draw samples from a marginal distribution and is at the core of most Monte Carlo statistical methods. But the problem is even worse. The unknown normalizing constant, that is, the innovation defined in section 1.3, precludes us from sampling directly from the posterior. Instead, we draw samples from a proposal distribution q and weight the particles according to the following importance ratio:

$$w_t(\mathbf{x}_{1:t}) = \frac{p(\mathbf{x}_{1:t}|\mathbf{y}_{1:t})}{q(\mathbf{x}_{1:t}|\mathbf{y}_{1:t})}$$

This is called the importance sampling algorithm. But, when doing filtering, there is one extra limitation, data comes sequentially, thus we need to compute the distribution in the same way. For instance, the proposal distribution is constructed sequentially:

$$q(\mathbf{x}_{1:t}|\mathbf{y}_{1:t}) = q(\mathbf{x}_{1:t-1}|\mathbf{y}_{1:t-1})q(\mathbf{x}_t|\mathbf{y}_t, \mathbf{x}_{t-1})$$

and, hence, given a set of N particles $\{\mathbf{x}_{1:t-1}^{(i)}\}_{i=1}^N$, we obtain a set of particles $\{\mathbf{x}_{1:t}^{(i)}\}_{i=1}^N$ by sampling from $q(\mathbf{x}_t|\mathbf{y}_t, \mathbf{x}_{t-1}^{(i)})$ and keeping the old samples:

$$\left\{\mathbf{x}_t^{(i)}\right\}_{i=1}^N \sim q\left(\mathbf{x}_t|\mathbf{y}_t, \mathbf{x}_{t-1}^{(i)}\right)$$
$$\left\{\mathbf{x}_{1:t}^{(i)}\right\}_{i=1}^N = \left\{\mathbf{x}_t^{(i)}, \mathbf{x}_{1:t-1}^{(i)}\right\}_{i=1}^N$$

In the same way, the importance weights can be updated recursively in time:

$$w_t(\mathbf{x}_{1:t}) = \frac{p(\mathbf{x}_{1:t}|\mathbf{y}_{1:t})}{p(\mathbf{x}_{1:t-1}|\mathbf{y}_{1:t-1})q(\mathbf{x}_t|\mathbf{y}_t, \mathbf{x}_{t-1})}w_{t-1}(\mathbf{x}_{1:t-1}) \tag{4.1}$$

In conclusion, sequential Monte Carlo methods takes a set of N particles $\{\mathbf{x}_{1:t-1}^{(i)}\}_{i=1}^N$, to obtain a set of particles $\{\mathbf{x}_{1:t}^{(i)}\}_{i=1}^N$ by sampling from $q(\mathbf{x}_t|\mathbf{y}_t, \mathbf{x}_{t-1}^{(i)})$ and applying the weights of equation (4.1).

The familiar particle filtering equations for Markov models are obtained by remarking that:

$$p(\mathbf{x}_{1:t}|\mathbf{y}_{1:t}) \propto p\left(\mathbf{x}_{1:t}, \mathbf{y}_{1:t}\right) = \prod_{k=1}^{t} p(\mathbf{y}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{x}_{k-1}),$$

given which, equation (4.1) becomes:

$$\widetilde{w}_t^{(i)} \propto \frac{p(\mathbf{y}_t|\mathbf{x}_t^{(i)})p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})}{q(\mathbf{x}_t^{(i)}|\mathbf{y}_t, \mathbf{x}_{t-1}^{(i)})}\widetilde{w}_{t-1}^{(i)}$$

This iterative scheme produces a weighted measure $\{\mathbf{x}_{1:t}^{(i)}, w_t^{(i)}\}_{i=1}^N$, where $w_t^{(i)} = \widetilde{w}_t^{(i)} / \sum_j \widetilde{w}_t^{(j)}$, and is known as Sequential Importance Sampling (SIS). This is the procedure in common use by practitioners. It can be deceptive: although only the state $\mathbf{x}_t$ is being updated every round, the algorithm is nonetheless importance sampling in the *growing joint path space* $\mathcal{X}^t$.

It has been proved [Doucet 98] that *the variance of the importance weights in SIS increases over time*. This causes most particles to have very small probability. A common strategy to solve this *degeneracy*, consist in using a resampling step after updating the weights to replicate samples with high probability and prune those with negligible weight [Doucet 01]. This is called as the bootstrap filter of Sequential Importance Resampling (SIR).

The Curse of Dimensionality in Particle Filters for SLAM

The following theorems hold for any bounded Borel measurable function in a n-dimensional Borel space, i.e. $f \in B(\mathbb{R}^n)\|f\| = \sup_{z\in\mathbb{R}^n} f(z)$, provided that the Monte Carlo estimator is computed based on the bootstrap filter algorithm [Crisan 02]. For robotics, every continuous probability distribution, defined on a Euclidean space, is a especial case of the Borel measurable function.

Theorem 1. *[Crisan 02] For all $t \geq 0$, there exists c_t independent of N such that for any $f_t \in B\left((\mathbb{R}^{n_x})\right)^{t+1}$*

$$\mathbb{E}\left[\left(\frac{1}{N}\sum_{i=1}^{N} f_t\left(\mathbf{x}_{1:t}^{(i)}\right) - \int f_t\left(\mathbf{x}_{1:t}\right) p\left(\mathbf{x}_{1:t}|\mathbf{y}_{1:t}\right)\right)^2\right] \leq c_t \frac{\|f_t\|^2}{N}$$

We can see that the error bound is independent of the dimension of the state space n_x. Thus, Monte Carlo methods allows to overcome the curse of

dimensionality. However, without any additional assumption, the sequence c_t increases over time. Formally, the resampling step should be done along the full path $\{\mathbf{x}_{1:t}^{(i)}, w_{1:t}^{(i)}\}_{i=1}^{N}$, increasing the number of particles as time t increases to maintain the desired level of accuracy. In practice, as the number of particles increases exponentially, the system becomes intractable.

For dynamic systems, the previous constraint can be relaxed provided that the dynamic model is ergodic[4] and that we are interesting only on the filtering distribution.

Theorem 2. *[Crisan 02] For all $t \geq 0$, there exists a constant c independent of N such that for any $f_t \in B\left((\mathbb{R}^{n_x})\right)^{t+1}$*

$$\mathbb{E}\left[\left(\frac{1}{N}\sum_{i=1}^{N} f_t\left(\mathbf{x}_t^{(i)}\right) - \int f_t\left(\mathbf{x}_t\right) p\left(\mathbf{x}_t|\mathbf{y}_{1:t}\right)\right)^2\right] \leq c\frac{\|f_t\|^2}{N}$$

This is sometimes referred to as the *exponential forgeting* property of dynamic systems. Therefore, since dynamic systems forget the past exponentially fast, we can reduce the number of particles exponentially, compensating the previous exponential increase. Thus, several authors carry out resampling over the marginal space $\{\mathbf{x}_t^{(i)}, w_t^{(i)}\}_{i=1}^{N}$ with a fixed number of particles. In that way, dynamic systems compensates the exponential complexity due to the temporal dimensions added. This would be fine if, for example, we were interested in tracking *only* dynamic elements [Doucet 00].

Static parameter estimation and model selection problems do not necessarily exhibit an exponential forgetting behavior. In the case where a (random) fixed parameter is part of the state, the "dynamic" model is not ergodic, and it is thus expected that whatever the particle filtering one uses, one cannot obtain uniform convergence results. In practice, it has been observed that as time increases, such algorithms indeed diverge [Andrieu 99, Crisan 02]. Intuitively, estimates of static maps features depend on all the observations made from different robot locations. Thus, it depends on the whole robot trajectory. Resampling these trajectories in the joint path space using a fixed number of particles is guaranteed to deplete the past in finite time; since there is only a limited number of trajectories. Alternatively, resampling from the marginal space still leaves us with an accumulation of Monte Carlo errors over time. The estimate of the map will depend on this increasing sequence of errors,

[4]Intuitively, we can see an ergodic system as any dynamic system that keeps on moving or changing. That is, for every time step, the system state is completely different from the previous state, $\mathbf{x}_t \cap \mathbf{x}_{t-1} = \emptyset$.

that are never *forgotten*. Some implementations have introduced artificial dynamics or Markov chain Monte Carlo (MCMC) rejuvenation steps to reduce the severity of the problem [Liu 01]. In the context of SLAM, some heuristics has been proposed to exploit the structural properties of the problem [Stachniss 05b, Eliazar 05]. But these approaches do not overcome the problem.

Furthermore, to ensure a given precision on the mean square error, the number of particles N also depends on c_t or c, which may depend on n_x. However, as seen in previous sections, SLAM problem requires sampling in an exponentially increasing state-space n_x which becomes infeasible for a tractable number of particles. Intuitively, the bootstrap filter overcomes the curse of dimensionality in the sense that convergency is guaranteed for no matter how many dimensions include the state-space. As the number of dimensions increases, the required number of particles does not grows exponentially fast. However, to maintain a certain level of accuracy as the state-space increses, the required number of particles still grows. Therefore, many high dimensional problems may become intractable for real-time applications.

But there are methods to reduce the dimensionality of the problem. Murphy [Murphy 99] proposed a technique for the problem of mapping dynamic environments where samples are drawn only from the robot configuration space, which is a lower, and also fixed, dimensional state-space. This method, named Rao-Blackwellized particle filtering [Doucet 00] is based on the following decomposition of the joint posterior:

$$p(\theta, \mathbf{x}_{1:t}|\mathbf{y}_{1:t}) = p(\theta|\mathbf{x}_{1:t}, \mathbf{y}_{1:t})p(\mathbf{x}_{1:t}|\mathbf{y}_{1:t})$$

Consequently, given the state path $\mathbf{x}_{1:t}$, and assuming that the distribution $p(\theta|\mathbf{x}_{1:t}, \mathbf{y}_{1:t})$ is Gaussian and the observation model is linear, then we can solve the map parameters θ analytically, for instance, using RLS[5]. This leaves us with only having to carry out particle filtering to compute the posterior distribution of the robot state $p(\mathbf{x}_{1:t}|\mathbf{y}_{1:t})$, which has a spatially bounded dimension.

This factorization is even better for SLAM, because it can be shown that every map feature is independent given the trajectory of the robot. That is, if we condition the features on the robot trajectory, every feature can be updated

[5]In the literature, this is usually referred to as EKF due to the analogy with Gaussian SLAM, but it should be noted that we are not doing filtering in the map, because it is purely static. Other authors may be confused with the original derivation of the RLS algorithm as a frequentist approach. But RLS updates both the posterior mean and covariance, used for the step size. Then, the belief is completely defined.

separately:

$$p(\theta, \mathbf{x}_{1:t}|\mathbf{y}_{1:t}) = p(\theta|\mathbf{x}_{1:t}, \mathbf{y}_{1:t})p(\mathbf{x}_{1:t}|\mathbf{y}_{1:t}) = \prod_{j=1}^{M} p(\theta_j|\mathbf{x}_{1:t}, \mathbf{y}_{1:t})p(\mathbf{x}_{1:t}|\mathbf{y}_{1:t})$$

Even if the parameters distribution is non-Gaussian or nonlinear, we can use other suboptimal methods to find an analytical approximation. In constrast, the conditional independency property holds for any SLAM problem with onboard sensors.

Nevertheless, the static parameters are still not an ergodic systems but they are conditionally based on a sampled distribution. Therefore, the algorithm may diverge. *In conclusion, whether we resample or not, in full state or Rao-Blackwellized particle filters, learning the static map as a function of a growing path in Monte Carlo simulation is a bad idea.*

4.3 Parameter learning for SLAM

Then, if we want to include fixed parameters in the estimation algorithm, we have to learn it as proper parameters. Up to now, no full Bayesian solution has been found to the joint problem of filtering and on-line parameter learning. However, there are approximations in the statistical literature that combine Bayesian estimates with frequentist point-based optimization. The most extended one being the maximum likelihood (or maximum a posteriory) stochastic approximation.

4.3.1 Stochastic Approximation for SLAM

One of the most general techniques for parameter learning is stochastic approximation [Spall 03], which can be formulated as

$$\theta_t = \theta_{t-1} + \gamma_t V_t(\theta_t, \mathbf{x}_{1:t}) \tag{4.2}$$

where $\{\mathbf{x}_t\}_{t\geq 1}$ is an stochastic process with information about the parameters, γ_t is a discount factor[6], and $V_t(\theta_t, \cdot)$ being a mapping from the joint space to the parameter space. In the adaptive control and signal processing literature, the $\{\mathbf{x}_t\}_{t\geq 1}$ process is sometimes addressed as the *observation* process. However, in this context, the name can be deceptive since it is, in fact, the latent process

[6]To guarantee convergency, the discount factor should be defined such us ($\gamma_t \to 0, \sum \gamma_t = +\infty$), although for practical reasons, that requirements can be relaxed. See [Spall 03] for a complete description about this parameter.

$\{\mathbf{x}_t\}_{t\geq 1}$ and not the actual measurement process $\{\mathbf{y}_t\}_{t\geq 1}$. In SLAM, the latent process is the robot trajectory and the parameters are the elements of the map. The idea is that the estimate of the state contains all the information required to update the parameters.

It is important to note that the temporal subindex in the parameters estimate θ_t does not mean an actual evolution of the true parameters $\boldsymbol{\theta}^\star$ but the iterative optimization process. However, the t subindex in the iteration matches the temporal evolution of the state, because recursive stochastic approximation only updates the parameters when new information is available. That is, the estimate of the parameter changes every time, but the actual parameter remains fixed.

In our case, provided that we can estimate the latent process $\{\mathbf{x}_t\}_{t\geq 1}$ using some algorithm (e.g.: the Monte Carlo method that we will explain in subsequent sections), then we can access the set of pairs $\{\mathbf{x}_1, \mathbf{y}_1\}, \{\mathbf{x}_2, \mathbf{y}_2\}, \ldots, \{\mathbf{x}_t, \mathbf{y}_t\}$ and compute the parameters using maximum likelihood of those pairs.

We follow the approach in [LeGland 97] for recursive maximum likelihood (ML) in state space models. First, we consider the set of log-likelihood[7] functions

$$L_t(\theta, \cdot) = \frac{1}{t+1}\{\log p_\theta(\mathbf{y}_{1:t})\}_{t\geq 0}$$

then, the true parameters $\boldsymbol{\theta}^\star$ are the global maximum of $L(\theta, \cdot)$, which is the limit of the average log-likelihood:

$$L(\theta, \cdot) = \lim_{t\to\infty} L_t(\theta, \cdot) \tag{4.3}$$

$$= \lim_{t\to\infty} \frac{1}{t+1}\{\log p_\theta(\mathbf{y}_{1:t})\}_{t\geq 0} \tag{4.4}$$

In general, the function $L(\theta, \cdot)$ does not have an analytical expression and we do not have access to it. However, we can still find the parameters $\boldsymbol{\theta}^\star$, provided that

$$\log p_\theta(\mathbf{y}_{1:t}) = \sum_{k=1}^{t} \log p_\theta(\mathbf{y}_k|\mathbf{y}_{1:k})$$

which represent the objetive function as a set of accessible functions $p_\theta(\mathbf{y}_t|\mathbf{y}_{1:t-1})$ that converge to $L(\theta, \cdot)$. As introduced in previous sections, the expresion $p_\theta(\mathbf{y}_t|\mathbf{y}_{1:t-1})$ is called the innovation and can be written as

$$p_\theta(\mathbf{y}_t|\mathbf{y}_{1:t-1}) = \int\int p_\theta(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1})p_\theta(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})d\mathbf{x}_{t-1:t} \tag{4.5}$$

[7]Since we are interesting in maximize the likelihood function, it is equivalent to maximize the log-likelihood. However, the later is frequently used because is analytically easier.

which can be obtained from the filtering process. The subindex θ of certain probabilities represents that those probabilies are computed as a function of the parameters, but we are doing filtering on the state space which includes only the robot locations. Although $p_\theta(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = p(\mathbf{x}_t, \theta|\mathbf{y}_{1:t-1})$, we stress the idea that the map is not part of the filtering process, but it is computed separately.

Once we have defined the optimization process in terms of accesible functions, the parameters θ can be recursively approximated using only the current marginal innovation $p_\theta(\mathbf{y}_t|\mathbf{y}_{1:t-1})$. The simplest way to implement this optimization in a stochastic approximation framework is to implement a simple gradient ascent method

$$\theta_t = \theta_{t-1} + \gamma_t \nabla_\theta \log p_\theta(\mathbf{y}_t|\mathbf{y}_{1:t-1}) \tag{4.6}$$

It can be proved that, under certain conditions, θ_t converges to true parameters $\boldsymbol{\theta}^\star$. The convergency of the algorithm will be explained in section 4.5.1. A detailed analysis is presented in [Tadic 05].

But we can also implement an adaptive stochastic approximation method, which provides a Newton-type method based on a recursive calculation of the sample mean of the per-iteration Hessian $\mathcal{H}_t = \nabla^2_\theta \log p_\theta(\mathbf{y}_t|\mathbf{y}_{1:t-1})$. That is, we have two parallel recursions: one for θ and one for the Hessian of the log-likelihood $\bar{\mathcal{H}}$ which are

$$\theta_t = \theta_{t-1} + \gamma_t \psi(\bar{\mathcal{H}}_t)^{-1} \nabla_\theta \log p_\theta(\mathbf{y}_t|\mathbf{y}_{1:t-1}) \tag{4.7}$$

$$\bar{\mathcal{H}}_t = \frac{k}{1+k}\bar{\mathcal{H}}_{t-1} + \frac{1}{1+k}(\mathcal{H}_t) \tag{4.8}$$

where γ_t is a new sequence of learning coeficients and $\psi(\cdot)$ is a mapping to the set of negative definite matrices, which guarantees that the matrix is invertible. This function, which is sometimes called *regularizer* is just necessary in practice, where the approximations of the innovation and the per-iteration Hessian may incurr in numerical errors.

The advantage of using stochastic approximation (SA) methods against other Maximum Likelihood methods is that, SA operates under weaker conditions. In particular, SA does not require assumptions about the filtering or the innovation distribution.

Example 1 (Relationship with RLS and KF)**.** *For a linear observation model*

$$\mathbf{y}_t = \mathbf{H} \begin{bmatrix} \mathbf{x}_t \\ \theta_t \end{bmatrix} + \mathbf{w}_t$$

the innovation function can be computed analytically

$$\nabla_\theta \log p_\theta(\mathbf{y}_t|\mathbf{y}_{1:t-1}) = \mathbf{H}^T \left\{ \mathbf{H} \begin{bmatrix} \mathbf{x}_t \\ \theta_t \end{bmatrix} - \mathbf{y}_t \right\}$$

Then we can apply the stochastic approximation algorithm directly as a gradient ascent optimization:

$$\theta_t = \theta_{t-1} + \gamma_t \mathbf{H}^T \left\{ \mathbf{H} \begin{bmatrix} \mathbf{x}_t \\ \theta_t \end{bmatrix} - \mathbf{y}_t \right\}$$

However, if the underlying distributions are Gaussian, i.e. the state and parameters follow a distribution

$$\begin{bmatrix} \mathbf{x}_t \\ \theta_t \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \widehat{\mathbf{x}}_t \\ \widehat{\theta}_t \end{bmatrix}, \mathbf{P}_t \right)$$

where $\widehat{\mathbf{x}}_t$ *and* $\widehat{\theta}_t$ *are the mean of the distribution w.r.t. the state and the parameters respectively and* $\mathbf{P}_t$ *is the joint state and parameters covariance at time* t*. Then, the cost function becomes quadratic and the optimal step size comes from the Newton-Rapson method which relies on the Hessian of the cost function. This is the idea behind RLS estimate:*

$$\theta_t = \theta_{t-1} + \mathcal{H}_t^{-1} \mathbf{H}^T \left\{ \mathbf{H} \begin{bmatrix} \mathbf{x}_t \\ \theta_t \end{bmatrix} - \mathbf{y}_t \right\} \tag{4.9}$$

$$\simeq \theta_{t-1} + \mathbf{P}_t \mathbf{H}^T \left\{ \mathbf{H} \begin{bmatrix} \mathbf{x}_t \\ \theta_t \end{bmatrix} - \mathbf{y}_t \right\} \tag{4.10}$$

The discrepancy between Newton-Rapson, equation (4.9)*, and RLS, equation* (4.10)*, is due to the assumption that the prior distribution is noninformative compared to the posterior* $\mathbf{P}_0 \gg \mathbf{P}_t$ *and the assumption that the current estimate* θ_t *is the optimal solution to the current cost function* $L_t(\theta, \cdot)$ *[Spall 03].*

We have described earlier that the Kalman filter for parameter estimation is equivalent to RLS for parameter learning. Also, as stated before, the frequentist and Bayesian approach for optimal parameter learning assuming Gaussian distributions is equivalent. Therefore, RLS and KF for parameter estimation are equivalent to the stochastic approximation for linear-Gaussian systems with quasi-optimal Newton-type steps:

$$\theta_t = \theta_{t-1} + \mathbf{K}_t \left\{ \mathbf{H} \begin{bmatrix} \mathbf{x}_t \\ \theta_t \end{bmatrix} - \mathbf{y}_t \right\}$$

where $\mathbf{K}_t$ *is the Kalman filter gain.*

Nevertheless, the innovation for general nonlinear systems, and also in SLAM, is intractable. Therefore, we need to find a suitable numerical approximation of the gradient and hessian of the log-likelihood, $\nabla_\theta \log \widehat{p_\theta}(\mathbf{y}_t|\mathbf{y}_{1:t-1})$ and $\nabla^2_\theta \log \widehat{p_\theta}(\mathbf{y}_t|\mathbf{y}_{1:t-1})$ respectively. Thus, the only remaining question is how to compute the gradient of the innovation distribution using Monte Carlo methods. But for the time being, we only know how to use Monte Carlo in a filtering scheme. The question is: Can we obtain the gradient innovation function as a function of the filtering distribution $p_\theta(\mathbf{x}_t|\mathbf{y}_{1:t})$?

First of all, to simplify the exposition later on, we introduce the following notation [Poyadjis 05a]:

$$p_\theta(\mathbf{x}_t|\mathbf{y}_{1:t}) \triangleq \frac{\xi_\theta(\mathbf{x}_t, \mathbf{y}_{1:t})}{\int \xi_\theta(\mathbf{x}_t, \mathbf{y}_{1:t}) d\mathbf{x}_t} \tag{4.11}$$

where ξ represents the unnormalized filtering distribution. Thus,

$$\xi_\theta(\mathbf{x}_t, \mathbf{y}_{1:t}) = p_\theta(\mathbf{y}_t|\mathbf{x}_t) p_\theta(\mathbf{x}_t|\mathbf{y}_{1:t-1}) \tag{4.12}$$

$$= p_\theta(\mathbf{y}_t|\mathbf{x}_t) \int p(\mathbf{x}_t|\mathbf{x}_{t-1}) p_\theta(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1} \tag{4.13}$$

which is related to the innovation function $p_\theta(\mathbf{y}_t|\mathbf{y}_{1:t-1}) = \int \xi_\theta(\mathbf{x}_t, \mathbf{y}_{1:t}) d\mathbf{x}_t$. Using expansions of the derivatives of logs[8], the gradient of $\xi_\theta(.,.)$ with respect to the map parameters θ is

$$\nabla_\theta \xi_\theta(\mathbf{x}_t, \mathbf{y}_{1:t}) = p_\theta(\mathbf{y}_t|\mathbf{x}_t) \nabla_\theta \log p_\theta(\mathbf{y}_t|\mathbf{x}_t) \int p(\mathbf{x}_t|\mathbf{x}_{t-1}) p_\theta(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1}$$
$$+ p_\theta(\mathbf{y}_t|\mathbf{x}_t) \int p(\mathbf{x}_t|\mathbf{x}_{t-1}) \nabla_\theta p_\theta(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1}$$

which is equivalent to

$$\nabla_\theta \xi_\theta(\mathbf{x}_t, \mathbf{y}_{1:t}) = \xi_\theta(\mathbf{x}_t, \mathbf{y}_{1:t}) \nabla_\theta \log p_\theta(\mathbf{y}_t|\mathbf{x}_t) \tag{4.14}$$

$$+ p_\theta(\mathbf{y}_t|\mathbf{x}_t) \int p(\mathbf{x}_t|\mathbf{x}_{t-1}) \nabla_\theta p_\theta(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1} \tag{4.15}$$

These equations can be updated recursively knowing that the derivative of the previous filtering distribution $\nabla_\theta p_\theta(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})$ can be rewriten as a function of the previous unnormalized functions. Applying derivatives to equation

[8] For maximum likelihood, it is very common to use the *score function*, which is the partial derivative, with respect to some parameters θ, of the logarithm of the likelihood function. Following the chain rule $\nabla_\theta \log p_\theta(\mathbf{y}_t|\mathbf{x}_t) = \nabla_\theta p_\theta(\mathbf{y}_t|\mathbf{x}_t) / p_\theta(\mathbf{y}_t|\mathbf{x}_t)$.

(4.11), we have

$$\nabla_\theta p_\theta(\mathbf{x}_t|\mathbf{y}_{1:t}) = \frac{\nabla_\theta \xi_\theta(\mathbf{x}_t, \mathbf{y}_{1:t})}{\int \xi_\theta(\mathbf{x}_t, \mathbf{y}_{1:t}) d\mathbf{x}_t} - p_\theta(\mathbf{x}_t|\mathbf{y}_{1:t}) \frac{\int \nabla_\theta \xi_\theta(\mathbf{x}_t, \mathbf{y}_{1:t}) d\mathbf{x}_t}{\int \xi_\theta(\mathbf{x}_t, \mathbf{y}_{1:t}) d\mathbf{x}_t}. \tag{4.16}$$

Finally, we can manipulate the score of the innovation function

$$\nabla_\theta \log p_\theta(\mathbf{y}_t|\mathbf{y}_{1:t-1}) = \frac{\nabla_\theta p_\theta(\mathbf{y}_t|\mathbf{y}_{1:t-1})}{p_\theta(\mathbf{y}_t|\mathbf{y}_{1:t-1})} = \frac{\int \nabla_\theta \xi_\theta(\mathbf{x}_t, \mathbf{y}_{1:t}) d\mathbf{x}_t}{\int \xi_\theta(\mathbf{x}_t, \mathbf{y}_{1:t}) d\mathbf{x}_t} \tag{4.17}$$

which can be recursively computed using equations (4.13) and (4.15).

In the next section, we are going to see how to approximate these functions and their gradients using Monte Carlo methods.

4.3.2 Monte Carlo Implementation

The score of the innovation function from equation (4.17) can be expresed as a function of the unnormalized filtering distribution $\xi_\theta(\mathbf{x}_t, \mathbf{y}_{1:t})$. Thus, we need to draw samples from the marginal space $\mathbf{x}_t$. However, classical Monte Carlo methods generates samples on the joint path space $\mathbf{x}_{1:t}$. This technique will fail, and more advanced Monte Carlo methods are needed.

The Joint Path Space Approach

As commented before, the basic Sequential Monte Carlo algorithm assumes that the system is ergodic and consequently, it can be approximated with a finite set of particles using a resampling step (SIR) after updating the weights to replicate samples with high probability and prune those with negligible weight [Doucet 01]. Formally, the resampling step should be done along the full path $\{\mathbf{x}_{1:t}^{(i)}, w_{1:t}^{(i)}\}_{i=1}^N$. Since dynamic systems forget the past exponentially fast, several authors carry out resampling over the marginal space $\{\mathbf{x}_t^{(i)}, w_t^{(i)}\}_{i=1}^N$. This would be fine if, for example, we were interested in tracking dynamic maps.

The same degeneracy problem arises if we try to obtain estimates of the filter derivative $\nabla_\theta p_\theta(\mathbf{x}_t|\mathbf{y}_{1:t})$, or the unnormalized distribution $\widehat{\nabla_\theta \xi_\theta}(\mathbf{x}_t, \mathbf{y}_{1:t})$, for recursive (online) stochastic approximation. To see this, let $\nabla_\theta p_\theta(\mathbf{x}_{1:t}|\mathbf{y}_{1:t})$ denote the gradient vector of the path posterior with respect to the map, which can be rewritten as:

$$\nabla_\theta p_\theta(\mathbf{x}_{1:t}|\mathbf{y}_{1:t}) = \frac{\nabla_\theta p_\theta(\mathbf{x}_{1:t}|\mathbf{y}_{1:t})}{p_\theta(\mathbf{x}_{1:t}|\mathbf{y}_{1:t})} p_\theta(\mathbf{x}_{1:t}|\mathbf{y}_{1:t})$$

and, consequently the filter derivative, necessary for online map learning, is given by:

$$\nabla_\theta p_\theta(\mathbf{x}_t|\mathbf{y}_{1:t}) = \int_{\mathcal{X}^{t-1}} \frac{\nabla_\theta p_\theta(\mathbf{x}_{1:t}|\mathbf{y}_{1:t})}{p_\theta(\mathbf{x}_{1:t}|\mathbf{y}_{1:t})} p_\theta(\mathbf{x}_{1:t}|\mathbf{y}_{1:t}) d\mathbf{x}_{1:t-1} \tag{4.18}$$

Using standard particle filters to approximate the filter derivative we are implicitly carrying out importance sampling on a vast growing space with proposal $p_\theta(\mathbf{x}_{1:t}|\mathbf{y}_{1:t})$ *and weight* $\frac{\nabla_\theta p_\theta(\mathbf{x}_{1:t}|\mathbf{y}_{1:t})}{p_\theta(\mathbf{x}_{1:t}|\mathbf{y}_{1:t})}$. This should be enough reason to call for a new approach. Yet, the problem is even worse.

The filter derivative is a signed-measure, and not a standard probability measure. It consists of positive and negative functions over disjoint parts of the state space and it sums to zero over the entire state space. A serious problem, when carrying out classical particle filtering to estimate this signed-measure, is that particles with positive and negative weights will cancel each other, say, in parts of the space where the derivative is close to zero (figure 4.1, top). This is computationally wasteful and statistically harmful.

The technique presented in the following section overcomes these deficiencies.

The Marginal Space Approach

To eliminate the problems discussed in the previous section, we will perform particle filtering *directly* on the marginal distribution $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ instead of on the joint space. In the literature, this technique is called the Marginal Particle Filter (MPF) [Fearnhead 98, Pitt 99, Klaas 05, Poyadjis 05b, Poyadjis 05a]. To do so, we begin by remembering that the predictive density can be obtained by marginalization:

$$p_\theta(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1}) p_\theta(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1} \tag{4.19}$$

using the Chapman-Kolmogorov equation (section 1.3).

Assume that at time $t-1$, we have a Monte Carlo approximations of the filter and its gradient. We denote the normalized and unnormalized filter and

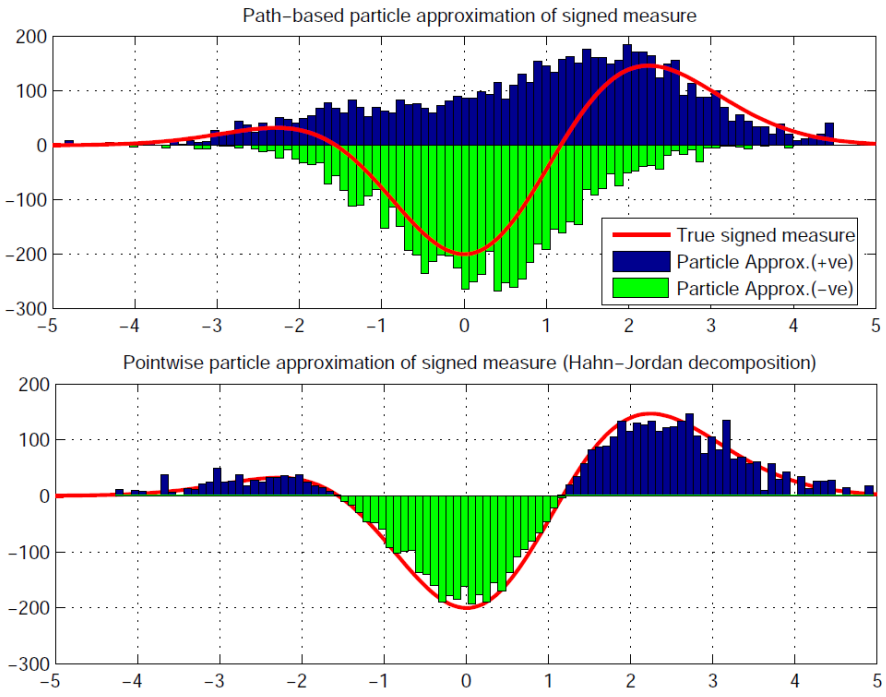


Figure 4.1: Top plot: Histogram representation of a path-based particle approximation of $\nabla_\theta p_\theta(\mathbf{x}_t|\mathbf{y}_{1:t})$ w.r.t. a one-dimensional parameter θ. Bottom plot: Point-wise particle approximation of the same signed measure that maintains the positive and negative weights on separate regions of the state support (reproduced with permission from [Poyadjis 05a]).

gradient approximations by:

$$\widehat{p_\theta}(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) = \sum w_{t-1}^{(i)} \delta_{\mathbf{x}_{t-1}^{(i)}}$$

$$\widehat{\xi}_\theta(\mathbf{x}_t, \mathbf{y}_{1:t}) = \frac{1}{N} \sum \widetilde{w}_t^{(i)} \delta_{\mathbf{x}_t^{(i)}},$$

$$\widehat{\nabla_\theta p_\theta}(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) = \sum_{i=1}^{N} w_{t-1}^{(i)} \beta_{t-1}^{(i)} \delta_{\mathbf{x}_{t-1}^{(i)}}(\mathbf{x}_{t-1})$$

$$\widehat{\nabla_\theta \xi_\theta}(\mathbf{x}_t, \mathbf{y}_{1:t}) = \frac{1}{N} \sum_{i=1}^{N} \widetilde{\rho}_t^{(i)} \delta_{\mathbf{x}_t^{(i)}}(\mathbf{x}_t).$$

The integral in equation (4.19) is generally not solveable analytically, but since

we have the particle approximation, we can approximate it as the weighted kernel density estimate

$$\widehat{p\,_\theta}(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \sum_{j=1}^{N} w_{t-1}^{(j)} p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(j)}).$$

This is another example of kernel smoothing for density estimation as introduced in section 2.3.1. In this case, the kernel is the transition distribution or motion model.

The main idea behind Marginal Particle Filter is to sample directly from the marginal space, that is, we need a marginal proposal. While we are free to choose any proposal distribution that has appropriate support, it is convenient to assume that the marginal proposal takes a similar form of weighted kernel density estimate, namely

$$q_\theta(\mathbf{x}_t|\mathbf{y}_{1:t}) = \sum_{j=1}^{N} w_{t-1}^{(j)} q_\theta(\mathbf{x}_t|\mathbf{y}_t, \mathbf{x}_{t-1}^{(j)}).$$

We can easily draw particles from this proposal, provided that we know how to sample from $q_\theta(\mathbf{x}_t|\mathbf{y}_t, \mathbf{x}_{t-1}^{(j)})$, using multinomial or stratified sampling [Kitagawa 96]. Figure 4.2 shows the difference between marginal sampling and joint path sampling.

Once we have samples in the marginal space, we can compute the new unnormalized importance weights as:

$$\widetilde{w}_t^{(i)} = \frac{p_\theta(\mathbf{y}_t|\mathbf{x}_t^{(i)}) \sum_{j=1}^{N} w_{t-1}^{(j)} p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(j)})}{q_\theta(\mathbf{x}_t^{(i)}|\mathbf{y}_{1:t})}$$

$$\widetilde{\rho}_t^{(i)} = \widetilde{w}_t^{(i)} \nabla_\theta \log p_\theta(\mathbf{y}_t|\mathbf{x}_t^{(i)}) + \frac{\sum_j w_{t-1}^{(j)} p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(j)}) \beta_{t-1}^{(j)}}{q_\theta(\mathbf{x}_t^{(i)}|\mathbf{y}_{1:t})}$$

using equations (4.11) and (4.15). Finally, substituting the above Monte Carlo estimates into the expression for the derivative of p_θ in terms of ξ_θ, we obtain the normalized weights at time t.

$$w_t^{(i)} = \frac{\widetilde{w}_t^{(i)}}{\sum_j \widetilde{w}_t^{(j)}} \; ; \; w_t^{(i)} \beta_t^{(i)} = \frac{\widetilde{\rho}_t^{(i)}}{\sum_j \widetilde{w}_t^{(j)}} - w_t^{(i)} \frac{\sum_j \widetilde{\rho}_t^{(j)}}{\sum_j \widetilde{w}_t^{(j)}}$$

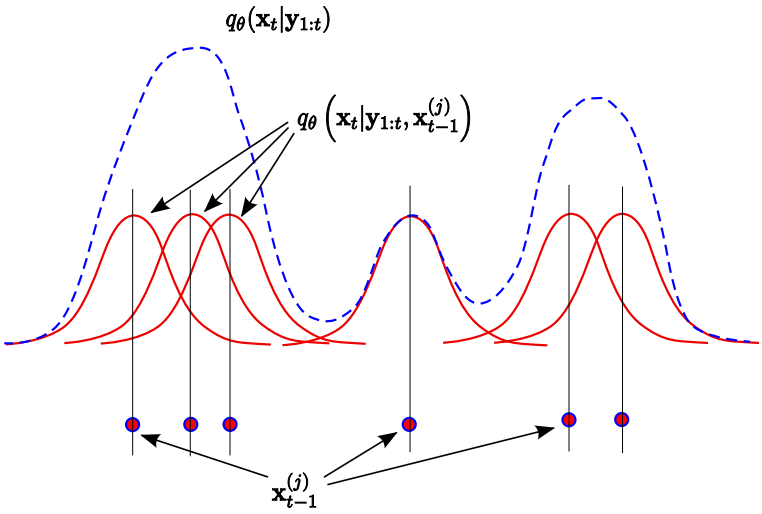


Figure 4.2: Joint path sampling takes every particle, that is, every path, and extend the Markov chain using the corresponding proposal. Then, every sample independently extends its path. It can be seen that any particle generates a single particle from the corresponding proposal (red solid lines). Marginal sampling computes the joint proposal using kernel smoothing (blue dashed line). Then, all the samples are drawn jointly from that function.

Then, we can approximate score function from equation (4.17) by the correspoind Monte Carlo approximation:

$$\nabla_\theta \log \widehat{p_\theta}(\mathbf{y}_t|\mathbf{y}_{1:t-1}) = \frac{\int \widehat{\nabla_\theta \xi_\theta}(\mathbf{x}_t, \mathbf{y}_{1:t}) d\mathbf{x}_t}{\int \widehat{\xi_\theta}(\mathbf{x}_t, \mathbf{y}_{1:t}) d\mathbf{x}_t} = \frac{\sum_j \widetilde{w}_t^{(j)}}{\sum_j \widetilde{\rho}_t^{(j)}} \tag{4.20}$$

Note the advantages of marginal filtering. First, *the importance sampling process now happens in the marginal space*. In addition, the last integral in equation (4.15) can be expanded using the score identity:

$$\int p(\mathbf{x}_t|\mathbf{x}_{t-1}) \nabla_\theta \log[p_\theta(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})] p_\theta(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1}$$

That is we sample from the marginal filtering distribution and weight with $\beta \triangleq \nabla_\theta \log[p_\theta(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})]$. Contrast this with equation (4.18). The other thing to note, as pointed out in [Poyadjis 05a] is that the marginal filter derivative allows us to obtain a particle approximation of the Hahn-Jordan decomposition. This implies that samples are drawn in disjoint regions of the state

Marginal-SLAM

- For $i = 1, ..., N$, sample the robot state from the proposal

$$\mathbf{x}_t^{(i)} \sim \sum_{j=1}^{N} w_{t-1}^{(j)} q(\mathbf{x}_t | \mathbf{y}_t, \mathbf{x}_{t-1}^{(j)})$$

- For $i = 1, ..., N$, evaluate the importance weights

$$\widetilde{w}_t^{(i)} = \frac{p_\theta(\mathbf{y}_t|\mathbf{x}_t^{(i)}) \sum_{j=1}^{N} w_{t-1}^{(j)} p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(j)})}{q_\theta(\mathbf{x}_t^{(i)}|\mathbf{y}_{1:t})}$$

$$\widetilde{\rho}_t^{(i)} = \widetilde{w}_t^{(i)} \nabla_\theta \log p_\theta(\mathbf{y}_t|\mathbf{x}_t^{(i)}) + \frac{\sum_j w_{t-1}^{(j)} p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(j)}) \beta_{t-1}^{(j)}}{q_\theta(\mathbf{x}_t^{(i)}|\mathbf{y}_{1:t})}$$

- Normalise the importance weights

$$w_t^{(i)} = \frac{\widetilde{w}_t^{(i)}}{\sum_j \widetilde{w}_t^{(j)}} \; ; \; w_t^{(i)} \beta_t^{(i)} = \frac{\widetilde{\rho}_t^{(i)}}{\sum_j \widetilde{w}_t^{(j)}} - w_t^{(i)} \frac{\sum_j \widetilde{\rho}_t^{(j)}}{\sum_j \widetilde{w}_t^{(j)}}$$

- Update the map vector

$$\theta_t = \theta_{t-1} + \gamma_t \frac{\sum_j \widetilde{\rho}_t^{(j)}}{\sum_j \widetilde{w}_t^{(j)}}$$

- Update the learning rate γ_t.

Figure 4.3: The Marginal-SLAM algorithm at time t.

space. Then, we can surmount the problem of particles of opposite signs cancelling each other out in infinitesimal neighborhoods of the state space (figure 4.1, bottom).

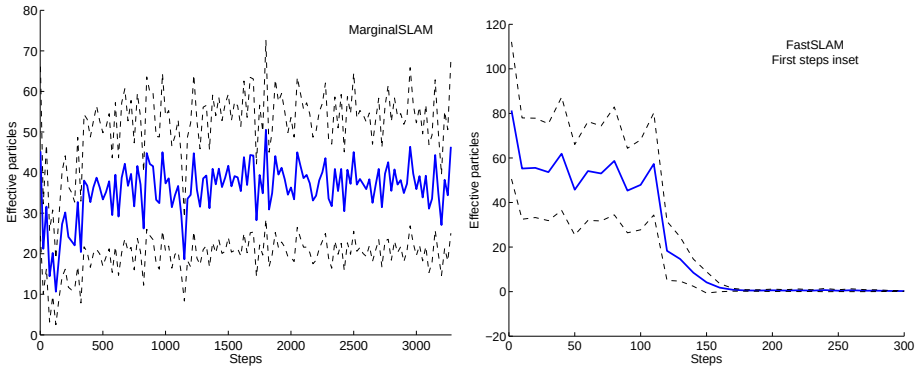


Figure 4.4: The plots represent the average number of effective particles N_{eff}, with corresponding confidence intervals, in 25 long term simulations. The total number of particles is 200 in both cases. The left plot corresponds to Marginal-SLAM, where the filter achieves a steady state with a 25% effective number of particles, allowing to update the same are for indefinite tima. The right plot shows the same experiment for FastSLAM. It quickly drops in every simulation and fails after 200 steps. Note that the time scale in the Marginal-SLAM plot is even longer, but the results remains steady.

4.3.3 Pseudo-Code for Marginal-SLAM

The Marginal-SLAM algorithm is depicted in Figure 4.3. Note that it is linear in the number of features. It has an $\mathcal{O}(N^2)$ complexity in terms of the number of samples, but this can be reduced to $\mathcal{O}(N \log N)$ using the fast multipole expansions and metric tree recursions proposed in [Klaas 05].

The marginal particle filter is an old idea [Fearnhead 98, Pitt 99]. Yet, because of its large computational cost, it was not fully explored until the introduction of fast methods [Klaas 05]. When using the transition prior as proposal, the marginal filter and classical particle filter are equivalent, but this is no longer true when computing the derivative of the filter as outlined in [Poyadjis 05b] and this paper.

4.4 Experiments

We compare Marginal-SLAM and FastSLAM in a large scale, highly noisy simulated environment with known data association. The environment is a square-like corridor with point landmarks in the walls. For simplicity, we use the transition prior as the proposal distribution for both techniques. In future

work, optimal proposals could be considered like in FastSLAM 2.0.

One of the main SLAM difficulties is partial observability of the parameters. The choice of learning rates γ_t is affected by this partial observability. In our case, we chose to implement separate learning rates for each landmark. Each individual rate depends on the number of times its corresponding feature is observed.

The robot motion, that is, the transition model, is based on a simple differential drive vehicle like the one used in other simulated experiments in this thesis. The observations are gathered using range and bearing sensors $\mathbf{y}_t = [\rho, \phi]$ with a point feature detector

$$\begin{bmatrix} \hat{\rho} \\ \hat{\phi} \end{bmatrix} = \begin{bmatrix} \sqrt{\Delta_x^2 + \Delta_y^2} \\ \arctan\left(\frac{-\Delta_x \sin(\varphi_t) + \Delta_y \cos(\varphi_t)}{\Delta_x \cos(\varphi_t) + \Delta_y \sin(\varphi_t)}\right) \end{bmatrix}$$

where $\Delta_x = \theta_x - X_t$ and $\Delta_y = \theta_y - Y_t$; with $[\theta_x, \theta_y]$ and $[X_t, Y_t, \varphi_t]$ denoting the feature and robot location and respectively[9]. The sensor has white Gaussian noise $v_t \sim \mathcal{N}(0, diag(\sigma_\rho, \sigma_\phi))$.

Hence, the likelihood function for a single feature is

$$p(\rho, \phi | \mathbf{x}_t) = \frac{1}{\sqrt{2\pi}\sigma_\rho \sigma_\phi} exp\left[-\frac{1}{2}\left(\frac{(\rho - \hat{\rho})^2}{\sigma_\rho^2} + \frac{(\phi - \hat{\phi})^2}{\sigma_\phi^2}\right)\right]$$

Thus, the gradient of the log-likelihood corresponds to

$$\nabla \log(p(\rho, \phi | \mathbf{x}_t)) = \frac{1}{\rho}\begin{bmatrix} \frac{\Delta_x(\rho - \hat{\rho})}{\sigma_\rho^2} - \frac{\Delta_y(\phi - \hat{\phi})}{\rho\sigma_\phi^2} \\ \frac{\Delta_y(\rho - \hat{\rho})}{\sigma_\rho^2} + \frac{\Delta_x(\phi - \hat{\phi})}{\rho\sigma_\phi^2} \end{bmatrix}$$

We carried out several simulations by varying the amount of sensor and motion noise, landmark density and loop size. The system is able to close large loops with large range and bearing noise. However, very large sensor noise (*e.g.* sonar) in large loops is still a difficult task. Data driven proposals could be adopted in the future to aleviate this problem.

Figure 4.4 shows a comparison between the number of effective particles $N_{eff} = 1/\sum_{i=1}^{N} w_i^2$ in Marginal-SLAM and FastSLAM[10]. Clearly, the marginal particle filter reaches a steady state, but FastSLAM quickly loses particle diversity.

[9] Due to rotation simetry and the robot moving on a plane, point feature locations can be represented only with two parameters in a plane

[10] Results are based on the following parameter settings: $N = 200$, $\sigma_d = 0.1m, \sigma_\alpha = 0.5deg, \sigma_\rho = 0.025m, \sigma_\phi = 3deg$.

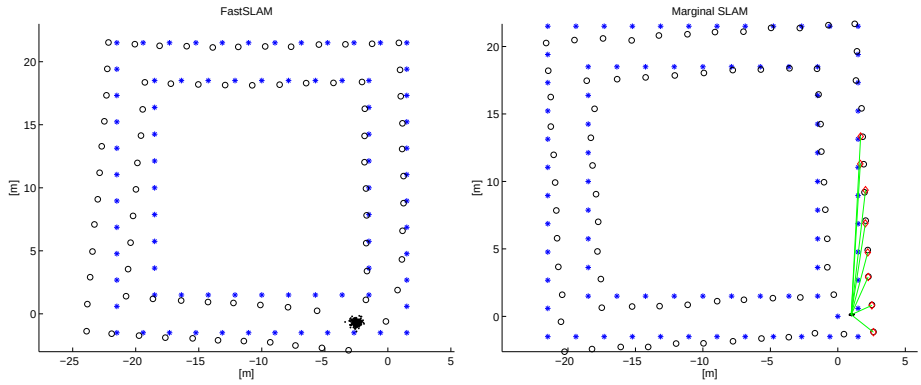


Figure 4.5: a) Particle degeneracy in FastSLAM prevents it from closing the loop. b) Marginal-SLAM is able to close the loop and converges to the true map after 10 laps. Although the map seems rotated, the relative location of the features and the robot location is almost perfect.

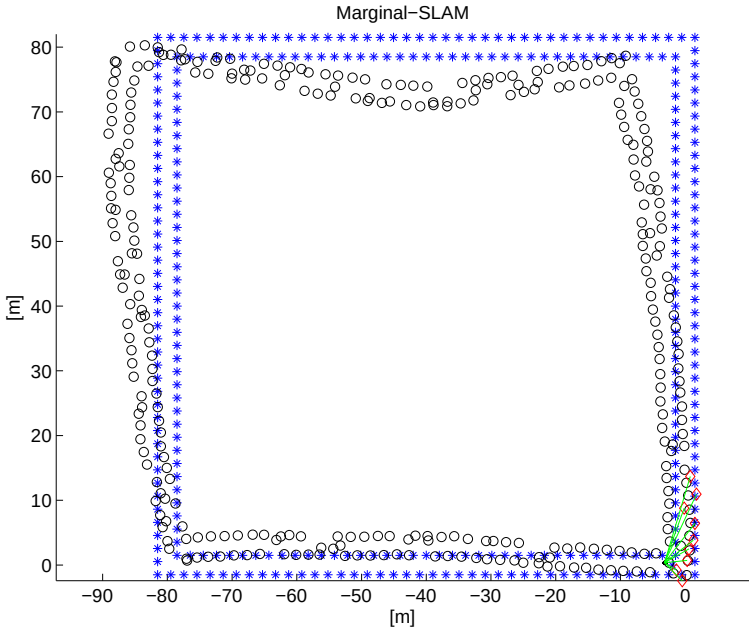


Figure 4.6: Marginal-SLAM after 10 laps in a bigger loop with high motion and observation noise $\sigma_\alpha = 10deg, \sigma_\phi = 10deg$. The loop is closed but the convergence is slow. Also the map becomes stuck in a bad optimum, due to the nonstationarity of the problem.

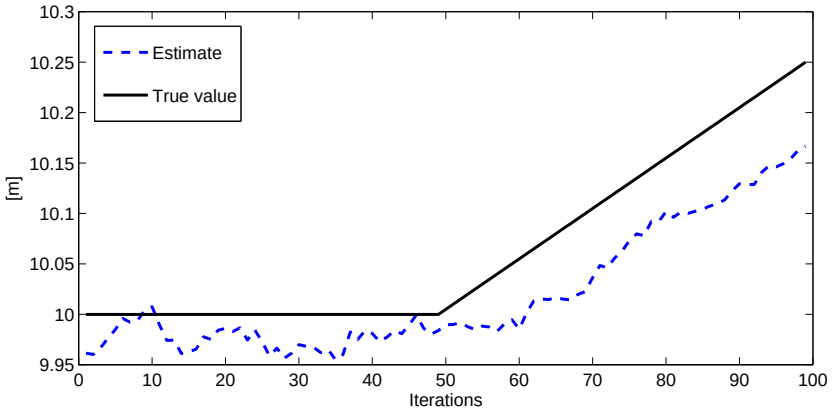


Figure 4.7: Tracking a moving map feature with Marginal-SLAM with $\sigma_d = 0.01m, \sigma_\alpha = 0.05deg, \sigma_\rho = 0.1m, \sigma_\phi = 0.5deg$. Black thick line: true location. Blue thin line: estimated location.

The final map using both approaches is shown in figure 4.5. In the Marginal-SLAM plot, the relative locations of the landmarks and the robot converge to the true solution. However, the global location is biased with respect to the ground truth. This is due to the observability assumptions in our SLAM model. The robot location can only be measured through the landmark location. The final map is valid up to an isometric transformation (translation plus rotation). This effect can be reduced if we fix a landmark location or use a robocentric representation. Figure 4.6 represents an even larger experiment and higger noise in the models. In this case, the map is biased. This may be an effect of the nonstationarity of SLAM due to the high odometric error.

Although our goal has been to develop a method for static maps using decreasing learning rates, it is possible to adopt small constant learning rates to track slowly changing map features. Mapping in real scenarios requires the ability to deal with pseudo-dynamic objects, like chairs and doors. Those elements are difficult to identify as dynamic, but their movements can lead to inconsistent maps. Figure 4.7 shows the evolution of a parameter estimate when the true landmark location changes. This ability to track moving features also implies that it would enable Marginal-SLAM to recover from wrong data association, biased map merging and loop closing, provided that the correct correspondences are obtained in subsequent steps. The selection of the learning ratio depends on the required accuracy, the convergency speed and the ability to track objects.

4.5 Discusion

Although our system is able to deal with general nonlinear non-Gaussian joint parameter and state estimation, the experimental results still introduce some open questions in terms of convergency. In this section, we introduce some intuitions about convergence of the stochastic approximation method. From the SLAM point of view, if we model the system using standard geometric definitions, then the optimal solution can not be achieved using filtering techniques, becase the robot motion is a nonstationary process. In nonstationary processes, we need to smooth the trajectory, that is, gather all the information until the end before starting to learn the parameters. Intuitively, we can see that map learning in a nonstationary process is like learning to solve murders: hindsight is always required to infer what happened at the murder scene, that is, the crime must happen *before* trying to find the murderer [Russell 02].

We also relate the SLAM problem to the adaptive control problem, which also needs to deal with nonstationarity issues.

4.5.1 Convergence of the Stochastic Approximation algorithm

Several proofs has been presented in the control literature that guarantee the convergent of the stochastic approximation parameter estimation to a stationary point under weak conditions. The more general is the Kushner and Clark lemma which assumes that the learning function $V_t(\theta_t, \mathbf{x}_t)$ can be split in two additive components: a deterministic error function and a stochastic perturbation. In this framework, the Kushner and Clark lemma guarantees convergence based on the ergodicity of the parameters sequence and the limit properties of the sequence of stochastic perturbations [Delyon 96].

This general result has been applied to the special case where the random variable $\mathbf{x}_t$ represents the state of a Markov system [Ljung 77, Metivier 84, Delyon 96]. Then, the Kushner and Clark conditions can be expressed in terms of the underlying Markov chain. In fact, the latent process $\{\mathbf{x}_k,\ 0 \leq k < \infty\}$ needs to be *somehow stationary*. Detailled explanation on the assumptions for the linear case can be found in [Ljung 77] which are generalized for arbitrary Markov chains in [Metivier 84, Delyon 96].

Apart from the standard contractive properties of Markov chains which has to be preserved for any possible parameter value, the convergence proof relies on the assumption that a unique invariant probability of the Markov chain exists. Using the standard formulation of SLAM, the Markov chain is nonstationary. For instance, the transition probability depends on the motion

command which is definitely nonstationary, except in the trivial case of the robot being stopped or in constant motion.

Assuming that the latent space is on a compact, metrizable topological space and being the transition function a continuous map $f : \mathbb{R}^n \longrightarrow \mathbb{R}^n$, then the Krylov-Bogolyubov theorem guarantees the existence of an invariant measurement[11] for f. In SLAM, the assumption is trivial since the robot is moving on an finite Euclidean space. However, measure theoretical definitions are in general difficult to obtain from the standard geometrical definitions. In fact, only few systems like Hamiltonian systems have natural invariant measures. For robotics, this is still an open question.

In our approach to maximum likelihood parameter estimation [LeGland 97], the Markov chain has to be extended to consider the joint state, observation and belief process $\mathbf{z}_t = (\mathbf{x}_t, \mathbf{y}_t, p_\theta(\mathbf{x}_t|\mathbf{y}_{1:t-1}), \nabla_\theta p_\theta(\mathbf{x}_t|\mathbf{y}_{1:t-1}))$. Therefore, invariant distributions should take into account the different dimensional spaces along time and space.

The main difference between nonlinear stochastic approximation and recursive linear estimation is because the latter computes the optimal estimate every step. Therefore, the algorithm is recursive but not iterative. For instance, the Kalman Filter has a probability in convergence in one single step. The computed set of parameters is an stationary point. Then, the Markov process does not need to be stationary, because the dynamic system is ergodic given a stationary set of parameters.

On the other hand, nonlinear stochastic approximation, even adaptive SA, is an iterative process. They have asymptotic convergence. In the recursive form, we compute a single iteration of the algorithm on every recursion. Thus, the computed parameter estimate is not an stationary point. Then, the underlying process needs to be stationary to guarantee ergodicity of the stochastic approximation process. The problem appears when both the process and the parameters are not stationary, because then, the system is not ergodic and iterations can not be delayed in time.

4.5.2 Subspace methods for SLAM

A interesting approach for joint state and parameter estimation is the branch of subspace methods for system identification [Larimore 83, Benveniste 07, Mercère 07]. These methods share the formulation of the state-space approach although, analogously to the standard stochastic approximation method, the

[11] The concept of probability is a special case of the concept of measure. Therefore, the existence of an invariant measurement implies the existence of an invariant distribution

state only represents the dynamic variables. In the case of linear dynamic systems, the problem is equivalent to the eigenstructure identification, that is, identifying the eigenvalues (the poles of the system) and the corresponding eigenvectors of the linear system. The main advantage is that, in that case, the system is fully defined and does not require an explicit parametric respresentation. It can be seen as a generalized model selection technique.

Subspace methods for multiple inputs and multiple outputs are, in fact, a dimensionality reduction technique, where the action and observation spaces are mapped in the latent space through the manifold of the dynamic model. In general, subspace approaches are batch algorithms in the sense that they learn the best eigenstructure provided all available data. However, some recursive approaches update the current estimates based on the new observations and inputs. The reader can find the analogy between the filtering and smoothing problem.

Statistical consistency of general subspaces methods resemble some assumptions from the stochastic approximation literature, although some proves can be seen purely geometrical, without any probabilistic assumption. This is the case of general batch subspace methods, which are consistent for non-stationary and even unstable processes [Benveniste 07]. The reason of the behavior is that batch estimations do not delay iterations in time, so they do not require the system to be ergodic. Again, recursive subspace methods does require an underlying stationary process [Mercère 07].

Another issue of nonlinear subspace methods is the general assumption that the inputs $\mathbf{u}$ are independent of the outputs $\mathbf{y}$ [Kawahara 06]. This is true for passive navigation methods, but in active SLAM, this proposition can not be fulfilled as explained in section 1.5.

4.6 Conclusions

The experiments and arguments indicate that Marginal-SLAM is an important new direction in the design of particle methods for SLAM. Algorithms designed to work on the marginal space appear to behave better than the ones designed to work on the path space.

In this preliminary work, Marginal-SLAM exhibits nice properties, such as being able to track slowly moving objects and potentially being able to recover from erroneous data-association. Marginal-SLAM does not suffer from some shortcomings of existing particle methods for SLAM. When mapping known areas, the algorithm reaches an accurate steady state without diverging. However, efficient convergence in large-scale SLAM domains is still an open

question. The presented method requires a considerable amount of information to converge to the solution, which is a strong assumption in SLAM. *What is missing is a fully Bayesian way of estimating the static map parameters, while integrating over the states recursively in time.*

In future work, we plan to test the algorithm more thoroughly in real domains and introduce known improvements like the N-body methods or the more efficient Hessian based maximum likelihood. We also plan to focus on solving the problem of designing efficient full-Bayesian recursive parameter estimators for nonlinear state spaces.

Chapter 5

Active Policy Learning

5.1 Introduction

In previous sections, we have seen that the source of inconsistency in KF and SMC based algorithms is related to the amount of uncertainty. In chapter 3 we proposed some techniques to reduce it pasively. That is, given a fixed amount of data, we reduce the total uncertainty involved in the approximation steps. However, we can do better with autonomous robots. They can decide where to go, therefore, they can decide what to see. An autonomous robot can reason about the uncertainty level and plan an execution policy to reduce it actively. The problem is how to choose that policy. It has to be information driven, thus, the amount of information becomes a reward function to the robot. Learning a policy in terms of a reward or cost function is called reinforcement learning.

The direct policy search method for reinforcement learning has led to significant achievements in control and robotics [Kohl 04, Lawrence 03, Ng 04, Peters 06]. The success of the method does often, however, hinge on our ability to formulate expressions for the gradient of the expected cost [Baxter 01, Peters 06, Singh 05], which is extremely useful to reduce the complexity of the problem in high dimensional problems. In some important applications in robotics, such as robot exploration with discontinuities in the measurement model (caused either by occlusions or the robot's limited field of view), the expected cost is discontinuous and hence one cannot compute gradients easily. In this chapter, we present a direct policy search method for continuous policy spaces that relies on active learning to side-step the need for gradients. Also, the new method enables us to attack problems with non-differentiable cost functions.

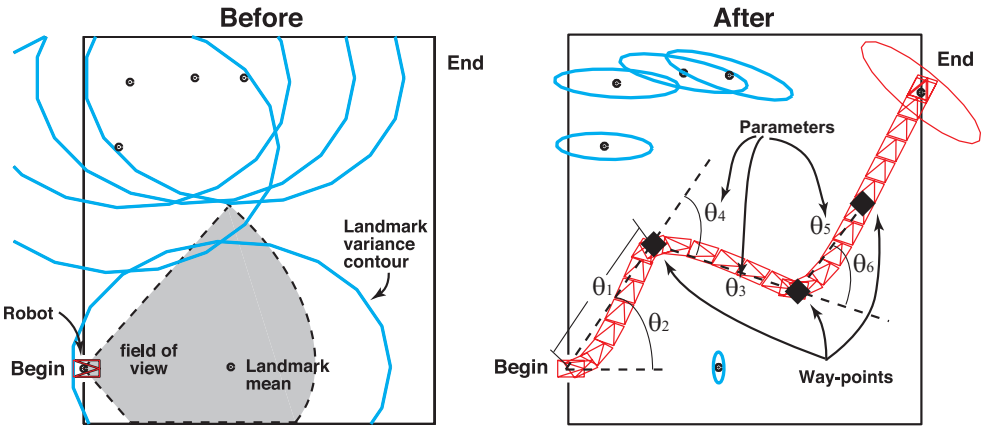


Figure 5.1: In this example, the map is represented by a set of point landmarks that can be observed with a sensor that provides range and bearing information from the robot point of view, like a sonar. The robot is able to navigate in a planar surface, having 3 degrees of freedom (pose and heading). The robot plans a path that allows it to accomplish the task of going from "Begin" to "End" while simultaneously reducing the uncertainty in the map and the robot pose estimates, represented by the ellipses. At the beginning, the robot has poor information over its pose and the landmark locations, and it can only see the landmarks within its field of view (left). In the planning stage, the robot must compute the optimal set of control parameters that define the next waypoints. At the end, the new observations are integrated in the recursive estimation process and the variance of the filtering distribution has decreased. Hence, posterior navigation inside the room will be easier and more accurate.

The proposed active policy learning approach also seems to be more appropriate in situations where the cost function has many local minima that cause the gradient methods to get stuck. Moreover, in situations where the cost function is very expensive to evaluate by simulation, an active learning approach that is designed to minimize the number of evaluations might be more suitable than gradient methods, which often require small step sizes for stable convergence (and hence many cost evaluations).

We demonstrate the new approach on a hard robotics problem: planning and exploration under uncertainty. This problem plays a key role in simultaneous localization and mapping (SLAM), see for example [Sim 05, Stachniss 05a]. Mobile robots must maximize the size of the explored terrain, but, at the same time, they must ensure that localization errors are minimized. While exploration is needed to find new features, the robot must return to places were

known landmarks are visible to maintain reasonable map and pose (robot location and heading) estimates.

In our setting, the robot is assumed to have a rough *a priori* estimate of the map features and its own pose. The robot must accomplish a series of tasks while simultaneously maximizing its information about the map and pose. For example, let us assume the situation presented in figure 5.1, where a robot has to move from "Begin" to "End" by planning a path that satisfies logistic and physical constraints. The planned path must also result in improved map and pose estimates. *As soon as the robot accomplishes a task, it has a new a posteriori map that enables it to carry out future tasks in the same environment more efficiently.* This sequential decision making problem is exceptionally difficult because the actions and states are continuous and high-dimensional. Moreover, the cost function is not differentiable and depends on the posterior belief (filtering distribution). Even a toy problem, like this one with only five landmarks, requires enormous computational effort. As a result, it is not surprising that most existing approaches relax the constraints. For instance, full observability is assumed in [Paris 02, Sim 05], known robot location is assumed in [Leung 05], a small set of actions and myopic planning is adopted in [Stachniss 05a, Vidal-Calleja 06, Bryson 08], and discretization of the state and/or actions spaces appears in [Hernandez 04a, Kollar 06, Sim 05]. *The method proposed in this work does not rely on any of these assumptions.*

Our direct policy solution uses an any-time probabilistic active learning algorithm to predict what policies are likely to result in higher expected returns. The method effectively balances the goals of exploration and exploitation in policy search. It is motivated by work on experimental design [Sacks 89, Jones 98, Santner 03, Siah 04]. Simpler variations of our ideas appeared early in the reinforcement learning literature. In [Kaelbling 90], the problem is treated in the framework of exploration/exploitation with bandits. An extension to continuous spaces (infinite number of bandits) using locally weighted regression was proposed in [Moore 96]. Our work presents richer criteria for active learning as well suitable optimization objectives.

We also present Posterior Cramér-Rao Bounds (PCRB) to approximate the cost function in robot exploration. These bounds are easy to compute and are not susceptible to errors introduced by suboptimal filtering techniques for SLAM. The experiments show that, in this domain, the bounds seem to be tight and hence they allow for the development of efficient algorithms.

Although the discussion is focused on robot exploration and planning, our policy search framework extends naturally to other domains. Related problems appear the fields of terrain-aided navigation [Bergman 99, Paris 02]

and dynamic sensor networks [Hernandez 04b, Singh 05].

5.2 Robot Exploration and Planning

Although the algorithm proposed in this work applies to many sequential decision making settings, we will restrict our attention to the robot exploration and planning domain. In this domain, the robot has to plan a path that will improve its knowledge of its pose (location and heading) and the location of navigation landmarks. In doing so, the robot might be subject to other constraints such as low energy consumption, limited time, safety measures and obstacle avoidance. However, for the time being, let us first focus on the problem of minimizing posterior errors in localization and mapping as this problem already captures a high degree of complexity.

There are many variations of this problem, but let us get back to the situation shown in figure 5.1 for illustration purposes. Here, the robot has to navigate from "Begin" to "End" while improving its estimates of the map and pose. For the time being, let us assume that the robot has no problem in reaching the target. Instead, let us focus on how the robot should plan its path so as to improve its map and pose posterior estimates. Initially, as illustrated by the ellipses on the left plot, the robot has vague priors about its pose and the location of landmarks. We want the robot to plan a path (parameterized policy $\pi(\theta)$) so that by the time it reaches the target, it has learned the most about its pose and the map. This way, *if the robot has to repeat the task, it will have a better estimate of the map and hence it will be able to accomplish the task more efficiently.*

The policy is simply a path parameterized as a set of ordered way-points $\theta^{(i)} = [\theta_d^{(i)}, \theta_\alpha^{(i)}]$, which represents the heading and distance to the next way-point, although different representations can be used depending on the robot capabilities[1]. A trajectory with 3 way-points, whose location was obtained using our algorithm, is shown on the right plot of figure 5.1. We use a standard model based motion controller to generate the motion commands $\mathbf{u} = \{\mathbf{u}_{1:\mathcal{T}}\}$ to follow the path for $\mathcal{T}$ steps. The controller moves the robot towards each way-point in turn while taking into account the kinematic and dynamic constrains of the problem. It is imperative to notice that the robot has a limited

[1]In this formulation the unknown parameters θ are the policy parameters. The map elements, no matter if they are static or dynamic should be included in augmented state vector $\mathbf{x}$. We are not using Marginal-SLAM because we need the belief of every landmark and the robot location.

field of view. It can only see the landmarks that "appear" within an *observation gate*.

Having restricted the problem to one of improving the posterior pose and the map estimates, a natural cost function is the average mean square error (AMSE) of the state:

$$C^{\pi}_{AMSE} = \mathbb{E}_{p(\mathbf{x}_{0:\mathcal{T}}, \mathbf{y}_{1:\mathcal{T}}|\boldsymbol{\pi})} \left[\sum_{t=1}^{\mathcal{T}} \lambda^{\mathcal{T}-t} (\widehat{\mathbf{x}}_t - \mathbf{x}_t)(\widehat{\mathbf{x}}_t - \mathbf{x}_t)^T \right], \tag{5.1}$$

where $\widehat{\mathbf{x}}_t = \mathbb{E}_{p(\mathbf{x}_t|\mathbf{y}_{1:t}, \boldsymbol{\pi})}[\mathbf{x}_t]$. The expectation is with respect to $p(\mathbf{x}_{0:\mathcal{T}}, \mathbf{y}_{1:\mathcal{T}}|\boldsymbol{\pi}) = p(\mathbf{x}_0) \prod_{t=1}^{\mathcal{T}} p(\mathbf{x}_t|\mathbf{u}_t, \mathbf{x}_{t-1}) p(\mathbf{y}_t|\mathbf{x}_t, \mathbf{u}_t)$, $\lambda \in [0, 1]$ is a discount factor, $\pi(\theta)$ denotes the policy parameterized by the way-points $\theta_i \in \mathbb{R}^{n_\theta}$, $\mathbf{x}_t \in \mathbb{R}^{n_x}$ is the hidden state (robot pose and location of map features) at time t, $\mathbf{y}_{1:\mathcal{T}} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{\mathcal{T}}\} \in \mathbb{R}^{n_y \mathcal{T}}$ is the history of observations along the planned trajectory for $\mathcal{T}$ steps, $\mathbf{a}_{1:\mathcal{T}} \in \mathbb{R}^{n_a \mathcal{T}}$ is the history of actions determined by the policy $\pi(\theta)$ and $\widehat{\mathbf{x}}_t$ is the posterior estimate of the state at time t.

In our application to robotics, we focus on the uncertainty of the posterior estimates at the end of the planning horizon. That is, we set λ so that the cost function reduces to:

$$C^{\pi}_{AMSE} = \mathbb{E}_{p(\mathbf{x}_{\mathcal{T}}, \mathbf{y}_{1:\mathcal{T}}|\boldsymbol{\pi})} \left[(\widehat{\mathbf{x}}_{\mathcal{T}} - \mathbf{x}_{\mathcal{T}})(\widehat{\mathbf{x}}_{\mathcal{T}} - \mathbf{x}_{\mathcal{T}})^T \right] \tag{5.2}$$

Note that the true state $\mathbf{x}_{\mathcal{T}}$ and observations are unknown in advance and so one has to marginalize over them.

The cost function hides an enormous degree of complexity. It is a matrix function of an *intractable filtering distribution* $p(\mathbf{x}_{\mathcal{T}}|\mathbf{y}_{1:\mathcal{T}}, \boldsymbol{\pi})$ (also known as the belief or information state). As we have seen along this thesis, the computation of this belief is the filtering process in SLAM, which is a challenging problem and even suboptimal methods require a high computation cost. Moreover, in our domain, the robot only sees the landmarks within and observation gate, and the motion commands are limited by nonholomic constraints which make it not derivable in certain points. *Consequently, the observation and motion models are discontinuous and hence one cannot compute derivatives of the cost function.*

The matrix function has to be mapped to a scalar value function. Several criteria have been proposed in the experimental design and information theory literature. For example, see Chaloner et al. [Chaloner 95] for a introductory review. For the problem of robot navigation, the different criteria were studied in detail in [Sim 05], where a very instructive comparison between A-optimality and D-optimality is presented. Although they assumed

total observability of the map, the discussion and conclusions are still valid for partial observability.

Since the models are not linear-Gaussian, one cannot use standard linear-quadratic-Gaussian (LQG) controllers [Bertsekas 95] to solve our problem. Moreover, since the action and state spaces are large-dimensional and continuous, one cannot discretize the problem and use closed-loop control as suggested in [Tremois 99]. That is, the discretized partially observed Markov decision process is too large for stochastic dynamic programming [Smallwood 73]. It is important to note that the cost function depends on the belief distribution, therefore, it does not admit close form recursion, and it cannot be evaluated instantly for any state and/or action. As a consequence, dynamic programming cannot be applied *at all*.

As a result of these considerations, we adopt the direct policy search method [Williams 92, Ng 00]. In particular, the initial policy is set either randomly[2] or using prior knowledge. Given this policy, we conduct simulations to estimate the AMSE. These simulations involve sampling states and observations using the prior, dynamic and observation models. They also involve estimating the posterior mean of the state with a suboptimal filter. After evaluating the AMSE using the simulated trajectories, we updtate the policy parameters and iterate with the goal of minimizing the AMSE. Note that in order to reduce Monte Carlo variance, the random seed should be frozen before each simulation as described in [Ng 00]. The pseudocode for this open-loop simulation-based controller (OLC) is shown in figure 5.2.

Note that as the robot moves along the planned path, it is possible to use the newly gathered observations to update the posterior distribution of the state. This distribution can then be used as the prior for subsequent simulations. This process of replanning is known as open-loop feedback control (OLFC) [Bertsekas 95]. We can also allow for the planning horizon to recede. That is, as the robot moves, it keeps planning $\mathcal{T}$ steps ahead of its current position. This control framework is also known as receding-horizon model-predictive control [Maciejowski 02]. In the experiments, we always use receding-horizon whenever is possible. Therefore, we use the terms *open-loop feedback control* and *model-predictive control* interchangeably.

In the following two subsections, we will describe a way of conducting the simulations to estimate the AMSE and, subsequently, proceed to describe the observation and odometry models in detail. The active policy update algorithm will be described in Section 5.3.

[2]In practice, we use latin hypercube sampling, which provides optimal sampling of the parameter space with uninformative prior.

1. Choose an initial policy π_0.
2. For $j = 1 : MaxNumberOfPolicySearchIterations$
   (a) For $i = 1 : N$
      i. Sample the prior states $\mathbf{x}_0^{(i)} \sim p(\mathbf{x}_0)$.
      ii. For $t = 1 : \mathcal{T}$
         A. Use a motion controller regulated about the path π_j to determine the current action $\mathbf{u}_t^{(i)}$.
         B. Sample the state $\mathbf{x}_t^{(i)} \sim p(\mathbf{x}_t|\mathbf{u}_t^{(i)}, \mathbf{x}_{t-1}^{(i)})$.
         C. Generate observations $\mathbf{y}_t^{(i)} \sim p(\mathbf{y}_t|\mathbf{u}_t^{(i)}, \mathbf{x}_t^{(i)})$ as described in section 5.2.1. There can be missing observations.
         D. Compute the filtering distribution $p(\mathbf{x}_t|\mathbf{y}_{1:t}^{(i)}, \mathbf{u}_{1:t}^{(i)})$ using a SLAM filter.
   (b) Evaluate the approximate AMSE cost function of equation ((5.3)) using the simulated trajectories.
   (c) Use the active learning algorithm with Gaussian processes, described in section 5.3, to generate the new policy π_{j+1}. The choice of the new policy is governed by our desire to exploit and our need to explore the space of policies (navigation paths). In particular, we give preference to policies for which we expect the cost to be minimized and to policies where we have high uncertainty about what the cost might be.

Figure 5.2: The overall solution approach in the open-loop control (OLC) setting. Here, N denotes the number of Monte Carlo samples and $\mathcal{T}$ is the planning horizon. In replanning with open-loop feedback control (OLFC), one simply uses the present position and the estimated posterior distribution (instead of the prior) as the starting point for the simulations. One can apply this strategy with either approaching or receding horizon control. It is implicit in the pseudo-code that we use common random numbers so as to reduce variance.

5.2.1 Simulation of the cost function

We can approximate the AMSE cost by simulating N state and observation trajectories $\{\mathbf{x}_{1:\mathcal{T}}^{(i)}, \mathbf{y}_{1:\mathcal{T}}^{(i)}\}_{i=1}^N$ and adopting the Monte Carlo estimator:

$$C_{AMSE}^{\pi} \approx \frac{1}{N} \sum_{i=1}^{N} (\widehat{\mathbf{x}}_{\mathcal{T}}^{(i)} - \mathbf{x}_{\mathcal{T}}^{(i)})(\widehat{\mathbf{x}}_{\mathcal{T}}^{(i)} - \mathbf{x}_{\mathcal{T}}^{(i)})^T \tag{5.3}$$

Assuming that π is given (we discuss the active learning algorithm to learn π in Section 5.3), one uses a motion controller to obtain the next action $\mathbf{u}_t$.

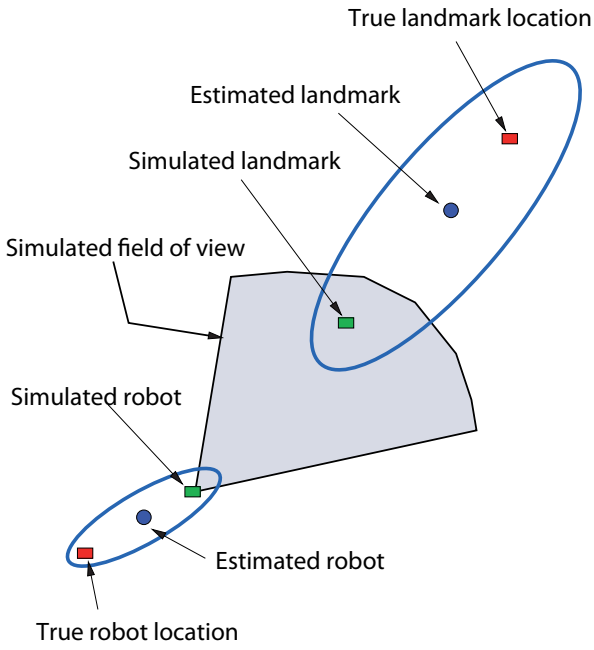


Figure 5.3: An observation is generated using the current map and robot pose estimates. Gating information is used to validate the observation. In this picture, the simulation validates the observation despite the fact that the true robot and feature locations (black boxes) are too distant for the given field of view. New information is essential to reduce the uncertainty and improve the simulations.

Once the action is set, the new state $\mathbf{x}_t$ is easily simulated using the odometry model. The process of generating observations is more involved. As shown in figure 5.3, for each landmark, one draws a sample from its posterior. If the sample falls within the observation gate, it is treated as an observation. As in most realistic settings, most landmarks will remain unobserved.

After the trajectories $\{\mathbf{x}_{1:\mathcal{T}}^{(i)}, \mathbf{y}_{1:\mathcal{T}}^{(i)}\}_{i=1}^{N}$ are obtained, one uses any SLAM filter (EKF, UKF or particle filter) to compute the posterior mean state $\widehat{\mathbf{x}}_{1:\mathcal{T}}^{(i)}$. *The evaluation of this cost function is therefore extremely expensive.* Moreover, since the model is nonlinear, it is hard to quantify the uncertainty introduced by the suboptimal filter. Later, in section 5.4, we will discuss an alternative cost function, which consists of a lower bound on the AMSE, thus being independent on the filter used. Yet, in both cases, it is imperative to minimize the number of evaluations of the cost functions. This calls for an active learning

approach.

5.3 Active Policy Learning

This section presents an active learning algorithm to update the policy parameters after each simulation. In particular, we adopt the expected cost simulation strategy presented in [Ng 00]. In this approach, a scenario consists of an initial choice of the state and a sequence of random numbers. Given a policy parameter vector and a set of fixed scenarios, the simulation is deterministic and yields an empirical estimate of the expected cost [Ng 00].

The simulations are typically very expensive and consequently cannot be undertaken for many values of the policy parameters. Discretization of the potentially high-dimensional and continuous policy space is out of the question[3]. The standard solution to this problem is to optimize the policy using gradients. However, the local nature of gradient-based optimization often leads to the common criticism that direct policy search methods "get stuck" in local minima. Even more pertinent to our setting, is the fact that the cost function is discontinuous and hence policy gradient methods do not apply. We present an alternative approach to gradient-based optimization for continuous policy spaces. This approach, which we refer to as active policy learning, is based on experimental design ideas based on response surfaces [Kushner 64, Jones 98, Jones 01, Sasena 02, Santner 03]. Active policy learning is an any-time, "black-box" statistical optimization approach. The special nature of policy learning in information systems makes it perfectly suitable for response surfaces optimization strategies. Figure 5.4 illustrates it for a simple one-dimensional example. The approach is iterative and involves three steps.

In the first step, a Bayesian regression model is learned to map the policy parameters to the estimates of the expected cost function obtained from previous simulations. In this work, the regression function is obtained using Gaussian processes (GPs), originally known as kriging [Krige 51, Rasmussen 06]. Though in figure 5.4 the GPs provide a good approximation to the expected cost, it should be emphasized that the objective is not to provide a good estimate of the regression funtion or to reduce the mean squared error over the entire feasible domain [Santner 03, Sacks 89]. Instead, we aim to predict the expected cost well near the minima [Kushner 64, Jones 98]. The details of the GP fit are presented in Section 5.3.1.

[3]For example, in our setup, a simple discretization of 100 cells per parameter, would represents 10^{12} possible policies. Even bootstrap or iterative methods would be intractable in real-time.

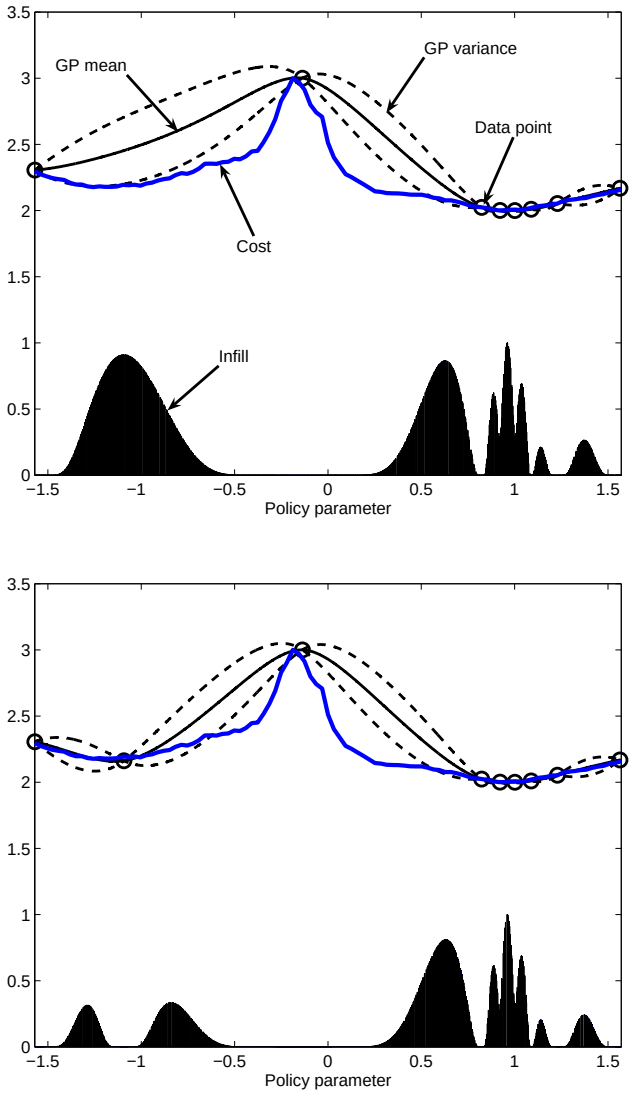


Figure 5.4: An example of active policy learning with a univariate policy using data generated by our simulator. The figure on top shows a GP approximation of the cost function using 11 simulated values. In reality, the true expected cost function is unknown. The figure also shows the expected improvement (infill) of each potential next sampling location in the lower shaded plot. The infill is high where the GP predicts a low expected cost (exploitation) and where the prediction uncertainty is high (exploration). Selecting and labelling the point suggested by the highest infill in the top plot produces the GP fit in the plot shown below. The new infill function, in the plot below, suggests that we should query a point where the cost is expected to be low (exploitation).

The second step involves active learning. Because the simulations are expensive, we must ensure that the selected samples (policy parameter candidates) will generate the maximum possible improvement in terms of utility/cost. Roughly speaking, it is reasonable to evaluate the policy parameters where the GP predicts a low expected cost (exploitation of a good policy) or where the GP variance is large (exploration of the parameter space). From the optimization point of view, the selected points must find the best local minimum while searching for a better global minimum. These intuitions can be incorporated in the design of a statistical measure indicating where to sample the cost function. This measure is known as the infill function, borrowing the term from the geostatistics literature. Figure 5.4 depicts a simple infill function that captures our intuitions. More details on how to choose the infill are presented in section 5.3.2.

Having defined an infill function, still leaves us with the problem of optimizing it. This is the third and final step in the approach. Our thesis is that the infill optimization problem is more amenable than the original problem because in this case the cost function is known and easy to evaluate. Furthermore, for the purposes of our application, it is not necessary to guarantee that we find the global minimum, merely that we can quickly locate a point that is likely to be as good as possible. On the other hand, in section 5.3.2 we present some examples of the infill function which are highly multimodal and could be almost *flat* in many areas.

To deal with this nonlinear constrained optimization problem, we adopted the DIvided RECTangles (DIRECT) algorithm [Jones 93, Gablonsky 01]. DIRECT is a deterministic, derivative-free global optimization algorithm. It uses the existing samples of the objective function to decide how to proceed to divide the feasible space into finer rectangles. For low-dimensional parameter spaces, say up to 10D, DIRECT provides a better solution than gradient approaches because the infill function tends to have many local optima and flat regions. Another motivating factor is that DIRECT's implementation is easily available [Finkel 03, Gablonsky 01]. However, we conjecture that for large dimensional spaces, sequential quadratic programming or concave-convex programming [Smola 05] might be better algorithm choices for infill optimization.

5.3.1 Gaussian processes

A *Gaussian process* (GP), $\mathbf{z}(\cdot) \sim GP(m(\cdot), k(\cdot,\cdot))$, is an infinite random process indexed by the vector θ, such that any realization $\mathbf{z}(\theta)$ is Gaussian [Rasmussen 06]. From a regression point of view, a Gaussian process or krig-

ing is a way of *modeling the function as a realzation of a stochastic process.* Thus, as shown in figure 5.5, it can be seen as a distribution over functions:

$$m(\theta) = \mathbb{E}[\mathbf{z}(\theta)] \tag{5.4}$$
$$k(\theta, \theta') = \mathbb{E}[(\mathbf{z}(\theta) - m(\theta))(\mathbf{z}(\theta') - m(\theta'))] \tag{5.5}$$

A GP is also a Bayesian kernel method that exploits the kernel trick (appendix B) for nonlinear regression or classification. The idea behind the kernel trick is to map the nonlinear problem in a higher dimensional space $\phi(\mathbf{x})$ where it becomes linear $f(\mathbf{x}) = \phi(\mathbf{x})^T\mathbf{w}$.

Thus, if we represent the real process with our linear regression model $\mathbf{z}(\theta) = \phi(\theta)^T\mathbf{w}$ with prior $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \Sigma)$, then, the posterior mean and covariance functions are:

$$m(\theta) = \phi(\theta)^T\mathbb{E}[\mathbf{w}] = 0 \tag{5.6}$$
$$k(\theta, \theta') = \phi(\theta)^T\mathbb{E}[\mathbf{w}\mathbf{w}^T]\phi(\theta) = \phi(\mathbf{x})^T\Sigma\phi(\mathbf{x}') \tag{5.7}$$

which is exactly the same definition that we get for the kernel function. That is, the kernel function of a Gaussian process is the covariance function of the underlying Bayesian linear regression model. It is important to note that we are not doing smoothing in the parameter space, instead we translate the parameter space to a new feature space where the regression becomes a simple linear problem (appendix B). Using the kernel as a correlation function between the points of the Gaussian process, we can predict new realizations of the process. In that way, it is a non-parametric form of regression.

For a general model, where the mean and covariance functions are unknown and need to be explicitly computed, we can parameterize the GP hierarchically. Then we can split the effect of the bias and the uncertainty. Also, it is easier to work with a zero mean GP. Then, we can represent the regression function as:

$$C^\pi(\theta) = \mathbf{1}\mu + \mathbf{z}(\theta)$$
$$\mathbf{z}(\cdot) \sim GP(0, \sigma^2 K(\cdot, \cdot))$$

and subsequently estimate the posterior distributions of the mean μ and scale σ^2 using standard Bayesian conjugate analysis, see for example [Santner 03, Jones 98]. The symbol $\mathbf{1}$ denotes a column vector of ones. Assuming that n simulations have been conducted, the simulated costs $\{\mathbf{C}^\pi_{1:n}\}$ and the predicted

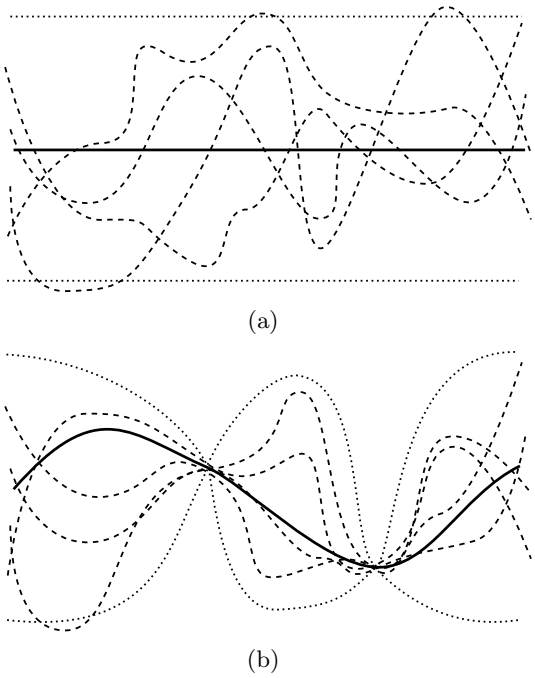


Figure 5.5: Gaussian Process can be seen as a distribution over function. The top plot represents the mean (solid), covariance (95% confidence intervals, dotted) an some sample functions (dashed). When new information is added, the distribution is updated. Intuitively, in the points where we have observations the uncertainty collapses. It is also propagated to the neighborhood based on the correlation function (bottom plot).

cost C^{π}_{n+1} for a new test point θ_{n+1} are jointly Gaussian:

$$\begin{bmatrix} C^{\pi}_{n+1} \\ C^{\pi}_{1:n} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 1 \\ \mathbf{1} \end{bmatrix} \mu, \sigma^2 \begin{bmatrix} k & \mathbf{k}^T \\ \mathbf{k} & \mathbf{K} \end{bmatrix} \right),$$

where $\mathbf{k}^T = [k(\theta_{n+1}, \theta_1) \cdots k(\theta_{n+1}, \theta_n)]$, $k = k(\theta_{n+1}, \theta_{n+1})$ and $\mathbf{K}$ is the training data kernel matrix with entries $k(\theta_i, \theta_j)$ for $i = 1, \ldots, n$ and $j = 1, \ldots, n$. Since we are interested in regression but also finding the minimum of the function, with points getting closer next to the minimum value, the Matérn kernel is a suitable choice for $k(\cdot|\cdot)$ [Santner 03, Stein 99, Sasena 02].

In the preliminar work of Jones et al. [Jones 98], the parameters are computed based on the maximum likelihood estimates. They assume noninfo-

mative priors. That approach is more general but also requires more data. Instead, we assign a normal-inverse-Gamma conjugate prior to the parameters: $\mu \sim \mathcal{N}(0, \sigma^2\delta^2)$ and $\sigma^2 \sim \mathcal{IG}(a/2, b/2)$. The priors play an essential role at the beginning of the design process, when there are only few data. The use of informative priors results in a speed up of the general algorithm, because the number of data points needed is reduced. Classical Bayesian analysis allow us to obtain analytical expressions for the posterior modes of these quantities:

$$\widehat{\mu} = (\mathbf{1}^T\mathbf{K}^{-1}\mathbf{1} + \delta^{-2})^{-1}\mathbf{1}^T\mathbf{K}^{-1}C^{\pi} \tag{5.8}$$

$$\widehat{\sigma}^2 = \frac{b + C^{\pi T}\mathbf{K}^{-1}C^{\pi} - (\mathbf{1}^T\mathbf{K}^{-1}\mathbf{1} + \delta^{-2})\widehat{\mu}^2}{n + a + 2} \tag{5.9}$$

Using the previous estimates, the GP predictive mean and variance are given by

$$\widehat{C}^{\pi}(\theta) = \widehat{\mu} + \mathbf{k}^T\mathbf{K}^{-1}(C^{\pi}_{1:n} - \mathbf{1}\widehat{\mu}) \tag{5.10}$$

$$\widehat{s}^2(\theta) = \widehat{\sigma}^2\left\{k - \mathbf{k}^T\mathbf{K}^{-1}\mathbf{k} + \frac{(1 - \mathbf{1}^T\mathbf{K}^{-1}\mathbf{k})^2}{(\mathbf{1}^T\mathbf{K}^{-1}\mathbf{1} + \delta^{-2})}\right\} \tag{5.11}$$

It is important to note that the GP predictive mean and variance is computed based on the assumption that $\widehat{\sigma}^2$ and all the parametes of the kernel funcion, like h are known (see appendix B). In practice, we have only estimates of those parameters, but we take those estimates as the true values. This mathematical *trick* only results in a small underestimation of the prediction error in small samples, although this effect can be compensated with a suitable prior.

Also, we found that learning all the GP parameters and the kernel hyperparameters in our setup may result in an overfitting of the function, because we want to find the minimum with few data points. Thus, we do not have enought information to learn all those parameters. In practice, we found that the properties of the cost function in different experiments are similar. Thus, the Matérn kernel hyperparameters are almost invariant, so we decided to manually fix it. It can be seen as a kernel calibration process.

A detailled derivation of the equations for the GP prediction can be found in [Santner 03].

5.3.2 Infill Function

Once the GP function is fitted, we need to find a criterion for the next query. The function that characterized that criterion is the *infill function*.

Let $C^{\pi}_{\min}$ denote the current lowest (best) estimate of the cost function. The simplest approach is to select that point as the next query. Due to the

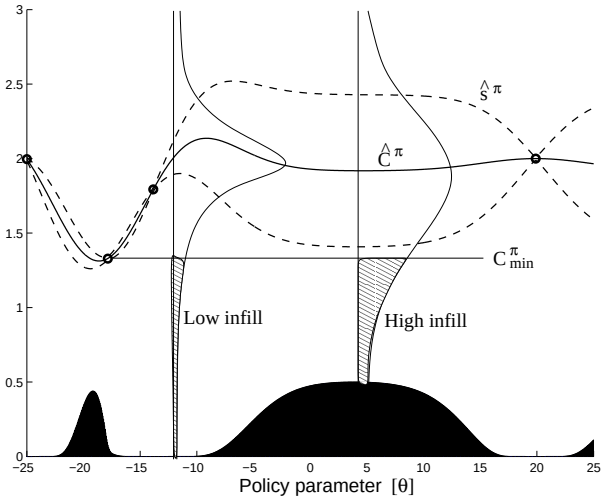


Figure 5.6: Probability of improvement (infill) function.

locality of interpolation techniques like Gaussian Processes, this method is easily trapped in local minima. Or it can be even worse and not even a local minimum is found, although the local minima can be forced using gradient estimation [Jones 01].

In order to achieve convergence of the global optimum, the sequence of queries, i.e. the cost function evaluations, must be dense. Intuitively, any globally convergent algorithm must explore regions of the parameter space that have been relatively unexplored and sample it. In that sense, Gaussian Processes have a statistical interpretation that can be exploited. They provide the regression function, but also the standard deviation. The later is a good measure of the information that we have on that region, that is, how much attention has been paid to that part of the space. Thus, we combine exploration and refinement of the parameters based on the standard deviation of the GP.

For example, as shown in figure 5.6, we can define the probability of improvement for any value $T \leq C^{\pi}_{\min}$ at a point θ to be

$$p(C^{\pi}(\theta) \leq T) = \Phi\left(\frac{T - \widehat{C}^{\pi}(\theta)}{\widehat{s}(\theta)}\right),$$

where $C^{\pi}(\theta) \sim \mathcal{N}(\widehat{C}^{\pi}(\theta), \widehat{s}(\theta)^2)$ and Φ denotes the cummulative density function (cdf) of the standard Normal distribution. This measure was proposed several decades ago by [Kushner 64], who used univariate Wiener process. Un-

der mild assumptions, these queries are dense, achieving global convergence [Gutmann 01].

However, as argued by [Jones 98], the probability of improvement is sensitive to the value of T. For example, if $T = C^{\pi}_{\min}$, the highest probability will ocurr exactly at $C^{\pi}_{\min}$. Again, no exploration is made and it will suffer from local minima. Lower values of T produce more exploratory behaviours, but the exploration/exploitation ratio depends also on the regression function, and it can be deceptive. To overcome this problem, Jones [Jones 01] suggested to use different values of T running in paralel. Although the performance of this technique is very promising, it incurs in extra query points, which is against the phylosophy of the whole optimization algorithm. As an alternative to the probability of improvement, [Mockus 78] defined the improvement over the current best point as $I(\theta) = \max\{0, C^{\pi}_{\min} - C^{\pi}(\theta)\}$. The expectation of this improvement for the GP can be computed integrating by parts

$$\mathbb{E}(I(\theta)) = \int_{I=0}^{I=\infty} I(\theta) \left\{ \frac{1}{\sqrt{2\pi}\widehat{s}(\theta)} \exp\left[-\frac{(C^{\pi}_{\min} - I(\theta) - \widehat{C}^{\pi}(\theta))^2}{2\widehat{s}^2(\theta)} \right] \right\} dI(\theta) \tag{5.12}$$

This resulted in the following infill function, the expected improvement (EI) [Mockus 78]

$$EI(\theta) = \begin{cases} (C^{\pi}_{\min} - \widehat{C}^{\pi}(\theta))\Phi(d) + \widehat{s}(\theta)\phi(d) & \text{if } \widehat{s} > 0 \\ 0 & \text{if } \widehat{s} = 0 \end{cases}$$

where ϕ is the pdf of the standard Normal distribution and $d = \frac{C^{\pi}_{\min} - \widehat{C}^{\pi}(\theta)}{\widehat{s}(\theta)}$.

A further generalization of the infill function, proposed by [Schonlau 98], is obtained by adding a non-negative integer parameter g, such that $I(\theta) = \max\{0, (C^{\pi}_{min} - C^{\pi}(\theta))^g\}$. This results in a generalized expected improvement

$$EI^g(\theta) = \widehat{s}^g(\theta) \sum_{j=0}^{g} (-1)^j \left(\frac{g!}{j!(g-j)!} \right) d^{g-j} T_j,$$

where $T_j = -\phi(d)d^{j-1} + (j-1)T_{j-2}$, starting with $T_0 = \Phi(d)$ and $T_1 = -\phi(d)$. The g parameter controls the trade-off between global search and local optimization (exploration/exploitation). When $g = 0$, the infill function is equivalent to the probability of improvement with $T = C^{\pi}_{\min}$. Then, emphasis is placed on trying to improve near the current best estimate $C^{\pi}_{\min}$, unless the observations strongly suggest improvement in areas of high variance. For $g = 1$, we have again the standard expected improvement function. The case

$g = 2$, i.e. $\mathbb{E}(I^2) = \mathbb{E}(I)^2 + Var(I)$, is interesting as it considers the standard expected improvement, but also the variance over the improvement; that is, the uncertainty over the improvement is explictly included. As g increases, areas of high model uncertainty are favoured. While there is no obvious way to select this parameter for an unknown cost function, the annealing strategy suggested by [Sasena 02] allows global search at the beggining to smoothly collapse to local improvement.

Convergence of the expected improvement algorithm is still an open question. Jones et al. [Jones 98] conjetured that the sequence of queries is dense. This has been proved for a similar one dimensional algorithm [Locatelli 97] or when the cost function belongs to the reproducing kernel Hilbert space generated by the kernel function of the GP [Vazquez 08].

5.4 A Cheaper Cost: The Posterior Cramér-Rao Bound

As we have seen, the cost function is directly related to the posterior distribution of poses and maps. Thus, there is no close form of the cost function. In section 5.2.1, we proposed a simulation approach that require to run an SLAM filter for each simulated scenario. This approximate filtering step is not only expensive, but also a possible source of errors when approximating the AMSE with Monte Carlo simulations.

The posterior Cramér-Rao bound (PCRB) for nonlinear systems leads to an alternative objective function that is cheaper to evaluate and *does not require that we run a SLAM filter*. That is, the criterion presented next does not require the adoption of an EKF, UKF, particle filter or any other suboptimal filter in order to evaluate it. The PCRB is a "measure" of the maximum information that can be extracted from the dynamic system when both the measurements and states are assumed random. It is defined as the inverse of the Fisher information matrix $\mathbf{J}$ and provides the following lower bound on the AMSE:

$$C^{\pi}_{AMSE} \geq C^{\pi}_{PCRB} = \mathbf{J}^{-1}$$

Several recursive algorithms have appeared in the literature to update the PCRB in filtering applications where the observations arrive sequentially.

5.4.1 PCRB for nonlinear models

Tichavský et al. [Tichavský 98], derived the following Riccati-like recursion to compute the Fisher information matrix for nonlinear (NL) filtering problems:

$$\mathbf{J}_{t+1} = \mathbf{D}_t - \mathbf{C}_t^T(\mathbf{J}_t + \mathbf{B}_t)^{-1}\mathbf{C}_t + \mathbf{A}_{t+1}, \tag{5.13}$$

where

$$\begin{aligned} \mathbf{A}_{t+1} &= \mathbb{E}[-\Delta_{x_{t+1}}^{x_{t+1}} \log p(\mathbf{y}_{t+1}|\mathbf{x}_{t+1})] \\ \mathbf{B}_t &= \mathbb{E}[-\Delta_{x_t}^{x_t} \log p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)] \\ \mathbf{C}_t &= \mathbb{E}[-\Delta_{x_{t+1}}^{x_t} \log p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)] \\ \mathbf{D}_t &= \mathbb{E}[-\Delta_{x_{t+1}}^{x_{t+1}} \log p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)], \end{aligned} \tag{5.14}$$

where the expectations are with respect to the simulated trajectories and Δ denotes the Laplacian operator. In general those equations are intractable, but assuming models with additive noise:

$$\begin{aligned} \mathbf{x}_{t+1} &= \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t) + \mathbf{v}_t \\ \mathbf{y}_t &= \mathbf{h}(\mathbf{x}_t) + \mathbf{w}_t \end{aligned} \tag{5.15}$$

with $\mathbf{v}_t = \mathcal{N}(\mathbf{0}, \mathbf{Q}_t)$ and $\mathbf{w}_t = \mathcal{N}(\mathbf{0}, \mathbf{R}_t)$, the Laplacians in (5.14) can be simplified to:

$$\begin{aligned} \mathbf{A}_{t+1} &= \mathbb{E}[\mathbf{H}^T(\mathbf{x}_{t+1}, \mathbf{y}_{t+1})\mathbf{R}_{t+1}^{-1}(\mathbf{y}_{t+1})\mathbf{H}(\mathbf{x}_{t+1}, \mathbf{y}_{t+1})] \\ \mathbf{B}_t &= \mathbb{E}[\mathbf{F}^T(\mathbf{u}_t)\mathbf{Q}_t^{-1}(\mathbf{u}_t)\mathbf{F}(\mathbf{u}_t)] \\ \mathbf{C}_t &= \mathbb{E}[-\mathbf{F}^T(\mathbf{u}_t)\mathbf{Q}_t^{-1}(\mathbf{u}_t)] \\ \mathbf{D}_t &= \mathbb{E}[\mathbf{Q}_t^{-1}(\mathbf{u}_t)], \end{aligned} \tag{5.16}$$

where the matrices $\mathbf{H}$ and $\mathbf{F}$ denote the Jacobians of the measurement and transition models respectively.

By simulating (sampling) trajectories, using our observation and transition models, one can easily approximate these expectations with Monte Carlo averages. These averages can be computed before the recursion and hence the expensive matrix update of equation (5.13) only needs to be done once for all

scenarios:

$$\begin{aligned}
\mathbf{A}_{t+1} &= \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}(\mathbf{x}_{t+1}^{(i)}, \mathbf{y}_{t+1}^{(i)})\mathbf{H}^T(\mathbf{x}_{t+1}^{(i)}, \mathbf{y}_{t+1}^{(i)})\mathbf{R}_{t+1}^{-1}(\mathbf{y}_{t+1}^{(i)})\mathbf{H}(\mathbf{x}_{t+1}^{(i)}, \mathbf{y}_{t+1}^{(i)}) \\
\mathbf{B}_t &= \frac{1}{N}\sum_{i=1}^{N} \mathbf{F}^T(\mathbf{u}_t^{(i)})\mathbf{Q}_t^{-1}(\mathbf{u}_t^{(i)})\mathbf{F}(\mathbf{u}_t^{(i)}) \\
\mathbf{C}_t &= -\frac{1}{N}\sum_{i=1}^{N} \mathbf{F}^T(\mathbf{u}_t^{(i)})\mathbf{Q}_t^{-1}(\mathbf{u}_t^{(i)}) \\
\mathbf{D}_t &= \frac{1}{N}\sum_{i=1}^{N} \mathbf{Q}_t^{-1}(\mathbf{u}_t^{(i)}),
\end{aligned}$$

where $\mathbb{I}(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$ represents the validation gate of the simulated observation.

In SLAM, the assumption of additive noise is an approximation of the true model. Hence, this can be a potential source of error. An alternative PCRB approximation method that overcomes this shortcoming, in the context of jump Markov linear (JML) models, was proposed by [Bergman 01].

5.4.2 PCRB for jump Markov linear models

An approximation of SLAM problem using Jump Markov Linear (JML) models can be represented as a function of an unobserved process s_t, which, in our case, represents the different sampling scenarios:

$$\begin{aligned}
\mathbf{x}_{t+1} &= \mathbf{F}_t(s_t)\mathbf{x}_t + \mathcal{Q}_t(s_t)\mathbf{v}_t + \mathbf{G}_t(s_t)\mathbf{u}_t \\
\mathbf{y}_t &= \mathbf{H}_t(s_t)\mathbf{x}_t + \mathcal{R}_t(s_t)\mathbf{w}_t
\end{aligned} \tag{5.17}$$

where, for simplicity in the derivation and generality, the noise covariance is also part of the unobserved process s_t, that is, $\mathbf{v}_t = \mathcal{N}(\mathbf{0}, \mathbf{I}_{n_{\mathbf{v}_t}})$, $\mathbf{w}_t = \mathcal{N}(\mathbf{0}, \mathbf{I}_{n_{\mathbf{w}_t}})$, $\mathbf{Q}_t = \mathcal{Q}_t\mathcal{Q}_t^T$ and $\mathbf{R}_t = \mathcal{R}_t\mathcal{R}_t^T$. The the posterior Cramér-Rao bound can be obtained recursively as:

$$C_t^{PCRB} = \mathbf{A}_t\Upsilon_t\mathbf{A}_t^T + (\mathbf{A}_t\Upsilon_t\mathbf{C}_t - \mathbf{B}_t)\Phi_t^{-1}(\mathbf{A}_t\Upsilon_t\mathbf{C}_t - \mathbf{B}_t)^T \tag{5.18}$$

with $\Upsilon_t = (\Upsilon_{t-1}^{-1} + \mathbf{D}_{t-1}^{-1})^{-1}$ and $\Phi_t^{-1} = \Lambda_t^{-1} - \mathbf{C}_t^T\Upsilon_t^1\mathbf{C}_t$. The matrices $\mathbf{A}_t$, $\mathbf{B}_t$, Λ_t, $\mathbf{C}_t$ and $\mathbf{D}_t$ are formed by averaging the matrices from the model (5.17) over the prior distribution of s_t, which represents the randon numbers used to

sample from $\mathbf{x}_t$, $\mathbf{y}_t$ and $\mathbf{u}_t$.

$$\begin{aligned}
\mathbf{A}_t &= \mathbb{E}(\mathbf{G}(\mathbf{s}_t)) \\
\mathbf{B}_t &= \mathbb{E}(\mathcal{Q}(\mathbf{s}_t)) \\
\Lambda_t^{-1} &= \mathbb{E}(-\Delta_{\mathbf{v}_t}^{\mathbf{v}_t} \log p(\mathbf{y}_t|\mathbf{x}_{t-1}, s_t, \mathbf{v}_t) p(v_t)) \\
&= \mathbb{E}(\mathcal{Q}(\mathbf{s}_t)\mathbf{R}_t^{-1}\mathcal{Q}(\mathbf{s}_t)) + \mathbf{I}_{n_{\mathbf{v}_t}} \\
\mathbf{D}_t^{-1} &= \mathbb{E}(-\Delta_{\mathbf{x}_{t-1}}^{\mathbf{x}_{t-1}} \log p(\mathbf{y}_t|\mathbf{x}_{t-1}, s_t, \mathbf{v}_t)) \\
&= \mathbb{E}(\mathbf{F}(\mathbf{s}_t)\mathbf{R}_t^{-1}\mathbf{F}(\mathbf{s}_t)) \\
\mathbf{C}_t^T &= \mathbb{E}(-\Delta_{\mathbf{v}_t}^{\mathbf{x}_{t-1}} \log p(\mathbf{y}_t|\mathbf{x}_{t-1}, s_t, \mathbf{v}_t)) \\
&= \mathbb{E}(\mathcal{Q}(\mathbf{s}_t)\mathbf{R}_t^{-1}\mathbf{F}(\mathbf{s}_t)),
\end{aligned}$$

Again, we can approximate these expectations based on the Monte Carlo samples over the scenarios $s_t^{(i)}$.

$$\begin{aligned}
\mathbf{A}_t &= \frac{1}{N}\sum_{i=1}^{N}(\mathbf{G}(s_t^{(i)})) \\
\mathbf{B}_t &= \frac{1}{N}\sum_{i=1}^{N}(\mathcal{Q}(s_t^{(i)})) \\
\Lambda_t^{-1} &= \frac{1}{N}\sum_{i=1}^{N}(\mathbb{I}(\mathbf{s}_t^{(i)})\mathcal{Q}(s_t^{(i)})\mathbf{R}_t^{-1}(s_t^{(i)})\mathcal{Q}(s_t^{(i)})) + \mathbf{I}_{n_{\mathbf{v}_t}} \\
\mathbf{D}_t^{-1} &= \frac{1}{N}\sum_{i=1}^{N}(\mathbb{I}(\mathbf{s}_t^{(i)})\mathbf{G}(s_t^{(i)})\mathbf{R}_t^{-1}(s_t^{(i)})\mathbf{G}(s_t^{(i)})) \\
\mathbf{C}_t^T &= \frac{1}{N}\sum_{i=1}^{N}(\mathbb{I}(\mathbf{s}_t^{(i)})\mathcal{Q}(s_t^{(i)})\mathbf{R}_t^{-1}(s_t^{(i)})\mathbf{G}(s_t^{(i)})),
\end{aligned}$$

The AMSE simulation approach of Section 5.2.1 using the EKF requires that we perform an expensive Ricatti update (EKF covariance update) for each simulated trajectory. In contrast, the simulation approach using the PCRB only requires one Ricatti update (equation (5.13) or (5.18)). Thus, the latter approach is considerably cheaper. Yet, the PCRB is only a lower bound and hence it is not guaranteed to be necessarily tight. Also, both PCRB are approximations of the exact bound. In the following section, we will provide empirical comparisons between the three simulation approaches, called AMSE, (nonlinear) NL-PCRB and (jump Markov linear) JML-PCRB respectively.

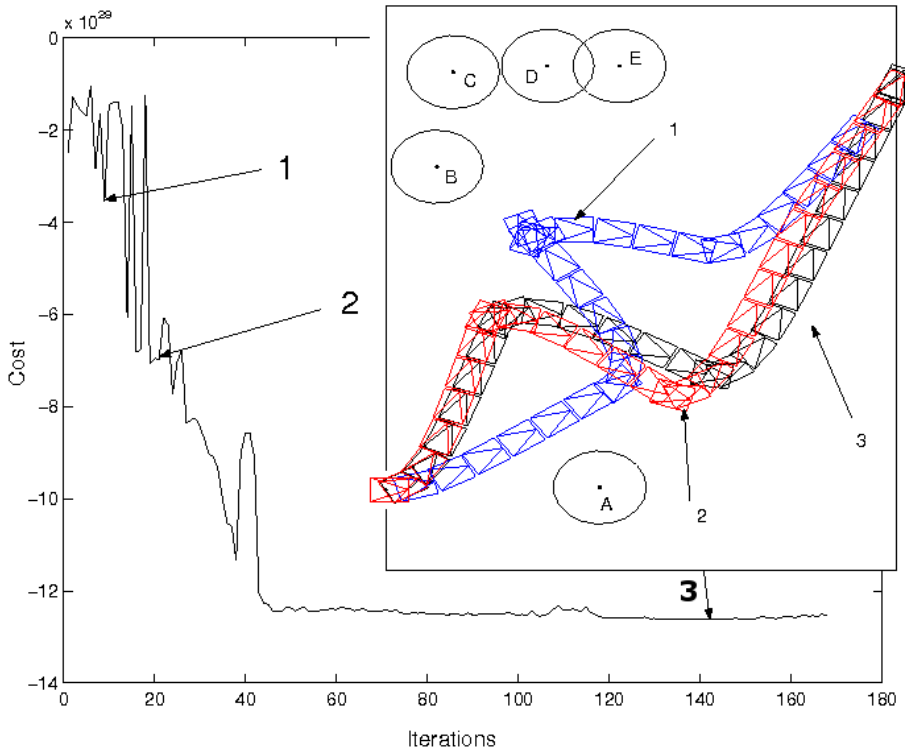


Figure 5.7: Empirical AMSE cost as a function of policy improvement iterations. For this 6D parameter space, the solution converges to the minimum in almost 40 iterations. The figure also shows the actual computed trajectories at three different iteration steps.

5.5 Experiments

We present two sets of experiments. The first experiment is very simple as it is aimed at illustrating the approach. It involves a fixed-horizon stochastic planning domain. The second set of experiments is concerned with exploration with receding horizon policies in more realistic settings. In all cases, the aim is to find the optimal path in terms of posterior information about the map and robot pose. For clarification, other terms contributing to the cost, such as time and obstacles are not considered, but the implementation should be straightforward.

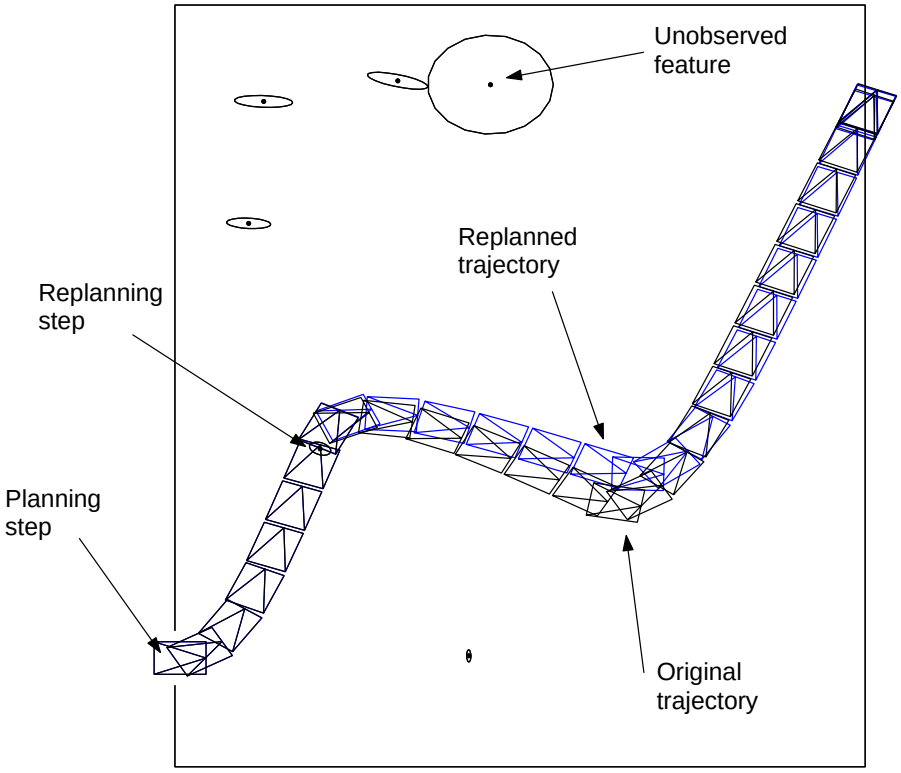


Figure 5.8: Replanning trajectory after reaching the first way-point. Due to the localization and mapping uncertainty, one of the landmarks remains unobserved. Consequently, the replanning system adapt the original planned trajectory to increase the likelihood of seing that landmark. However, it tries also to remain close to the landmark with the lowest uncertainty, to improve localization.

5.5.1 Fixed-horizon planning

The first experiment is the one that we introduced in figure 5.1. There, the start and end positions of the path are fixed. The robot has to compute the coordinates of three intermediate way-points and, hence, the policy has six parameters. For illustration purposes we chose a simple environment consisting of 5 landmarks (with vague priors). We placed an informative prior on the initial robot pose. When the robot fails to reach the target within a specified time, the corresponding simulation is rejected. This rejection sampling mechanism works well in this very simple simulation setting.

Figure 5.7 shows three different robot trajectories computed during policy

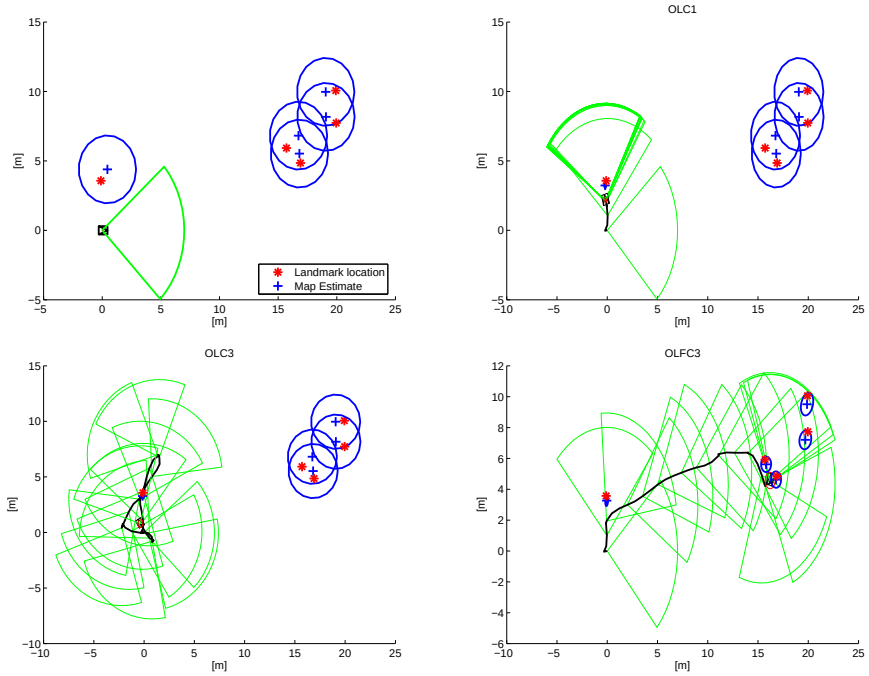


Figure 5.9: Trajectories generated using OLC1, OLC3 and OLFC3. The blue and red ellipses represent the landmark and robot location 95% confidence intervals. The robot field of view is shown in green. OLC3 is more exploratory than OLC1, which gets stuck repeatedly updating the first landmark it encounters. Yet, only OLFC3, because of being able to replan at each step, is able to fully explore the map and reduce the uncertainty.

optimization. The trajectories are also indicated in the Monte Carlo AMSE cost evolution plot. The 6D optimization requires less than 50 iterations. We found that the optimal trajectory allowed the robot to observe the maximum number of features. However, since the prior on the robot's initial pose is very informative, it is narrow Gaussian, feature A is originally detected with very low uncertainty (see figure 5.7). Consequently, the robot tries to maintain that feature in the field of view to improve the localization. A greedy strategy would have focused only on feature A, improving the estimation of that feature and the robot, but dropping the global posterior estimate. Figure 5.8 shows the process of replanning the trajectory after reaching the first way-point. We can see how the robot has missed one of the features (feature E), mainly due to the high noise in the prior map. As a result, the re-planning algorithm

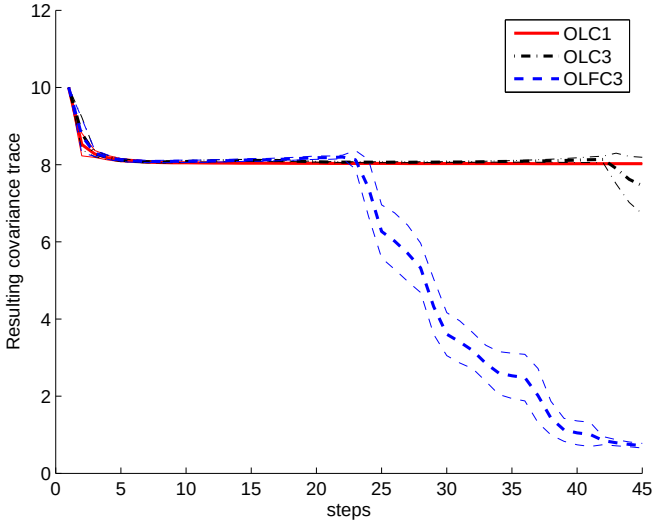


Figure 5.10: Evolution of the trace of the state covariance matrix for 15 runs on the trap experiment using OLC1, OLC3 and OLFC3, with 99% confidence intervals.

changes the planned path so as to increase the likelihood of observing that feature and reduce the uncertainty. Nevertheless, the robot remains close to feature A to improve the localization as mentioned in the previous paragraph.

5.5.2 Receding-horizon planning

We have created a simulated testbed for preliminary experimental validations. In this setup, the environment is a free space area with several point features distributed at random. An a priory map is known with very high uncertainty $\sigma_x = \sigma_y = 1$m. The robot is a differential drive vehicle equipped with odometers and a stereo camera that provides the location of features. The field of view is limited to 7 meters and 90^o, which are typical values for reliable stereo matching. We assume that the camera and a detection system that provides a set of observations every 0.5 seconds. The sensor noise is Gaussian for both range and bearing, with standard deviations $\sigma_{range} = 0.2 \cdot range$ and $\sigma_{bearing} = 0.5^o$. The policy is given by a set of ordered way-points. Each way-point is defined in terms of heading and distance with respect to the robot pose at the preceding way-point. The distance between way-points is limited to 10 meters and the heading should be in the interval $[-3\pi/4, 3\pi/4]$ to avoid

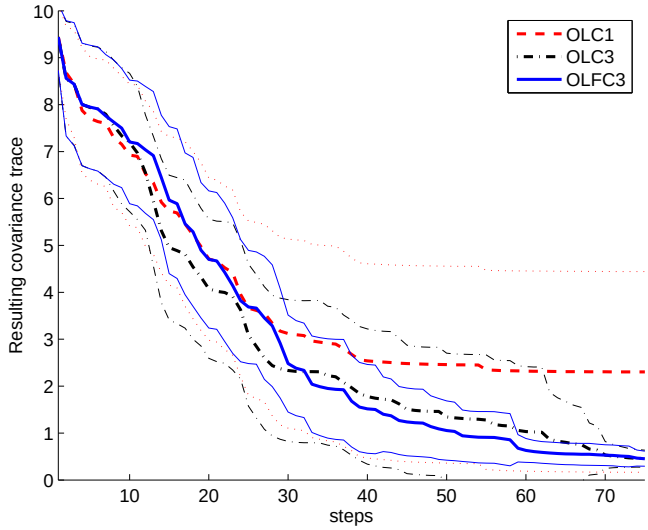


Figure 5.11: Evolution of the trace of the state covariance matrix for 15 random maps using OLC1, OLC3 and OLFC3, with 99% confidence intervals.

backwards trajectories. The motion commands are computed by a controller to guarantee that the goal is reached in 10 seconds.

First, we compare the behavior of the robot using different planning and acting horizons. The three methods that we implemented are:

OLC1 : This is an open loop greedy algorithm that plans with only 1 way-point ahead.

OLC3 : This is an open loop algorithm that plans with 3 way-points ahead.

OLFC3 : This is an open loop feedback controller with a receding horizon. The planning horizon is 3 way-points, but the execution horizon is only 1 step. Thus, the last 2 way-points plus a new way-point are recomputed after a way-point is reached (re-planning).

It is obvious that the OLC algorithms have a lower computational cost. Using the AMSE cost and the map of figure 5.9, the times for OLC1, OLC3 and OLFC3 are approximately 6, 30 and 75 minutes (using an un-optimized Matlab implementation). On the other hand, they can get easily trapped in local minima, as can be clearly shown in figure 5.9. Due to the limited planning horizon of OLC1, it barely explores new areas. This behavior can

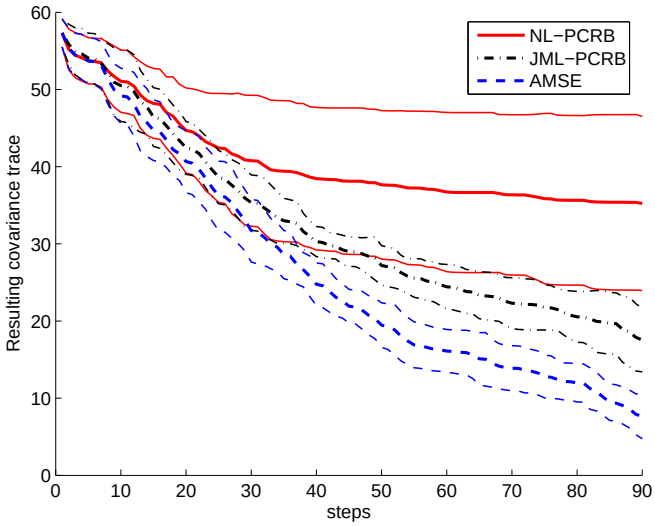


Figure 5.12: Evolution of the trace of the state covariance matrix for 15 runs with AMSE, Tychavsy-PCRB and Bergman-PCRB cost function using OLFC3.

be catastrophic when the distribution of landmarks is sparse. OLC3 tends to overshoot as it only replans at the third way-point. OLFC3, on the other hand, replans at each step and as a result is able to steer to the unexplored part of the map. Myopic algorithms, such as OLC1 and OLC3, use past observations to compute the path. However, new information during the execution of the trajectory is neglected. Figure 5.10 plots the evolution of the uncertainty in the trap situation in 15 runs. The controller with feedback is clearly the winner because it avoids the trap situation. This behavior is very stable across different runs.

As a generalization of previous results, we run the same experiment with different random landmark distributions (with 5 landmarks). It can be seen, in figure 5.11, that open loop approaches perform greedy and obtaining good results in the first steps. However, OLFC3 is able to generate more informative trajectories, and it is also more stable -tight error bounds-.

The previous experiments were based on AMSE cost function with high number of samples to reduce the approximation error. In this point, it is important to note that for long planning horizons and high uncertainties, inconsistency issues may appear, therefore, standard techniques for consistency improvements should be applied for the AMSE cost computations like those

presented in chapters 3 and 4. In general, AMSE approach is intractable for a real robot due to the high computational cost.

The next set of simulations is used to experimentally validate the PCRB approximation to the AMSE cost. Therefore, we increase the size and the complexity of the environment to 30 landmarks in a 25x25 meters squared area. Figure 5.12 shows the covariance matrix evolution of the map and the robot estimated using OLFC3 and the different cost functions presented in this paper. JML-PCRB remains close to the AMSE behavior, with narrow confidence intervals. However, NL-PCRB results in poor performance. This suggests, but does not confirm, that JML-PCRB is a tight bound in this exploration domain. This is particularly encouraging since the PCRB is a much cheaper simulation method and opens up room for real-time implementation.

5.6 Conclusions

We have presented an approach for stochastic exploration and planning rooted in strong statistical and decision-theoretic foundations. The next step is to test the proposed simulator on a real robotic domain. We also note that our method is directly applicable to the problem of planning the architecture of a dynamic sensor network. In terms of modelling, we need to introduce richer cost functions and constraints. In terms of algorithm improvement, we must design infill optimization strategies for high-dimensional policies. We also need to design non-parametric regression processes that are more suitable for handling discontinuities than GPs. Whenever gradients are available, the approach presented here could be improved by ensuring that the regression function matches the gradients at the query points. Finally, on the theoretical front, we plan to build upon early work on correlated bandits to obtain theoretical performance bounds.

Chapter 6

Conclusions

After twenty years of research and development, Simultaneous Localization And Mapping (SLAM) has become one of the most theoretical challenging problems in robotics. The recursive nature of the problem combined with the inherent correlation between all the variables in the problem requires a deep control of error and approximations. A small approximation error in the algorithm, propagated in time and through other variables may be catastrophic. Although there are promising implementations and algorithms in the literature, achieving excellent results, the lack of a strong theoretical background has created some concerns to the reliability of a commercial platform. From the theoretical point of view, the statistical consistency of the results is not clear.

In this thesis, we have provided some insights from the statistical and algorithmical point of view. We have analized the source of the problems, hidden in the roots and the fundamentals of the algorithms, and we have provided some clarification on the *hows* and *whys* of those problems. We have seen that the main difficulties arise from three properties of SLAM: nonlinearity of the models, non-Gaussianity of the underlying probability distributions and nonstationarity of the dynamic part of the system.

We started the journey by reviewing the fundamentals of the most extended algorithm in the field, the Extended Kalman Filter (EKF). We saw that the source of the statistical inconsistency is related to the linearization of the models and the underlying uncertainty that is propagated through. Thus, we provided improvements in both cases. First, we exploited the structure of the SLAM problem to reduce the uncertainty that has to be propagated through the models. Then, we validated different linearization techniques, which are specially intended for probabilistic models. The experimental results showed

some improvement over the standard methods, being more robust and reliable.

However, the theoretical consistency was still missing. Thus, we analized another technique with promising results in the robotics and machine learning fields: the Sequential Monte Carlo (SMC) methods. Those methods are specially suited for nonlinear and non-Gaussian problems like SLAM. Although being a numerical method, and therefore, with an inherent approximation error, it is not propagated in time. However, certain conditions must be satisfied before that claim. Some authors, after the astonishing results that SMC methods were providing in other fields, provided some implementations in SLAM, based on a straightforward modifications of the EKF algorithms. The results were good, but not as expected.

In this thesis, we proved that the required conditions for statistical consistency in SMC methods that are are not fulfilled in current SLAM implementations. We also analized that the inconsistency of the algorithm came from the fact that the Monte Carlo algorithm is based on the EKF, combining state variables and static parameters in a extended state space. This is a valid trick for EKF and related algorithms, but it jeopardizes the nice statistical properties of SMC algorithms.

Therefore, we proposed Marginal-SLAM, a SLAM algorithm that treats state and parameters variables separately. However, the lack of a full Bayesian algorithm for joint state and parameter estimation in general models, constraints our algorithm to a Maximum Likelihood (ML) algorithm. But, standard SMC methods still fail to learn maximum likelihood parameters. For that reason, we introduced some advanced Monte Carlo methods. This algorithm had none of the problems found in this thesis. It could deal with nonlinear, non-Gaussian, joint state and parameter estimation. It is a new promising field for SLAM.

Nevertheless, joint estimation has some extra requirements. Since there is no algorithm to estimate the belief of the parameters, we are limited to point-based approximations. Thus, nonstationarity appeared as a new source of errors.

Finally, our journey ended by addressing the problem of active SLAM. The aim of SLAM is to be used by autonomous vehicles and robots, where they can take decisions and move following some sort of optimality. We analized the idea of using the robot decisions to reduce the uncertainty and, therefore, overcome the inconsistency issues in an active way. However, the problem of finding the optimal decisions is based on a function that is unknown and really time consuming to evaluate. We presented a new algorithm, called Active Policy Learning, for optimizing such kind of functions using response

surfaces to reduce the number of evaluations. We also introduced a cheaper evaluation method to measure the information, and therefore, the uncertainty reduction, based on the Posterior Cramer-Rao Bound (PCRB). The validity of those methods was tested on carefully designed simulations.

As a future work, we continue analizing the theoretical insights of the Marginal-SLAM algorithm, trying to find a solution of the nonstationarity restriction. Some interesting results in measure theory seems to provide a way to overcome that limitation. On the other hand, we plan to continue improving the Active Policy Learning algorithm. Also, we want to find a mathematical proof to guarantee the convergence of the algorithm.

Appendix A

Transformation and Jacobians in 2D

Two basic operations used in stochastic mapping are transformation inversion and composition, which were represented by [Smith 88] using operators $\ominus$ and $\oplus$:

$$\hat{\mathbf{x}}_A^B = \ominus \hat{\mathbf{x}}_B^A$$
$$\hat{\mathbf{x}}_C^A = \hat{\mathbf{x}}_B^A \oplus \hat{\mathbf{x}}_C^B$$

The Jacobians of these operations are defined as:

$$\mathbf{J}_{\ominus}\left\{\hat{\mathbf{x}}_B^A\right\} = \left.\frac{\partial\left(\ominus \mathbf{x}_B^A\right)}{\partial \mathbf{x}_B^A}\right|_{(\hat{\mathbf{x}}_B^A)}$$
$$\mathbf{J}_{1\oplus}\left\{\hat{\mathbf{x}}_B^A,\, \hat{\mathbf{x}}_C^B\right\} = \left.\frac{\partial\left(\mathbf{x}_B^A \oplus \mathbf{x}_C^B\right)}{\partial \mathbf{x}_B^A}\right|_{(\hat{\mathbf{x}}_B^A, \hat{\mathbf{x}}_C^B)}$$
$$\mathbf{J}_{2\oplus}\left\{\hat{\mathbf{x}}_B^A,\, \hat{\mathbf{x}}_C^B\right\} = \left.\frac{\partial\left(\mathbf{x}_B^A \oplus \mathbf{x}_C^B\right)}{\partial \mathbf{x}_C^B}\right|_{(\hat{\mathbf{x}}_B^A, \hat{\mathbf{x}}_C^B)}$$

In 2D, the location of a reference B relative to a reference A (or transformation from A to B) can be expressed using a vector with three d.o.f.: $\mathbf{x}_B^A = [x_1, y_1, \phi_1]^T$. The location of A relative to B is computed using the

inversion operation:

$$\mathbf{x}_A^B = \ominus \mathbf{x}_B^A = \begin{bmatrix} -x_1 \cos\phi_1 - y_1 \sin\phi_1 \\ x_1 \sin\phi_1 - y_1 \cos\phi_1 \\ -\phi_1 \end{bmatrix}$$

The Jacobian of transformation inversion is:

$$\mathbf{J}_{\ominus}\{\mathbf{x}_B^A\} = \begin{bmatrix} -\cos\phi_1 & -\sin\phi_1 & -x_1 \sin\phi_1 - y_1 \cos\phi_1 \\ \sin\phi_1 & -\cos\phi_1 & x_1 \cos\phi_1 + y_1 \sin\phi_1 \\ 0 & 0 & -1 \end{bmatrix}$$

Let $\mathbf{x}_C^B = [x_2, y_2, \phi_2]^T$ be a second transformation. The location of reference C relative to A is obtained by the composition of transformations $\mathbf{x}_B^A$ and $\mathbf{x}_C^B$:

$$\mathbf{x}_C^A = \mathbf{x}_B^A \oplus \mathbf{x}_C^B = \begin{bmatrix} x_1 + x_2 \cos\phi_1 - y_2 \sin\phi_1 \\ y_1 + x_2 \sin\phi_1 + y_2 \cos\phi_1 \\ \phi_1 + \phi_2 \end{bmatrix}$$

The Jacobians of transformation composition are:

$$\mathbf{J}_{1\oplus}\{\mathbf{x}_B^A, \mathbf{x}_C^B\} = \begin{bmatrix} 1 & 0 & -x_2 \sin\phi_1 - y_2 \cos\phi_1 \\ 0 & 1 & x_2 \cos\phi_1 - y_2 \sin\phi_1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{J}_{2\oplus}\{\mathbf{x}_B^A, \mathbf{x}_C^B\} = \begin{bmatrix} \cos\phi_1 & -\sin\phi_1 & 0 \\ \sin\phi_1 & \cos\phi_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

A generalization of the $\oplus$ operator to also represent the composition of transformations with feature location vectors, which results in the change of base reference of the feature, can be found in [Tardós 02].

Appendix B

Kernel Methods for Learning

In this appendix we explain some kernel based techniques for regresion and clustering that appeared in the thesis.

B.1 Kernel trick

Solving a linear regresion problem like:

$$f(\mathbf{x}) = \mathbf{x}^T\mathbf{w} \qquad y = f(\mathbf{x}) + \varepsilon$$

where $\mathbf{x}$ is the input vector, $\mathbf{w}$ is the vector of parameters of the lineal model and y is the observed value; is trivial.

When the observations are perturbed by some white Guassian noise ε, the problem is ill-posed, but we can still solve it analytically. From a Bayesian view point, we define a prior distribution over the parameters $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \Sigma)$. Then we can compute the maximum a posteriori (MAP) estimate of the parameters. This is equivalent to the Tikhonov regularization.

However, linear models are very limited. A simple idea to extend the linear model applicability to nonlinear problems consists in projecting the inputs into a high dimensional space, where the linear model can be directly applied. For example, polinomial regression of a scalar x can be achieved using a linear model in the space of powers of x, i.e. $\phi(x) = (1, x, x^2, x^3, \dots)^T$. Then, the regression function is:

$$f(\mathbf{x}) = \phi(\mathbf{x})^T\mathbf{w}$$

If the mapping function ϕ is independent of the parameters $\mathbf{w}$, then the model become linear in the space of powers of x. Furthermore, the feature space only

appears in the posterior distribution in the form of:

$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \Sigma \phi(\mathbf{x}')$$

which is called the *kernel*. The kernel can also be seen as the result of a dot product in the high dimensional space. If the algorithm is defined solely in terms of dot products in the input space, it can be replaced by the kernel function. Thus, the feature space is never explicitly computed. This is desirable, because, as we have seen in the polynomial example, the high-dimensional space may be infinite-dimensional [Rasmussen 06].

A similar problem which is solved using the kernel trick is the kernel smoothing. The difference with respect to the linear model is that the kernel smoothing is a linear combination on the training set $\mathbf{y}$, that is:

$$f(\mathbf{x}) = \phi(\mathbf{x})^T \mathbf{y}$$

Also, the kernel smoother use an stationary kernel function centered on each data point.

B.2 Type of kernels

Selecting the specific kernel function for a concrete application or algorithm can be tricky. There are many kernel functions in the literature with interesting propperties. Here, we will show a brief survey of stationary kernel functions, where the covariance is expresed in terms of $\xi = (|\mathbf{x} - \mathbf{x}'|)/h$, being h the so-called scale factor. The kernels presented in this section can be used both for regression or smoothing. In general, regression admits any kind of covariance function as a kernel, even nonstationary or anisotropic. On the other hand, smoothing kernels does not need to be covariance functions.

B.2.1 Epanechnikov kernel

This is the simplest kernel in terms of a dot product:

$$k^{Epa}(\xi) = \begin{cases} \frac{1}{2} c_d^{-1}(d+2)(1-\xi^T\xi) & \text{if } \xi^T\xi \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{B.1}$$

where c_d is the volume of a d-dimensional hypersphere of unit radius (i.e: $c_1 = 2, c_2 = \pi$). In our setup, being $\xi = (|\mathbf{x} - \mathbf{x}'|)/h$, the kernel function is an hipersphere $S_h(\mathbf{x})$ defined in the sampling space centered at $\mathbf{x}$ with radius h.

Despite its simplicity, it provides excelent results for gradient estimation in terms of efficiency and accuracy. Theorethically, this kernel provides the minimum integrated squared error (MISE).

B.2.2 Exponential/Gaussian kernel

The Gaussian kernel, sometimes refered as the squared exponential, is the most widely-used kernel in the Bayesian community. It is a special case of the exponential kernel for $p = 2$, where the exponential kernel is:

$$k^{exp}(\xi, p) = \exp\left(-\frac{(\xi^T \xi)^p}{2}\right) \tag{B.2}$$

The exponential kernel is less flexible than other kernels, like the Matérn kernel (see below), because it is not mean square differentiable except for $p = 2$, when it is infinitely differentiable. It has been argued that the Gaussian kernel tends to oversmooth the process [Stein 99]. Also, Sasena [Sasena 02] found that the Gaussian kernel becomes numerically unstable when $\xi \to 0$.

B.2.3 Matérn kernel

The Matérn class of kernels is given by:

$$k^{Mat}(\xi, \nu) = \frac{2^{1-\nu}}{\Gamma\nu}\left(\sqrt{2\nu}\xi\right)^{\nu} K_\nu\left(\sqrt{2\nu}\xi\right) \tag{B.3}$$

where K_ν is a modified Bessel function [Rasmussen 06] and ν is an extra kernel parameter related to the differentiability of the regression function. That is, the function $f(\mathbf{x})$ is k-times mean square differentiable if and only if $\nu > k$. Note that, the Matérn kernel for $\nu \to \infty$ is equivalent to the Gaussian kernel.

The most interesting cases for learning are $\nu = 3/2$ and $\nu = 5/2$, which are sometimes named as the Matérn linear and cubic kernel.

$$k^{Mat}(\xi, \nu = 3/2) = (1 + \sqrt{3}\xi)\exp(-\sqrt{3}\xi) \tag{B.4}$$

$$k^{Mat}(\xi, \nu = 5/2) = \left(1 + \sqrt{5}\xi + \frac{5}{3}\xi^2\right)\exp(-\sqrt{5}\xi) \tag{B.5}$$

B.2.4 Piecewise polynomial kernels

Finally, an interesting family of kernels is the piecewise polynomial or spline kernels with compact support. For instance, a cubic spline kernel can be defined as:

$$k^{Spl}(\xi) = \begin{cases} 1 - 6\xi^2 + 6\xi^3 & \text{if } |\xi| < 1/2 \\ 2((1-\xi)^3) & \text{if } 1/2 \leq |\xi| < 1 \\ 0 & \text{otherwise} \end{cases} \tag{B.6}$$

These kernels, like the Epanechnikov kernel, are exactly zero except in a neighbourhood. For Bayesian regression, the posterior covariance will be sparse, having some computational advantages.

Bibliography

[Andrade-Cetto 05] J. Andrade-Cetto, T. Vidal-Calleja & A. Sanfeliu. *Unscented Transformation of Vehicle States in SLAM.* In Proc. of the IEEE Int. Conf. on Robotics & Automation, pages 324–329, Barcelona, Spain, April 2005.

[Andrieu 99] C. Andrieu, N. de Freitas & A. Doucet. *Sequential MCMC for Bayesian Model Selection.* In IEEE Higher Order Statistics Workshop, pages 130–134, Caesarea, Israel, 1999.

[Bailey 06a] T. Bailey, J. Nieto, J. Guivant, M. Stevens & E. Nebot. *Consistency of the EKF-SLAM Algorithm.* In Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2006.

[Bailey 06b] T. Bailey, J. Nieto & E. Nebot. *Consistency of the FastSLAM Algorithm.* In Proc. of the IEEE Int. Conf. on Robotics & Automation, 2006.

[Bar-Shalom 01] Y. Bar-Shalom, X. Rong Li & T. Kirubarajan. Estimation with applications to tracking and navigation. Wiley InterScience, 2001.

[Baxter 01] J. Baxter & P.L. Bartlett. *Infinite-Horizon Policy-Gradient Estimation.* J. of AI Research, vol. 15, pages 319–350, 2001.

[Beevers 07] K. Beevers & Huang W.H. *Fixed-lag Sampling Strategies for Particle Filtering SLAM.* In Proc. of the IEEE Int. Conf. on Robotics & Automation, 2007.

[Benveniste 07] A. Benveniste & L. Mevel. *Nonstationary Consistency of Subspace Methods*. IEEE Trans. Autom. Control, vol. 52, no. 6, pages 974–984, June 2007.

[Bergman 99] N. Bergman. *Recursive Bayesian Estimation: Navigation and Tracking Applications*. PhD thesis, Linköping University, 1999.

[Bergman 01] N. Bergman, A. Doucet & N.J. Gordon. *Optimal Estimation and Cramér-Rao Bounds for Partial Non-Gaussian State Space Models*. Ann. Inst. Stat. Math., vol. 52, no. 1, pages 1–17, 2001.

[Bertsekas 95] D.P. Bertsekas. Dynamic programming and optimal control. Athena Scientific, 1995.

[Bosse 03] M. Bosse, P. Newman, J. Leonard, M. Soika, W. Feiten & S. Teller. *An Atlas Framework for Scalable Mapping*. In Proc. of the IEEE Int. Conf. on Robotics & Automation, pages 1899–1906, Taipei, Taiwan, 2003.

[Brunskill 05] E. Brunskill & N. Roy. *SLAM using Incremental Probabilistic PCA and Dimensionality Reduction*. In Proc. of the IEEE Int. Conf. on Robotics & Automation, Barcelona, Spain, 2005.

[Bryson 08] M. Bryson & S. Sukkarieh. *Observability Analysis and Active Control for Airborne SLAM*. IEEE Trans. Aerosp. Electron. Syst., vol. 44, no. 1, pages 261–280, January 2008.

[Cacoullos 66] T. Cacoullos. *Estimation of a multivariate density*. Ann. Inst. Stat. Math., vol. 18, pages 179–189, 1966.

[Castellanos 99] J. A. Castellanos, J. M. M. Montiel, J. Neira & J.D. Tardós. *The SPmap: A Probabilistic Framework for Simultaneous Localization and Map Building*. IEEE Trans. Robot. Autom., vol. 15, no. 5, pages 948–952, 1999.

[Castellanos 04] J.A. Castellanos, J. Neira & J.D. Tardós. *Limits to the Consistency of EKF-based SLAM*. In Proc. of the

5th IFAC Symp. on Intelligent Autonomous Vehicles, Lisbon, July 2004.

[Castellanos 07] J.A. Castellanos, R. Martinez-Cantin, J.D. Tardós & J. Neira. *Robocentric Map Joining: Improving the Consistency of EKF-SLAM.* Robotics and Autonomous Systems, vol. 55, pages 21–29, 2007.

[Chaloner 95] K Chaloner & I Verdinelli. *Bayesian experimental design: A review.* J. of Statistical Science, vol. 10, pages 273–304, 1995.

[Cheng 95] Y. Cheng. *Mean Shift, Mode Seeking, and Clustering.* IEEE Trans. Pattern Anal. Mach. Intell., vol. 17, no. 8, pages 790–799, August 1995.

[Chong 99] K. S. Chong & L. Kleeman. *Mobile Robot Map Building for an Advanced Sonar Array and Accurate Odometry.* Int. J. Robotics Research, vol. 18, no. 1, pages 20–36, 1999.

[Civera 08] J. Civera, A.J. Davison & J.M.M. Montiel. *Inverse Depth Parametrization for Monocular SLAM.* IEEE Trans. Robot., 2008. to appear.

[Clark 99] J.M.C. Clark, D.L. Ocone & C. Coumarbatch. *Relative Entropy and Error Bounds for Filtering of Markov Processes.* Math. of Control, Signals, and Systems, vol. 12, no. 4, pages 346–360, Nov. 1999.

[Comaniciu 02] D. Comaniciu & P. Meer. *Mean Shift: A Robust Approach toward Feature Space Analysis.* IEEE Trans. Pattern Anal. Mach. Intell., vol. 24, no. 5, pages 603–619, 2002.

[Coquelin 07] P.A. Coquelin, R. Deguest & R. Munos. *Numerical methods for sensitivity analysis of Feynman-Kac models.* Technical report, INRIA, 2007.

[Cox 64] H. Cox. *On the estimation of state variables and parameters for noisy dynamic systems.* IEEE Trans. Autom. Control, vol. AC-9, pages 5–12, Feb. 1964.

[Crisan 02] D. Crisan & A. Doucet. *A Survey of Convergence Results on Particle Filtering for Practitioners.* IEEE Trans. Signal Process., vol. 50, no. 3, pages 736–746, 2002.

[Davison 05] A.J. Davison. *Active Search for Real-Time Vision.* In Proc. of the Int. Conf. on Computer Vision, 2005.

[Dellaert 06] F. Dellaert & M. Kaess. *Square Root SAM: Simultaneous Localization and Mapping via Square Root Information Smoothing.* Int. J. Robotics Research, vol. 25, no. 12, pages 1181–1204, 2006.

[Delyon 96] B. Delyon. *General results on the Convergence of Stochastic Algorithms.* IEEE Trans. Autom. Control, vol. 41, no. 9, pages 1245–1255, September 1996.

[Dissanayake 01] M.W.M.G. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte & M. Csorba. *A Solution to the Simultaneous Localization and Map Building (SLAM) Problem.* IEEE Trans. Robot. Autom., vol. 17, no. 3, pages 229–241, 2001.

[Doucet 98] A Doucet. *On Sequential Simulation-Based Methods for Bayesian Filtering.* Technical report CUED/F-INFENG/TR 310, Department of Engineering, Cambridge University, 1998.

[Doucet 00] A Doucet, N de Freitas, K Murphy & S Russell. *Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks.* In Proc. of the Sixteenth Conf. on Uncertainty in Artificial Intelligence, 2000.

[Doucet 01] A. Doucet, N. de Freitas & N.J. Gordon, editors. Sequential Monte Carlo methods in practice. Springer-Verlag, 2001.

[Doucet 06] A. Doucet, M. Briers & S. Senecal. *Efficient Block Sampling Strategies for Sequential Monte Carlo.* J. of Comp. and Graph. Stat., vol. 15, no. 3, pages 693–711, 2006.

[Duda 01] R.O. Duda, P.E. Hart & D.G. Stork. Pattern classification. Wiley InterScience, 2001.

[Elfes 87] A. Elfes. *Sonar-Based Real-World Mapping and Navigation.* IEEE Trans. Robot. Autom., vol. 3, no. 3, pages 249–265, 1987.

[Eliazar 05] A.I. Eliazar & R. Parr. *Hierarchical Linear/Constant Time SLAM using Particle Filters for Dense Maps.* In Advances in Neural Information Processing Systems, 2005.

[Elinas 06] P. Elinas, R. Sim & J. Little. *σSLAM: Stereo Vision SLAM Using the Rao-Blackwellised Particle Filter and a Novel Mixture Proposal Distribution,.* In Proc. of the IEEE Int. Conf. on Robotics & Automation, 2006.

[Estrada 05] C. Estrada, J. Neira & J.D. Tardós. *Hierarchical SLAM: real-time accurate mapping of large environments.* IEEE Trans. on Robotics, vol. 21, no. 4, pages 588–596, August 2005.

[Eustice 05] R. Eustice, H. Singh, J. Leonard, M. Walter & R. Ballard. *Visually Navigating the RMS Titanic with SLAM Information Filters.* In Proc. of Robotics: Science and Systems, Cambridge, MA, USA, June 2005.

[Fearnhead 98] P. Fearnhead. *Sequential Monte Carlo Methods in Filter Theory.* PhD thesis, Dept. of Statistics, Oxford University, England, 1998.

[Fearnhead 02] P. Fearnhead. *MCMC, Sufficient Statistics and Particle Filters.* J. of Comp. and Graph. Stat., vol. 11, pages 848–862, 2002.

[Finkel 03] D.E. Finkel. DIRECT optimization algorithm user guide. Center for Research in Scientific Computation, North Carolina State University, 2003.

[Fox 01] D. Fox, S. Thrun, W. Burgard & F. Dellaert. *Particle Filters for Mobile Robot Localization.* In A. Doucet, N. de Freitas & N.J. Gordon, editors, Sequential Monte Carlo Methods in Practice. Springer-Verlag, 2001.

[Frese 05] U. Frese, P. Larsson & T. Duckett. *A Multilevel Relaxation Algorithm for Simultaneous Localization and Mapping.* IEEE Trans. Robot., vol. 21, no. 2, pages 196–207, April 2005.

[Frese 06] U. Frese. *Treemap: An O(logn) Algorithm for Indoor Simultaneous Localization and Mapping.* Autonomus Robots, vol. 21, no. 2, pages 103–122, 2006.

[Fukunaga 75] K. Fukunaga & L.D. Hostetler. *The Estimation of the Gradient of a Density Function, with Applications in Pattern Recognition.* IEEE Trans. Inf. Theory, vol. IT-21, no. 1, pages 32–40, 1975.

[Gablonsky 01] J.M. Gablonsky. *Modification of the DIRECT Algorithm.* PhD thesis, Department of Mathematics, North Carolina State University, Raleigh, North Carolina, 2001.

[Guivant 01] J. Guivant & E. Nebot. *Optimization of the Simultaneous Localization and Map-Building Algorithm for Real-Time Implementation.* IEEE Trans. Robot. Autom., vol. 17, no. 3, pages 242–257, 2001.

[Gutmann 99] J.S. Gutmann & K. Konolige. *Incremental mapping of large cyclic environments.* In Proc. of the IEEE Int. Conf. on Intelligent Robots and Applications, 1999.

[Gutmann 01] H.M. Gutmann. *A radial basis function method for global optimization.* J. of Global Optimization, vol. 19, no. 3, pages 201–227, 2001.

[Hernandez 04a] M.L. Hernandez. *Optimal sensor trajectories in bearings-only tracking.* In Per Svensson & Johan Schubert, editors, Proc. of the Seventh Int. Conf. on Information Fusion, volume II, pages 893–900, Mountain View, CA, Jun 2004. International Society of Information Fusion.

[Hernandez 04b] M.L. Hernandez, T. Kirubarajan & Y. Bar-Shalom. *Multisensor resource deployment using posterior*

Cramèr-Rao bounds. IEEE Trans. Aerosp. Electron. Syst., vol. 40, no. 2, pages 399– 416, April 2004.

[Horowitz 76] S.L. Horowitz & T. Pavlidis. *Picture Segmentation by a Tree Traversal Algorithm*. J. of the ACM, vol. 23, no. 2, pages 368–388, April 1976.

[Hough 62] P.V.C. Hough. *Methods and Means for Recognising Complex Patterns*. US Patent 3 069 654, 1962.

[Ito 00] K. Ito & K. Xiong. *Gaussian Filters for Nonlinear Filtering Problems*. IEEE Trans. Autom. Control, vol. 45, no. 5, pages 910–927, 2000.

[Jazwinski 70] A.H. Jazwinski. Stochastic processes and filtering theory. Academic Press, 1970.

[Jones 93] D.R. Jones, C.D. Perttunen & B.E. Stuckman. *Lipschitzian Optimization Without the Lipschitz Constant*. J. of Optimiz. Theory. App., vol. 79, no. 1, pages 157–181, October 1993.

[Jones 98] D.R. Jones, M. Schonlau & W.J. Welch. *Efficient Global Optimization of Expensive Black-Box Functions*. J. of Global Optimization, vol. 13, no. 4, pages 455–492, 1998.

[Jones 01] D.R. Jones. *A Taxonomy of Global Optimization Methods Based on Response Surfaces*. J. of Global Optimization, vol. 21, pages 345–383, 2001.

[Julier 00] S. Julier, J.K. Uhlmann & H. Durrant-Whyte. *A New Method for the Nonlinear Transformation of Means and Covariances in Filters and Estimators*. IEEE Trans. Autom. Control, vol. 45, no. 3, pages 477–482, March 2000.

[Julier 01] S. Julier & J.K. Uhlmann. *A Counter Example to the Theory of Simultaneous Localization and Map Building*. In Proc. of the IEEE Int. Conf. on Robotics & Automation, pages 4238–4243, Seoul, Korea, 2001.

[Julier 02] S. Julier & J.K. Uhlmann. *The Scaled Unscented Transformation.* In IEEE American Control Conf., pages 4555–4559, Anchorage AK, USA, 8–10 May 2002.

[Julier 04] S. Julier & J.K. Uhlmann. *Unscented Filtering and Nonlinear Estimation.* Proceedings of the IEEE, vol. 92, no. 3, pages 401–422, 2004.

[Kaelbling 90] L.P. Kaelbling. *Learning in Embedded Systems.* PhD thesis, Stanford University, 1990.

[Kaess 07] M. Kaess, A. Ranganathan & F. Dellaert. *iSAM: Fast Incremental Smoothing and Mapping with Efficient Data Association.* In Proc. of the IEEE Int. Conf. on Robotics & Automation, pages 1670–1677, Rome, Italy, 2007.

[Kalman 60] R.E. Kalman. *A new approach to linear filtering and prediction problems.* Transactions of the ASME, vol. 82D, pages 35–45, March 1960.

[Kawahara 06] Y. Kawahara, T. Yairi & K. Machida. *A Kernel Subspace Method by Stochastic Realization for Learning Nonlinear Dynamical Systems.* In Advances in Neural Information Processing Systems, 2006.

[Kim 07] J. Kim & S. Sukkarieh. *Real-time implementation of airborne inertial-SLAM.* Robotics and Autonomous Systems, vol. 55, pages 62–71, 2007.

[Kitagawa 96] G. Kitagawa. *Monte Carlo filter and smoother for non-Gaussian nonlinear state space models.* J. of Comp. and Graph. Stat., vol. 5, no. 1, pages 1–25, 1996.

[Klaas 05] M. Klaas, N. de Freitas & A. Doucet. *Toward Practical N^2 Monte Carlo: The Marginal Particle Filter.* In Proc. of the 21st Conf. on Uncertainty in Artificial Intelligence, 2005.

[Klaas 06] M. Klaas, M. Briers, N. de Freitas, A. Doucet, S. Maskell & D. Lang. *Fast Particle Smoothing: If*

I Had a Million Particles. In Proc. of the Int. Conf. on Machine Learning, 2006.

[Knight 01] J. Knight, A.J. Davison & I. Reid. *Towards Constant Time SLAM using Postponement.* In Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, Acapulco, Mexico, 2001.

[Kohl 04] N. Kohl & P. Stone. *Policy gradient reinforcement learning for fast quadrupedal locomotion.* In Proc. of the IEEE Int. Conf. on Robotics & Automation, 2004.

[Kollar 06] T. Kollar & N. Roy. *Using Reinforcement Learning to Improve Exploration Trajectories for Error Minimization.* In Proc. of the IEEE Int. Conf. on Robotics & Automation, 2006.

[Kopp 63] R.E. Kopp & R.J. Orforf. *Linear regression applied to system identification for adaptive control systems.* AIAA J., vol. 1, no. 10, pages 2300–2306, 1963.

[Krige 51] D.G. Krige. *A Statistical Approach to Some Basic Mine Valuation Problems on the Witwatersrand.* J. of the Chemical, Metallurgical and Mining Society of South Africa, vol. 52, no. 6, pages 119–139, 1951.

[Kushner 64] H.J. Kushner. *A New Method of Locating the Maximum of an Arbitrary Multipeak Curve in the Presence of Noise.* J. of Basic Eng., vol. 86, pages 97–106, 1964.

[Kushner 67] H.J. Kushner. *Approximations to optimal nonlinear filters.* IEEE Trans. Autom. Control, vol. 12, pages 546–556, 1967.

[Larimore 83] W.E. Larimore. *System identification, reduced order filters and modelling via canonical variate analysis.* In Proc. of the Amer. Control Conf., pages 445–451, June 1983.

[Lawrence 03] G. Lawrence, N. Cowan & S. Russell. *Efficient Gradient Estimation for Motor Control Learning.* In Proc. of

the Nineteenth Conf. on Uncertainty in Artificial Intelligence, pages 354–36, San Francisco, CA, 2003. Morgan Kaufmann.

[Lee 98] K.M. Lee, P. Meer & R.H. Park. *Robust Adaptive Segmentation of Range Images.* IEEE Trans. Pattern Anal. Mach. Intell., vol. 20, no. 3, pages 200–205, February 1998.

[LeGland 97] F. LeGland & L. Mevel. *Recursive estimation in hidden Markov models.* In Proc. of the 36th IEEE Conf. on Decision and Control, volume 4, pages 3468–3473, Dec. 1997.

[Leonard 03] J.J. Leonard & P.M. Newman. *Consistent, Convergent and Constant-Time SLAM.* In Proc. of the Int. Joint Conf. on Artificial Intelligence, Acapulco, Mexico, August 2003.

[Leung 05] C Leung, S Huang, G Dissanayake & T Forukawa. *Trajectory Planning for Multiple Robots in Bearing-Only Target Localisation.* In Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2005.

[Liu 01] J. Liu & M. West. *Combined parameter and state estimation in simulation-based filtering.* In A. Doucet, N. de Freitas & N.J. Gordon, editors, Sequential Monte Carlo Methods in Practice. Springer-Verlag, 2001.

[Ljung 77] L. Ljung. *Analysis of Recursive Stochastic Algorithms.* IEEE Trans. Autom. Control, vol. AC-22, pages 551–575, Aug. 1977.

[Ljung 79] L. Ljung. *Asymptotic Behavior of the Extended Kalman Filter as a Parameter Estimator for Linear Systems.* IEEE Trans. Autom. Control, vol. AC-24, pages 36–50, Feb. 1979.

[Locatelli 97] M. Locatelli. *Bayesian Algorithms for One-Dimensional Global Optimization.* J. of Global Optimization, vol. 10, no. 1, pages 57–76, January 1997.

[Maciejowski 02] J.M. Maciejowski. Predictive control: with constraints. Prentice-Hall, 2002.

[Martinez-Cantin 05] R Martinez-Cantin & J.A. Castellanos. *Unscented SLAM for Large-Scale Outdoor Environments.* In Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pages 328–333, 2005.

[Martinez-Cantin 06a] R. Martinez-Cantin & J.A. Castellanos. *Bounding Uncertainty in EKF-SLAM: The Robocentric Local Approach.* In Proc. of the IEEE Int. Conf. on Robotics & Automation, pages 430–435, 2006.

[Martinez-Cantin 06b] R. Martinez-Cantin, J.A. Castellanos, J.D. Tardós & J.M.M. Montiel. *Adaptive Scale Robust Segmentation for 2D Laser Scanner.* In Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2006.

[Maybeck 79] P. Maybeck. Stochastic models, estimation, and control, volume 1. Academic Press, 1979.

[Maybeck 82] P. Maybeck. Stochastic models, estimation, and control, volume 2. Academic Press, 1982.

[Mercère 07] G. Mercère & M. Lovera. *Convergence analysis of instrumental variable recursive subspace identification algorithms.* Automatica, vol. 43, no. 8, pages 1377–1386, 2007.

[Metivier 84] M. Metivier & P. Priouret. *Applications of a Kushner and Clark Lemma to General Classes of Stochastic Algorithms.* IEEE Trans. Inf. Theory, vol. IT-30, pages 140–151, March 1984.

[Mockus 78] J. Mockus, V. Tiesis & A. Zilinskas. *The application of Bayesian methods for seeking the extremum.* In L.C.W. Dixon & G.P. Szego, editors, Towards Global Optimisation 2, pages 117–129. Elsevier, 1978.

[Montemerlo 03a] M. Montemerlo & S. Thrun. *Simultaneous Localization and Mapping with Unknown Data Association using FastSLAM.* In Proc. of the IEEE Int. Conf. on Robotics & Automation, 2003.

[Montemerlo 03b] M. Montemerlo, S. Thrun, D. Koller & B. Wegbreit. *FastSLAM 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Provably Converges*. In Proc. of the Int. Joint Conf. on Artificial Intelligence, Acapulco, Mexico, 2003.

[Montiel 06] J.M.M. Montiel & A.J. Davison. *A Visual Compass based on SLAM*. In Proc. of the IEEE Int. Conf. on Robotics & Automation, pages 1917–1922, 2006.

[Moore 96] A.W. Moore & J. Schneider. *Memory-based Stochastic Optimization*. In David S. Touretzky, Michael C. Mozer & Michael E. Hasselmo, editors, Advances in Neural Information Processing Systems, volume 8, pages 1066–1072. The MIT Press, 1996.

[Mourikis 07] A. I. Mourikis, N. Trawny, S.I. Roumeliotis, A.E. Johnson & L.H. Matthies. *Vision-Aided Inertial Navigation for Precise Planetary Landing: Analysis and Experiments*. In Proc. of Robotics: Science and Systems, Atlanta, GA, June 2007.

[Mozos 05] O. Martinez Mozos, C. Stachniss & W. Burgard. *Supervised Learning of Places from Range Data using AdaBoost*. In Proc. of the IEEE Int. Conf. on Robotics & Automation, pages 1742–1747, 2005.

[Murphy 99] K. Murphy. *Bayesian Map Learning in Dynamic Environments*. In Advances in Neural Information Processing Systems, 1999.

[Neira 01] J. Neira & J.D. Tardós. *Data Association in Stochastic Mapping Using the Joint Compatibility Test*. IEEE Trans. Robot. Autom., vol. 17, no. 6, pages 890–897, 2001.

[Newman 02] P.M. Newman, J.J. Leonard, J. Neira & J.D. Tardós. *Explore and Return: Experimental Validation of Real Time Concurrent Mapping and Localization*. In Proc. of the IEEE Int. Conf. on Robotics & Automation, pages 1802–1809, Washington D.C., May 2002.

[Ng 00] A.Y. Ng & M.I. Jordan. *PEGASUS: A Policy Search Method for Large MDPs and POMDPs.* In Proc. of the Sixteenth Conf. on Uncertainty in Artificial Intelligence, 2000.

[Ng 04] A. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger & E. Liang. *Inverted autonomous helicopter flight via reinforcement learning.* In Proc. of the Int. Symp. on Experimental Robotics, 2004.

[Nguyen 05] V. Nguyen, A. Martinelli, N. Tomatis & R. Siegwart. *A Comparison of Line Extraction Algorithms using 2D Laser Rangefinder for Indoor Mobile Robotics.* In Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2005.

[Nørgaard 00] M. Nørgaard, N.K. Poulsen & O. Ravn. *New Developments in State Estimation for Nonlinear Systems.* Automatica, vol. 36, no. 11, pages 1627–1638, November 2000.

[Olson 07] E. Olson, J. Leonard & S. Teller. *Spatially-Adaptive Learning Rates for Online Incremental SLAM.* In Proc. of Robotics: Science and Systems, Atlanta, GA, USA, June 2007.

[Olsson 08] J. Olsson, O. Cappé, R. Douc & E. Moulines. *Sequential Monte Carlo smoothing with application to parameter estimation in non-linear state space models.* Bernoulli, vol. 14, no. 1, pages 155–179, 2008.

[Paris 02] S. Paris & J.P. Le Cadre. *Planification for Terrain-Aided Navigation.* In Fusion 2002, pages 1007–1014, Annapolis, Maryland, July 2002.

[Parzen 62] E. Parzen. *On estimation of a probability density function and mode.* Ann. Math. Statist., vol. 33, pages 1065–1067, 1962.

[Paskin 03] M. A. Paskin. *Thin Junction Tree Filters for Simultaneous Localization and Mapping.* In G. Gottlob & T. Walsh, editors, Proc. of the Int. Joint Conf. on

Artificial Intelligence, pages 1157–1164. Morgan Kaufmann, 2003.

[Paskin 05] M.A. Paskin & S. Thrun. *Robotic Mapping with Polygonal Random Fields*. In Faheim Bacchus & Tommi Jaakkola, editors, Proc. of the 21st Conf. on Uncertainty in Artificial Intelligence, July 2005.

[Paz 07] L.M. Paz & J. Neira P. Jensfelt J.D. Tardós. *EKF SLAM updates in O(n) with Divide and Conquer SLAM*. In Proc. of the IEEE Int. Conf. on Robotics & Automation, 2007.

[Pearson 01] K. Pearson. *On Lines and Planes of Closest Fit to Systems of Points in Space*. Philoshophical Magazine, vol. 2, pages 559–572, 1901.

[Peters 06] J. Peters & S. Schaal. *Policy Gradient Methods for Robotics*. In Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2006.

[Pitt 99] M K Pitt & N Shephard. *Filtering Via Simulation: Auxiliary Particle Filters*. J. of the Amer. Stat. Assoc., vol. 94, no. 446, pages 590–599, 1999.

[Poyadjis 05a] G. Poyadjis, A. Doucet & S.S. Singh. *Maximum Likelihood Parameter Estimation using Particle Methods*. In Joint Statistical Meeting, 2005.

[Poyadjis 05b] G. Poyadjis, A. Doucet & S.S. Singh. *Particle methods for optimal filter derivative: Application to parameter estimation*. In Proc. of the Int. Conf. on Acoustics, Speech, and Signal Processing, 2005.

[Rasmussen 06] C.E. Rasmussen & C.K.I. Williams. Gaussian processes for machine learning. The MIT Press, Cambridge, Massachusetts, 2006.

[Ribas 08] D. Ribas, P. Ridao, J.D. Tardós & J. Neira. *Underwater SLAM in Man Made Structured Environments*. Journal of Field Robotics, 2008. to appear.

[Rousseeuw 87] P.J. Rousseeuw & A. Leroy. Robust regression and outlier detection. Wiley InterScience, 1987.

[Russell 02] S. Russell & P. Norvig. Artificial intelligence: A modern approachn. Prentice-Hall, 2002.

[Sacks 89] J. Sacks, W.J. Welch, T.J. Mitchell & H.P. Wynn. *Design and Analysis of Computer Experiments.* Statistical Science, vol. 4, no. 4, pages 409–423, 1989.

[Santner 03] T.J. Santner, B. Williams & W. Notz. The design and analysis of computer experiments. Springer-Verlag, 2003.

[Sasena 02] M.J. Sasena. *Flexibility and Efficiency Enhancement for Constrained Global Design Optimization with Kriging Approximations.* PhD thesis, University of Michigan, 2002.

[Schonlau 98] M. Schonlau, W. Welch & D. Jones. *Global Versus Local Search in Constrained Optimization of Computer Models.* In N. Flournoy, W.F. Rosenberger & W.K. Wong, editors, New Developments and Applications in Experimental Design, volume 34, pages 11–25. Institute of Mathematical Statistics, 1998.

[Siah 04] E.S. Siah, M. Sasena & J.L. Volakis. *Fast Parameter Optimization of Large-Scale Electromagnetic Objects Using DIRECT with Kriging Metamodeling.* IEEE Trans. Microw. Theory Tech., vol. 52, no. 1, pages 276–285, January 2004.

[Sim 05] R. Sim & N. Roy. *Global A-Optimal Robot Exploration in SLAM.* In Proc. of the IEEE Int. Conf. on Robotics & Automation, 2005.

[Sim 06] R. Sim & J.J. Little. *Autonomous vision-based exploration and mapping using hybrid maps and Rao-Blackwellised particle filters.* In Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pages 2082–2089, 2006.

[Singh 05] S. Singh, N. Kantas, A. Doucet, B.N. Vo & R.J. Evans. *Simulation-Based Optimal Sensor Scheduling with Application to Observer Trajectory Planning.* In Proc. of the IEEE Conf. on Decision and Control and Eur. Control Conference, pages 7296– 7301, 2005.

[Smallwood 73] R.D. Smallwood & E.J. Sondik. *The Optimal Control of Partially Observable Markov Processes over a Finite Horizon.* Operations Research, vol. 21, pages 1071–1088, 1973.

[Smith 88] R. Smith, M. Self & P. Cheeseman. *A Stochastic Map For Uncertain Spatial Relationships.* In O. Faugeras & G. Giralt, editors, Robotics Research, The Fourth Int. Symp., pages 467–474. The MIT Press, 1988.

[Smola 05] A.J. Smola, S.V.N. Vishwanathan & T. Hofmann. *Kernel Methods for Missing Variables.* In Proc. of the Int. Conf. on AI and Statistics, pages 325–332, 2005.

[Spall 03] J.C. Spall. Introduction to stochastic search and optimization. Wiley InterScience, 2003.

[Stachniss 05a] C. Stachniss, G. Grisetti & W. Burgard. *Information Gain-based Exploration Using Rao-Blackwellized Particle Filters.* In Proc. of Robotics: Science and Systems, Cambridge, USA, June 2005.

[Stachniss 05b] C. Stachniss, G. Grisetti & Wolfram Burgard. *Recovering Particle Diversity in a Rao-Blackwellized Particle Filter for SLAM After Actively Closing Loops.* In Proc. of the IEEE Int. Conf. on Robotics & Automation, pages 667–672, Bercelona, Spain, 2005.

[Stachniss 06] C. Stachniss. *Exploration and Mapping with Mobile Robots.* PhD thesis, Institute of Computer Science, University of Freiburg, 2006.

[Stein 99] M.L. Stein. Interpolation of spatial data. Springer-Verlag, 1999.

[Storvik 02] G. Storvik. *Particle Filters for State-Space Models With the Presence of Unknown Static Parameters.*

IEEE Trans. Signal Process., vol. 50, no. 2, pages 281–289, 2002.

[Tadic 05] V.B. Tadic & A. Doucet. *Exponential Forgetting and Geometric Ergodicity for Optimal Filtering in General State-Space Models.* Stochastic Processes and Their Applications, vol. 115, pages 1408–1436, 2005.

[Tardós 02] J. D. Tardós, J. Neira, P. M. Newman & J. J. Leonard. *Robust Mapping and Localization in Indoor Environments using Sonar Data.* Int. J. Robotics Research, vol. 21, no. 4, pages 311–330, 2002.

[Taylor 96] R. Taylor & P. Probert. *Range Finding and Feature Extraction by Segmentation of Images for Mobile Robot Navigation.* In Proc. of the IEEE Int. Conf. on Robotics & Automation, 1996.

[Thrun 93] S. Thrun. *Exploration and Model Building in Mobile Robot Domains.* In E. Ruspini, editor, Proc. of the IEEE Int. Conf. on Neural Networks, pages 175–180, 1993.

[Thrun 01] S. Thrun. *A Probabilistic Online Mapping Algorithm for Teams of Mobile Robots.* Int. J. Robotics Research, vol. 20, no. 5, pages 335–363, 2001.

[Thrun 02] S. Thrun. *Robotic Mapping: A Survey.* In G. Lakemeyer & B. Nebel, editors, Exploring Artificial Intelligence in the New Millenium. Morgan Kaufmann, 2002.

[Thrun 04] S. Thrun, Y. Liu, D. Koller, A.Y. Ng, Z. Ghahramani & H. Durrant-Whyte. *Simultaneous Localization and Mapping With Sparse Extended Information Filters.* Int. J. Robotics Research, vol. 23, pages 693–716, 2004.

[Thrun 05a] S. Thrun, W. Burgard & D. Fox. Probabilistic robotics. The MIT Press, 2005.

[Thrun 05b] S. Thrun & M. Montemerlo. *The GraphSLAM Algorithm With Applications to Large-Scale Mapping of Urban Structures.* Int. J. Robotics Research, vol. 25, no. 5/6, pages 403–430, 2005.

[Tichavský 98] P. Tichavský, C. Muravchik & A. Nehorai. *Posterior Cramér-Rao Bounds for Discrete-Time Nonlinear Filtering.* IEEE Trans. Signal Process., vol. 46, no. 5, pages 1386–1396, 1998.

[Tremois 99] O. Tremois & J.P. LeCadre. *Optimal Observer Trajectory in Bearings-Only Tracking for Manoeuvring Sources.* IEE Proc. Radar, Sonar Navig., vol. 146, no. 1, pages 31–39, 1999.

[van der Merwe 01] R. van der Merwe & E. Wan. *The Square-Root Unscented Kalman Filter for State and Parameter-Estimation.* In Proc. of the Int. Conf. on Acoustics, Speech, and Signal Processing, Salt Lake City, Utah, May 2001.

[van der Merwe 04] R. van der Merwe. *Sigma-Point Kalman Filters for Probabilistic Inference in Dynamic State-Space Models.* PhD thesis, OGI School of Science & Engineering, Oregon Health & Science University, April 2004.

[Vazquez 08] E. Vazquez & J. Bect. *On the convergence of the expected improvement algorithm.* arXiv.org, vol. arXiv:0712.3744v2 [stat.CO], February 2008.

[Vidal-Calleja 06] T. Vidal-Calleja, A.D. Davison, J. Andrade-Cetto & D.W. Murray. *Active Control for Single Camera SLAM.* In Proc. of the IEEE Int. Conf. on Robotics & Automation, pages 1930–1936, 2006.

[Walter 05] M. Walter, R. Eustice & J. Leonard. *A Provably Consistent Method for Imposing Exact Sparsity in Feature-based SLAM Information Filters.* In Proc. of the Int. Symp. of Robotics Research, San Francisco, CA, USA, October 2005.

[Wand 95] M.P. Wand & M. Jones. Kernel smoothing. Chapman & Hall, 1995.

[Wang 04] H. Wang & D. Suter. *Robust Adaptative-Scale Parametric Model Estimation for Computer Vision.* IEEE Trans. Pattern Anal. Mach. Intell., vol. 26, no. 11, pages 1459–1474, November 2004.

[Williams 92] R.J. Williams. *Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning.* Machine Learning, vol. 8, no. 3, pages 229–256, 1992.

[Wurm 07] K.M. Wurm, C. Stachniss, G. Grisetti & W. Burgard. *Improved Simultaneous Localization and Mapping using a Dual Representation of the Environment.* In Proc. of the Eur. Conf. on Mobile Robots, Freiburg, Germany, 2007.