DEGREE PROJECT IN MATHEMATICS,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2017

Application of new particle-based solutions to the Simultaneous Localization and Mapping (SLAM) problem

XAVIER SVENSSON DEPRAETERE

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ENGINEERING SCIENCES

Application of new particle-based solutions to the Simultaneous Localization and Mapping (SLAM) problem

XAVIER SVENSSON DEPRAETERE

Degree Projects in Mathematical Statistics (30 ECTS credits)
Degree Programme in Applied and Computational Mathematics (120 credits)
KTH Royal Institute of Technology year 2017
Supervisors at KTH: Jimmy Olsson and Johan Westerborn
Examiner at KTH: Jimmy Olsson

TRITA-MAT-E 2017:52
ISRN-KTH/MAT/E--17/52--SE

Royal Institute of Technology
School of Engineering Sciences
KTH SCI
SE-100 44 Stockholm, Sweden
URL: www.kth.se/sci

Sammanfattning

I detta examensarbete utforskas nya lösningar till Simultaneous Localization and Mapping (SLAM) problemet baserat på partikelfilter- och partikelglättnings-metoder. I sin grund består SLAM problemet av två av varandra beroende uppgifter: kartläggning och spårning. Tre lösningsmetoder som använder olika glättnings-metoder appliceras för att lösa dessa uppgifter. Dessa glättningsmetoder är *fixed lag smoothing* (FLS), *forward-only forward-filtering backward-smoothing* (forward-only FFBSm) och *the particle-based, rapid incremental smoother* (PaRIS). I samband med dessa glättningstekniker används den väletablerade *Expectation-Maximization* (EM) algoritmen för att skapa maximum-likelihood skattningar av kartan. De tre lösningsmetoderna utvärderas sedan och jämförs i en simulerad miljö.

Abstract

In this thesis, we explore novel solutions to the Simultaneous Localization and Mapping (SLAM) problem based on particle filtering and smoothing methods. In essence, the SLAM problem constitutes of two interdependent tasks: map building and tracking. Three solution methods utilizing different smoothing techniques are explored. The smoothing methods used are *fixed lag smoothing* (FLS), *forward-only forward-filtering backward-smoothing* (forward-only FFBSm) and *the particle-based, rapid incremental smoother* (PaRIS). In conjunction with these smoothing techniques the well-established *Expectation-Maximization* (EM) algorithm is used to produce maximum-likelihood estimates of the map. The three solution method are then evaluated and compared in a simulated setting.

Acknowledgements

I am using this opportunity to thank my supervisors Jimmy Olsson and Johan Westerborn for their guidance and time spent introducing me to the fascinating subject of sequential Monte Carlo methods. I have enjoyed exploring its intricacies and applications throughout the writing of this thesis and beyond.

Contents

Chapter 1

Introduction

The Simultaneous Localization and Mapping (SLAM) problem originates from the field of robotics and the need of autonomous robots to be able to accurately orient themselves in an unknown environment. It is the problem of constructing a map of the surrounding area while simultaneously tracking the robots position on said map. Focus lies on the word "simultaneous" as interdependent localization and mapping can be viewed as a catch-22 problem since accurate estimates of the robots position requires a map of the environment and vice versa. By concurrently performing localization and map building it is possible to overcome this obstacle.

SLAM is commonly applied in situations where no, or little previous knowledge of the environment is available. Often it is not possible to engineer the environment (such as by placing beacons) to aid in the process. An example of an application where these restrictions are prevalent is that of indoor-mapping. However, SLAM has also seen applications in outdoor, aerial and sub sea environments [8].

At disposal is commonly a robot with on-board sensors. These sensors detects environmental features from which landmarks are later abstracted. In order to more accurately determine the robots state the control input dictating the robots movement is known. In simpler SLAM models the robot state consist of its position and bearing while the map consist solely of the positions of stationary landmarks. However, it is possible to include other state variables such as the robots velocity or to take dynamic landmarks into consideration.

Typically, a SLAM system can be divided into two modules, the front-end and the back-end [3]. The front-end is responsible for feature extraction and data association. More specifically, given sensor data the front-end must be able to extract relevant features, such as the range and bearing of landmarks in relation to the robot. It must also manage to identify new, and recognize old landmarks whilst correctly associating the extracted features to the observed landmarks.The recognition of old landmarks (also referred to as loop-closure) is a key component to decrease errors and produce a cohesive map. The information of which landmarks were observed and the corresponding features are handed to the back-end of the SLAM system. The responsibility of the back-end is to estimate the map and robot state given previous estimates and features with landmark correspondence produced by the front-end. The estimated map and robot state are usually handed back to the front-end in order to support the data association process. The focus of this thesis is the back-end of the SLAM system.

An important distinction to make is the that between the online SLAM problem and the offline, or full SLAM problem. In the online problem formulation any parameter estimates need to be updated as new data is gathered. Whereas in the offline problem formulation any parameter need only be estimated after all the input and output data has been collected. This thesis will explore solutions to the later, but have the online problem in mind as it is prominent in many real life applications.

1.1 Previous work

One of the earliest prominent solutions to the SLAM problem was introduced by Smith, Self and Cheeseman [20]. Their solution is often referred to as the EKF-SLAM method and made use of the extended Kalman filter (EKF) to incrementally estimate the landmark and robot positions. In short, the extended Kalman filter is an adaptation of the Kalman filter to nonlinear models. As the original Kalman filter [13] requires linear models the EKF utilizes Taylor expansions in order to linearize the nonlinear models. A Kalman filter is then applied to solve the resulting linear system. Furthermore any system noise is, by application of the Kalman filter, presumed to be Gaussian.This method

had a computational complexity of $\mathcal{O}(N^2)$, where N is the number of landmarks. Hence for environments for a large amount of landmarks the computation time proved to be an hindrance. Furthermore, the use of linearization and the underlying assumption that all included distributions are Gaussian could cause problems with regards to consistency [2].

These hurdles were later overcome with the introduction of the FastSLAM algorithm [16]. The FastSLAM algorithm introduced a Bayesian standpoint to the SLAM problem and could handle non-Gaussian distributions better than EKF-SLAM. Key to the FastSLAM algorithm was the observation that the landmark readings could be considered conditionally independent given the robots position. This meant that the problem could be divided into $N + 1$ smaller estimation problems. With this approach the robots trajectory and the maps were estimated in different fashion. The trajectory was estimated using a particle filter, which is Monte Carlo-based estimation method that can handle non-linear models as well as non-Gaussian noise terms. The map on the other hand was estimated analytically using N separate EKFs, one for each landmark. Still, consistency problems persisted. In the long run the FastSLAM method does not produce consistent estimates as the algorithm was shown to degenerate over time [1]. The computational complexity of FastSLAM was originally $\mathcal{O}(N^2)$, but later reduced to $\mathcal{O}(N \log N)$ with the introduction of FastSLAM 2.0.

Nowadays, the de-facto standard method for solving SLAM is based on viewing the problem as a factor graph [3], also referred to as Graph-Based SLAM. In short, a factor graph is a visual tool where every factor, such as the robots state or the landmark position, is represented as a node. These nodes are then connected by lines representing a function describing a relation between the connected factors. This can for instance be the observed distance from the robot to a landmark. With this perspective, all robot transitions and landmark observations are viewed as soft constraints. By resolving these constraints a map and trajectory estimation can be obtained. If there are many such constraints, this results in solving a very large least square problem. Using naive methods this is computationally heavy problem. However, by utilizing the sparsity of the constraint matrix the problem can in some cases be solved in linear time with respect to the number of robot transitions and landmarks [21]. This sparsity also allows for online solu-

tions such as the iSAM algorithm [12].

1.2 Objective

The aim of this paper is to explore new, and compare solutions to the back-end part of SLAM using particle-based methods (also refereed to as sequential Monte Carlo methods). The use of particle-based methods to solve the back-end problem is not a novel approach. However, in the light of advancements in particle based smoothing techniques with lower computational complexity we find it fruitful to explore these techniques in the SLAM setting.

1.3 Limitations

In this paper we will only take the problem of offline SLAM into consideration. However, we note that it is possible to put all the used smoothing algorithms in an online setting or use an online counterpart of the algorithm. Furthermore, this paper will only focus on the SLAM back-end. Hence we assume an ideal front-end module, i.e., perfect landmark detection and data association. For ease of mathematical formulation and implementation the number of landmarks is assumed to be known and fixed, which is not the case in most applications.

We also assume that the only unknown parameters are the landmark locations and the robot trajectory. Hence the variances related to landmark observation and the robots dynamics are considered known. Note that in a real life applications these parameters can often be calibrated beforehand.

Chapter 2

Background

This section aims to present some of the theoretical framework behind the methods used in this thesis.

2.1 Hidden Markov Models (HMM)

In this paper the SLAM problem is formalized using an Hidden Markov Model (HMM). A HMM consists of a Markov process with initial probability density function $p(\boldsymbol{X}_0)$ and transition probability density $p(\boldsymbol{X}_t|\boldsymbol{X}_{t-1} = \boldsymbol{x}_{t-1})$. The Markov process generates a state sequence $\boldsymbol{x}_{0:T}$ that cannot be directly observed and is therefore "hidden" from view. However, the state $\boldsymbol{X}_t$ can be indirectly observed through the corresponding observation variable $\boldsymbol{Z}_t$. The observation variables are conditionally independent given the corresponding state and are determined by the probability density function $p(\boldsymbol{Z}_t|\boldsymbol{X}_t = \boldsymbol{x}_t)$ known as the emission density function. Figure 2.1 presents an illustration of an HMM.

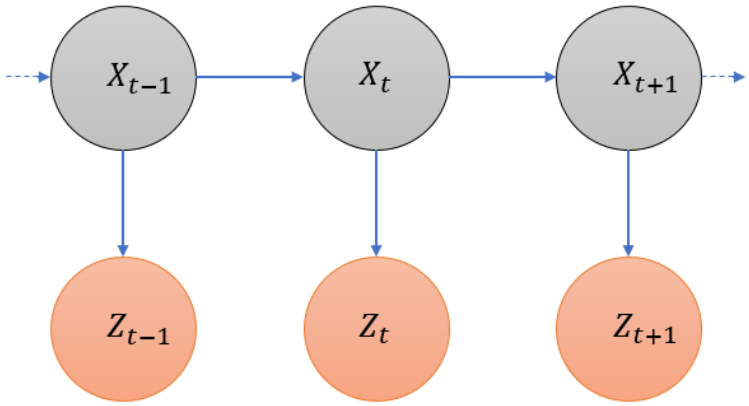


Figure 2.1: An illustration of a general HMM. The hidden states $\boldsymbol{X}_j$ are only indirectly observable through the random variable $\boldsymbol{Z}_j$. The blue arrows indicate interdependencies between the random variables.

In regards to HMMs one is often interested in estimating the sequence of hidden states $\boldsymbol{x}_{0:T}$ or a function of said states. Broadly speaking there are three types of problem formulations that are common in regards such estimation; the prediction problem, the filtering problem and the smoothing problem. The difference between the three is the amount of observational data that is used to support the estimate. In the prediction problem we look for the best estimate of a function $f(\boldsymbol{X}_t)$ based on past data points. I.e. we wish to form an estimator targeting $\mathbb{E}\left[f(\boldsymbol{X}_t)|\boldsymbol{Z}_{0:\tau} = \boldsymbol{z}_{0:\tau}\right]$ where $\tau < t$. Similarly in the filtering problem we look for an estimator using past and present data points targeting $\mathbb{E}\left[f(\boldsymbol{X}_t)|\boldsymbol{Z}_{0:t} = \boldsymbol{z}_{0:t}\right]$ and in the smoothing problem we make use of all available data to form an estimator targeting $\mathbb{E}\left[f(\boldsymbol{X}_t)|\boldsymbol{Z}_{0:T} = \boldsymbol{z}_{0:T}\right]$ where $T > t$. Between the three SLAM solving methods explored in this paper the main difference is the technique used to solve the smoothing problem. As such we will not compare different filtering techniques or solve the prediction problem. However, as all of the employed smoothing techniques require filter estimates we will need to solve the filtering problem as well.

A recurring theme in this paper will be to estimate smoothed expectations of *additive functionals*. That is, computing functions on the form

$$\hat{S}_T = \mathbb{E}\left[S_T(\boldsymbol{X}_{0:T})|\boldsymbol{Z}_{0:T} = \boldsymbol{z}_{0:T}\right] \tag{2.1}$$

where

$$S_T(\boldsymbol{X}_{0:T}) = \sum_{t=1}^{T} s_t(\boldsymbol{X}_{t-1}, \boldsymbol{X}_t) \tag{2.2}$$

and $\{s_t\}_{1 \leq t \leq T}$ is a sequence of measurable functions. Functions of this form are often encountered when estimating model parameters of an HMM using maximum likelihood methods. In this paper, this is especially relevant to the estimation of landmark locations.

2.2 Importance sampling (IS)

Given the observation sequence realization $\boldsymbol{Z}_{0:T} = \boldsymbol{z}_{0:T}$ we want to be able to compute

$$\mathbb{E}[f_t(\boldsymbol{X}_{0:t})|\boldsymbol{z}_{0:t}] = \int f_t(\boldsymbol{x}_{0:t})\ p(\boldsymbol{x}_{0:t}|\boldsymbol{z}_{0:t})\ d\boldsymbol{x}_{0:t} \tag{2.3}$$

for some integrable function $f_t(\boldsymbol{X}_{0:t})$, for instance the sequence of hidden states $\boldsymbol{X}_{0:t}$. Utilizing Bayes theorem and properties of the HMM the probability density function of the *joint smoothing distribution* can be expressed as

$$p(\boldsymbol{x}_{0:t}|\boldsymbol{z}_{0:t}) = \frac{p(\boldsymbol{x}_0)p(\boldsymbol{z}_0|\boldsymbol{x}_0)\prod_{k=1}^{t} p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})p(\boldsymbol{z}_k|\boldsymbol{x}_k)}{p(\boldsymbol{z}_{0:t})} \tag{2.4}$$

where

$$p(\boldsymbol{z}_{0:t}) = \int \cdots \int p(\boldsymbol{x}_0)p(\boldsymbol{z}_0|\boldsymbol{x}_0) \prod_{k=1}^{t} p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})p(\boldsymbol{z}_k|\boldsymbol{x}_k) d\boldsymbol{x}_0 \ldots d\boldsymbol{x}_t \tag{2.5}$$

is the *likelihood function*. Unfortunately the likelihood function is generally intractable making analytical solutions unviable. One method of dealing with this problem is the *importance sampling* method. Assume that we have a known *proposal distribution* $\pi(\boldsymbol{x}_{0:t})$ with a support that includes the support of $p(\boldsymbol{x}_{0:t}|\boldsymbol{z}_{0:t})$. Then we can form the following expression

$$\mathbb{E}[f_t(\boldsymbol{X}_{0:t})|\boldsymbol{Z}_{0:t}] = \frac{\int f_t(\boldsymbol{x}_{0:t})\omega(\boldsymbol{x}_{0:t})\pi(\boldsymbol{x}_{0:t})\ d\boldsymbol{x}_{0:t}}{\int \omega(\boldsymbol{x}_{0:t})\pi(\boldsymbol{x}_{0:t})\ d\boldsymbol{x}_{0:t}} \tag{2.6}$$

where

$$\omega_t(\boldsymbol{x}_{0:t}) = \frac{p(\boldsymbol{x}_0)p(\boldsymbol{z}_0|\boldsymbol{x}_0)\prod_{k=1}^{t} p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})p(\boldsymbol{z}_k|\boldsymbol{x}_k)}{\pi(\boldsymbol{x}_{0:t})} \tag{2.7}$$

is known as an *importance weight*. Hence if can draw M independent samples $\{\boldsymbol{x}_{0:t}^j\}_{1\leq j\leq M}$ (hereby referred to as particles) from our proposal distribution $\pi(\boldsymbol{x}_{0:t})$, we can form the Monte Carlo estimate

$$\hat{\mathbb{E}}[f_t(\boldsymbol{X}_{0:t})|\boldsymbol{Z}_{0:t}] = \sum_{j=1}^{M} f_t(\boldsymbol{x}_{0:t}^j)W_t(\boldsymbol{x}_{0:t}^j) \tag{2.8}$$

where

$$W_t(\boldsymbol{x}_{0:t}^j) = \frac{\omega_t(\boldsymbol{x}_{0:t}^j)}{\sum_{j=1}^{M}\omega_t(\boldsymbol{x}_{0:t}^j)} \tag{2.9}$$

is the *normalized importance weight*. Note that the denominator of the normalized importance weight forms a Monte Carlo estimate of the likelihood function $p(\boldsymbol{z}_{0:t})$.

However, the importance sampling algorithm as presented here is not suitable for recursion. This is because the estimation process requires all the observational data $\boldsymbol{z}_{0:t}$ for every time step t. If, for example, we are interested in estimating the underlying state sequence $\boldsymbol{x}_{0:t}$, then for each new observation $\boldsymbol{z}_{t+1}$ we must redraw the particle sample $\{\boldsymbol{x}_{0:t}^j\}_{1\leq j\leq M}$. Hence as t increases, the complexity increases.

2.3 Sequential importance sampling (SIS)

In order to allow the use of importance sampling method in a recursive setting we introduce the sequential importance sampling (SIS) method [15]. Key to the method is the specification of the proposal distribution as a sequence of conditional distributions such that

$$\pi(\boldsymbol{x}_{0:t}) = \pi(\boldsymbol{x}_0)\prod_{k=1}^{t}\pi(\boldsymbol{x}_k|\boldsymbol{x}_{0:k-1}). \tag{2.10}$$

Thus we are able to propagate the particles by sampling $\{\boldsymbol{x}_t^j\}_{1\leq j\leq M}$ from $\pi(\boldsymbol{x}_t|\boldsymbol{x}_{0:t-1}^j)$ and without needing to redraw $\{\boldsymbol{x}_{0:t-1}^j\}_{1\leq j\leq M}$. These

new samples can then be used to form filter estimates by again using equation (2.8). However to do so we also need to compute the associated importance weights. Using equations (2.7) and (2.10) the importance weights can be determined recursively using

$$\begin{aligned}\omega_t(\boldsymbol{x}_{0:t}) &= \underbrace{\frac{p(\boldsymbol{x}_0)p(\boldsymbol{z}_0|\boldsymbol{x}_0)\prod_{k=1}^{t-1}p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})p(\boldsymbol{z}_k|\boldsymbol{x}_k)}{\pi(\boldsymbol{x}_0)\prod_{k=1}^{t-1}\pi(\boldsymbol{x}_k|\boldsymbol{x}_{0:k-1})}}_{\omega_{t-1}(\boldsymbol{x}_{0:t-1})}\frac{p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})p(\boldsymbol{z}_t|\boldsymbol{x}_t)}{\pi(\boldsymbol{x}_t|\boldsymbol{x}_{0:t-1})} \\ &= \omega_{t-1}(\boldsymbol{x}_{0:t-1})\frac{p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})p(\boldsymbol{z}_t|\boldsymbol{x}_t)}{\pi(\boldsymbol{x}_t|\boldsymbol{x}_{0:t-1})}. \end{aligned} \tag{2.11}$$

Notably if we let the proposal distribution $\pi(\boldsymbol{x}_{0:t})$ be distributed according to $p(\boldsymbol{x}_{0:t})$, then the recursion formula simplifies to

$$\omega_t(\boldsymbol{x}_{0:t}) = \omega_{t-1}(\boldsymbol{x}_{0:t-1})p(\boldsymbol{z}_t|\boldsymbol{x}_t). \tag{2.12}$$

This formula lends itself open to an intuitive interpretation of the importance weight as a measure of the combined likelihood of an observation sequence outcome given a underlying state sequence. Unfortunately the SIS algorithm degenerates over time. The problem is that as t increases the variance of the weights increases [15]. After some time any estimate will be dominated by a few particles with relatively large weights. Hence only a few particle trajectories will mainly contributing to the estimate. Again this lends itself to an intuitive explanation. As the particles are only directed by the transition density there is no control mechanism guiding any stray particle trajectory to states with higher importance weights. Hence the simulated particle trajectories are likely to deviate more from the true trajectory as more steps in time are taken.

2.4 Sequential importance sampling with resampling (SISR)

In order to address the weight degeneracy problem prevalent in SIS a resampling step is employed, leading to the sequential importance

sampling with resampling (SISR) algorithm (also known as the bootstrap filter)[11]. The filtering procedure of the SISR algorithm is often described as being performed in two steps; the mutation step and the selection step. The mutation step is simply the SIS method described earlier in this paper. The selection step is a resampling procedure where the goal is to eliminate particles with low importance weight and duplicate the ones with a high importance weight. The rationale behind this being that by resampling the particles will not be able to deviate as far from more likely states. Therefore filter estimates will not be dominated by only a few particles and the weight degeneracy problem is solved. More specifically selection step is preformed using the following resampling procedure. Assume that we are given a particle set $\{\boldsymbol{x}_{0:t}^j\}$ and associated normalized importance weights $\{W_t^j\}_{1\leq j\leq M}$ (see equation (2.9)). We then sample with replacement a new particle set $\{\tilde{\boldsymbol{x}}_{0:t}^j\}_{1\leq j\leq M}$ from the original set $\{\boldsymbol{x}_{0:t}^j\}_{1\leq j\leq M}$. The sampling probability of a particle $\boldsymbol{x}_{0:t}^j$ is equal to the corresponding normalized importance weight W_t^j. Thus we end up with a new particle set $\{\tilde{\boldsymbol{x}}_{0:t}^j\}_{1\leq j\leq M}$. Lastly, these new particles are all given equal importance weights of $\omega_t(\tilde{\boldsymbol{x}}_{0:t}^j) = 1$. The new particle set is then used in the next mutation step and the old set is discarded.

The selection step may not necessarily be performed after every mutation step. One reason for not liberally performing resampling is that this limits the explored state space by the particles. One way to determine when to resample is to consider some form of threshold criteria depending on the importance weight variance.

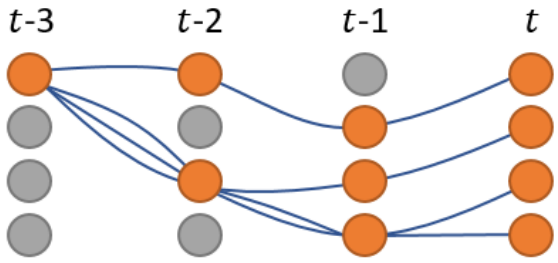


Figure 2.2: An illustration of the genealogical trace generated by SISR. The circles represents particles and the lines represents the particle history for the samples at time t. Note that there is less particle diversity further in the past and more diversity closer to the present time t. This is what as known as the path degeneracy problem

Any estimate using SISR can be formed in the same way as in SIS using equation (2.8). Interestingly, the selection step entails that the genealogical track, or trajectory, of the particles $\boldsymbol{x}_{0:t}^j$ are realizations of the joint smoothing distribution defined by $p(\boldsymbol{x}_{0:t}|\boldsymbol{z}_{0:t})$. However, estimates depending on past states become worse the further back we go. This is due to the collapse in the genealogical track of the particles (i.e. particles history) induced by the resampling mechanic. An illustration of this phenomena is presented in Figure 2.2. It is therefore ill-advised to simply use the genealogical particle history of current particles in order to approximate smoothed expectations, such as smoothed state expectaions $\mathbb{E}[\boldsymbol{X}_t|\boldsymbol{Z}_{0:T}]$. Thus SISR is most useful when determining filter estimates, i.e. estimates on the form $\mathbb{E}[f_t(\boldsymbol{X}_t)|\boldsymbol{Z}_{0:t}]$. Other methods are required in order to obtain more reliable smoothed estimates. The employed version of SISR in this paper is described in Algorithm 2 in Appendix A.

2.5 Fixed lag smoothing (FLS)

One method for acquiring smoothed estimations is the fixed lag smoothing (FLS) technique presented in [6] and further studied in [19]. FLS aims to work around the path degeneracy problem of SISR by retrieving approximate smoothed estimates before the path trajectories fully collapses. This technique relies on the so called forgetting properties of the conditional underlying Markov chain. I.e that the distributions of two Markov chains with the same transition density but with different initial distributions will, as time passes, approach each other [19]. FLS takes advantage of this forgetting property by making the approximation

$$\mathbb{E}[s_t(\boldsymbol{X}_{t-1}, \boldsymbol{X}_t)|\boldsymbol{Z}_{0:T}] \approx \mathbb{E}[s_t(\boldsymbol{X}_{t-1}, \boldsymbol{X}_t)|\boldsymbol{Z}_{0:h(\Delta_T)}] \tag{2.13}$$

where $h(\Delta_T) = \min(t + \Delta_T, T)$ and $\Delta_T \leq T - t$ is a lag parameter which dictates how many future data points we consider. Given particles $\{\boldsymbol{x}_{0:h(\Delta_T)}^j\}_{1\leq j\leq M}$ and associated importance weights $\{\omega_{h(\Delta_T)}^j\}_{1\leq j\leq M}$ generated from the SISR algorithm we can construct smoothed estimates of some function $f_t(\boldsymbol{X}_t)$ according to

$$\hat{\mathbb{E}}[f_t(\boldsymbol{X}_t)|\boldsymbol{Z}_{0:T}] = \frac{\sum_{j=1}^{M} \omega^j_{h(\Delta_T)} f_t[\boldsymbol{x}^j_{0:h(\Delta_T)}(t)]}{\sum_{j=1}^{M} \omega^j_{h(\Delta_T)}} \tag{2.14}$$

where $\boldsymbol{x}^j_{0:h(\Delta_T)}(t)$ is the state at time t for the particle trajectory $\boldsymbol{x}^j_{0:h(\Delta_T)}$. [1] Smoothed estimates of the of the sum of additive functionals as presented in equation (2.2) can be obtained using

$$\hat{S}_T = \mathbb{E}\left[\sum_{t=1}^{T} s_t(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t)|\boldsymbol{Z}_{0:T}\right] \approx \sum_{t=1}^{T}\sum_{j=1}^{M} \frac{\omega^j_{h(\Delta_T)} s_t[\boldsymbol{x}^j_{0:h(\Delta_T)}(t-1:t)]}{\sum_{j=1}^{M} \omega^j_{h(\Delta_T)}}. \tag{2.15}$$

Fixed lag smoothing has a computational complexity with respect to particles of $\mathcal{O}(M)$, but requires a recent history of all particles genealogical trace of size $\Delta_T \times M$ to be stored in memory. Choosing the lag Δ_T may not be trivial due to a bias-variance trade-off inherit in the method. A low value of the lag Δ_T will lead to a bias being present as the approximation in equation (2.13) becomes coarse. On the other hand, a too high value of the lag Δ_T might decrease the particle diversity used in the estimation due to the collapse of the path trajectories. Leading to not only higher estimation variance but also an increase in bias [19]. If there are many additive functionals to be computed, then it might be less cumbersome to simply calculate smoothed expectation of the underlying state variable $\boldsymbol{X}_t$ by utilizing equation (2.14). The smoothed state estimates can then be used to estimate the additive functionals directly.

[1] Note that trajectory sample $\boldsymbol{x}^j_{0:h(\Delta_T)}(t)$ may be different from the particle filter sample $\boldsymbol{x}^j_t$. The difference can be seen in Figure 2.2 where $\boldsymbol{x}^j_{0:t}(t-3)$ has the state value of the orange samples for all indices j. Whereas the particle filter samples $\boldsymbol{x}^j_{t-3}$ are all the state values (orange and grey) at time $t-3$.

2.6 Forward-filtering backward-smoothing (FFBSm)

As mentioned in Section 2.4, one can form smoothed estimates using SISR. However due to the path degeneracy problem these estimates are often poor. The forward-filtering backward-smoothing (FFBSm) algorithm seeks to remedy this by reevaluating the importance weights generated by SISR in a backwards pass of the data set.

Key to the FFBSm algorithm is the following decomposition of the smoothing distribution

$$p(\boldsymbol{x}_t|\boldsymbol{z}_{0:T}) = p(\boldsymbol{x}_t|\boldsymbol{z}_{0:t}) \int \frac{p(\boldsymbol{x}_{t+1}|\boldsymbol{z}_{0:T})p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t)}{p(\boldsymbol{x}_{t+1}|\boldsymbol{z}_{0:t})} d\boldsymbol{x}_{t+1}. \tag{2.16}$$

Note that $p(\boldsymbol{x}_t|\boldsymbol{z}_{0:t})$ is the filtering distribution, which can be accurately approximated using SISR. In order to present the algorithm in a more intuitive way we first present the first backward step, approximating $p(\boldsymbol{x}_{T-1}|\boldsymbol{z}_{0:T})$ in a similar way to [7]. Given a observation sequence realization $\boldsymbol{Z}_{0:T} = \boldsymbol{z}_{0:T}$ a forward pass of the data is made using SISR, storing the particle filter samples $\{\boldsymbol{x}_t^j\}_{1\leq j\leq M}$ and their associated normalized importance weights $\{W_t^j\}_{1\leq j\leq M}$ for every time step t. Using the decomposition presented in equation (2.16) we have that

$$p(\boldsymbol{x}_{T-1}|\boldsymbol{z}_{0:T}) = p(\boldsymbol{x}_{T-1}|\boldsymbol{z}_{0:T-1}) \int \frac{p(\boldsymbol{x}_T|\boldsymbol{z}_{0:T})p(\boldsymbol{x}_T|\boldsymbol{x}_{T-1})}{p(\boldsymbol{x}_T|\boldsymbol{z}_{0:T-1})} d\boldsymbol{x}_T. \tag{2.17}$$

By applying the Monte Carlo estimate described in equation (2.8) we can form the following approximation

$$\begin{aligned}\int \frac{p(\boldsymbol{x}_T|\boldsymbol{z}_{0:T})p(\boldsymbol{x}_T|\boldsymbol{x}_{T-1})}{p(\boldsymbol{x}_T|\boldsymbol{z}_{0:T-1})} d\boldsymbol{x}_T &= \int \frac{p(\boldsymbol{x}_T|\boldsymbol{x}_{T-1})}{p(\boldsymbol{x}_T|\boldsymbol{z}_{0:T-1})} p(\boldsymbol{x}_T|\boldsymbol{z}_{0:T}) d\boldsymbol{x}_T \\ &. \approx \sum_{j=1}^{M} W_T^j \frac{p(\boldsymbol{x}_T^j|\boldsymbol{x}_{T-1})}{p(\boldsymbol{x}_T^j|\boldsymbol{z}_{0:T-1})}\end{aligned} \tag{2.18}$$

Where, again using the Monte Carlo filter estimation, we have that can

approximate $p(\boldsymbol{x}_T^j|\boldsymbol{z}_{0:T-1})$ via

$$\begin{aligned} p(\boldsymbol{x}_T^j|\boldsymbol{z}_{0:T-1}) &= \int p(\boldsymbol{x}_T^j|\boldsymbol{x}_{T-1})p(\boldsymbol{x}_{T-1}|\boldsymbol{z}_{0:T-1})d\boldsymbol{x}_{T-1} \\ &\approx \sum_{l=1}^{M} W_{T-1}^l p(\boldsymbol{x}_T^j|\boldsymbol{x}_{T-1}^l). \end{aligned} \tag{2.19}$$

Using equations (2.17) - (2.19) we can form the following smoothed estimate

$$\hat{\mathbb{E}}[f_{T-1}(\boldsymbol{X}_{T-1})|\boldsymbol{Z}_{0:T}] = \sum_{i=1}^{M} W_{T-1}^i \left[\sum_{j=1}^{M} W_T^j \frac{p(\boldsymbol{x}_T^j|\boldsymbol{x}_{T-1}^i)}{\sum_{l=1}^{M} W_{T-1}^l p(\boldsymbol{x}_T^j|\boldsymbol{x}_{T-1}^l)} \right] f_{T-1}(\boldsymbol{x}_{T-1}^i) \tag{2.20}$$

where $f_{T-1}(\boldsymbol{X}_{T-1})$ is an integrable function. By defining the reevaluated importance weight at time $T-1$ as

$$W_{T-1|T}^i \coloneqq \sum_{j=1}^{M} W_T^j \frac{W_{T-1}^i p(\boldsymbol{x}_T^j|\boldsymbol{x}_{T-1}^i)}{\sum_{l=1}^{M} W_{T-1}^l p(\boldsymbol{x}_T^j|\boldsymbol{x}_{T-1}^l)} \tag{2.21}$$

we end up with the following shorter formulation of equation (2.20)

$$\hat{\mathbb{E}}[f_{T-1}(\boldsymbol{X}_{T-1})|\boldsymbol{Z}_{0:T}] = \sum_{i=1}^{M} W_{T-1|T}^i f_{T-1}(\boldsymbol{x}_{T-1}^i). \tag{2.22}$$

In general, any reevaluated importance weight can be computed using the following recursive formula

$$W_{t|T}^i \coloneqq \sum_{j=1}^{M} W_{t+1|T}^j \frac{W_t^i p(\boldsymbol{x}_{t+1}^j|\boldsymbol{x}_t^i)}{\sum_{l=1}^{M} W_t^l p(\boldsymbol{x}_{t+1}^j|\boldsymbol{x}_t^l)} \tag{2.23}$$

with $W_{T|T}^i \coloneqq W_T^i$. Using this recursive expression smoothed estimates of any integrable function $f_t(\boldsymbol{X}_t)$ can be retrieved using

$$\hat{\mathbb{E}}[f_t(\boldsymbol{X}_t)|\boldsymbol{Z}_{0:T}] = \sum_{i=1}^{M} W_{t|T}^i f_t(\boldsymbol{x}_t^i). \tag{2.24}$$

FFBSm is not an online algorithm as it requires all particle filter samples $\{\boldsymbol{x}_t^j\}_{1\leq j\leq M}$ and their associated normalized importance weights

$\{W_t^j\}_{1\leq j\leq M}$ for every time step t to be computed before the backwards pass. FFBSm also suffers from high computational complexity with respect to the number of particles M of $\mathcal{O}(M^2)$.

2.7 Forward-only forward-filtering backward-smoothing (forward-only FFBSm)

Since application areas, such as SLAM, often require smoothed estimations to be performed online we will present the forward-only FFBSm algorithm proposed in [17]. Using this algorithm it is possible to recursively compute smoothed expectations of additive functionals (see equation 2.1) in a single forward pass of the data. However, unless we specifically target every single underlying state $\boldsymbol{X}_t$ (which is impractical), we sacrifice information of the complete smoothed state sequence $\boldsymbol{X}_{0:T}|\boldsymbol{Z}_{0:T}$.

We define the following auxiliary function $\tau_{t+1}(\boldsymbol{x}_{t+1}) \coloneqq \mathbb{E}[S_t(\boldsymbol{X}_{0:t})|\boldsymbol{X}_t = \boldsymbol{x}_t, \boldsymbol{z}_{0:t}]$ and note that this function can be updated recursively using the tower rule of conditional expectations by

$$\begin{aligned}\tau_{t+1}(\boldsymbol{x}_{t+1}) =&\mathbb{E}[S_{t+1}(\boldsymbol{X}_{0:t+1})|\boldsymbol{X}_{t+1} = \boldsymbol{x}_{t+1}, \boldsymbol{z}_{0:t+1}]\\ =&\mathbb{E}[s_{t+1}(\boldsymbol{X}_t, \boldsymbol{X}_{t+1}) + \mathbb{E}[S_t(\boldsymbol{X}_{0:t})|\boldsymbol{X}_t = \boldsymbol{x}_t, \boldsymbol{z}_{0:t}]|\boldsymbol{X}_{t+1} = \boldsymbol{x}_{t+1}, \boldsymbol{z}_{0:t+1}]\\ =&\mathbb{E}[s_{t+1}(\boldsymbol{X}_t, \boldsymbol{X}_{t+1}) + \tau_t(\boldsymbol{x}_t)|\boldsymbol{X}_{t+1} = \boldsymbol{x}_{t+1}, \boldsymbol{z}_{0:t+1}]\\ =&\int [\tau_t(\boldsymbol{x}_t) + s_t(\boldsymbol{x}_t, \boldsymbol{x}_{t+1})]\, p(\boldsymbol{x}_t|\boldsymbol{z}_{0:t}, \boldsymbol{x}_{t+1}) d\boldsymbol{x}_t. \end{aligned} \tag{2.25}$$

One can then form a smoothed estimate of an additive functional as

$$\begin{aligned}\hat{S}_t &= \mathbb{E}[\mathbb{E}[S_t(\boldsymbol{X}_{0:t})|\boldsymbol{X}_t = \boldsymbol{x}_t, \boldsymbol{z}_{0:t}]|\boldsymbol{z}_{0:t}]\\ &= \mathbb{E}[\tau_t(\boldsymbol{x}_t)|\boldsymbol{z}_{0:t}] = \int \tau_t(\boldsymbol{x}_t) p(\boldsymbol{x}_t|\boldsymbol{z}_{0:t}) d\boldsymbol{x}_t \end{aligned} \tag{2.26}$$

As the function $\tau_t(\boldsymbol{x}_t)$ can be computed recursively using equation (2.25) we now have an online method for determining $\hat{S}_t$. Using Bayes theorem we have that

$$p(\boldsymbol{x}_t|\boldsymbol{z}_{0:t}, \boldsymbol{x}_{t+1}) = \frac{p(\boldsymbol{x}_t, \boldsymbol{x}_{t+1}|\boldsymbol{z}_{0:t})}{p(\boldsymbol{x}_{t+1}|\boldsymbol{z}_{0:t})} \tag{2.27}$$

where again by Bayes and conditional independence we have that

$$p(\boldsymbol{x}_t, \boldsymbol{x}_{t+1}|\boldsymbol{z}_{0:t}) = p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t)p(\boldsymbol{x}_t|\boldsymbol{z}_{0:t}) \tag{2.28}$$

and

$$\begin{aligned} p(\boldsymbol{x}_{t+1}|\boldsymbol{z}_{0:t}) &= \int p(\boldsymbol{x}_t, \boldsymbol{x}_{t+1}|\boldsymbol{z}_{0:t})d\boldsymbol{x}_{t+1} \\ &= \int p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t)p(\boldsymbol{x}_t|\boldsymbol{z}_{0:t})d\boldsymbol{x}_{t+1}. \end{aligned} \tag{2.29}$$

Using equation (2.28) and (2.29) we can rewrite equation (2.25) as

$$\tau_{t+1}(\boldsymbol{x}_{t+1}) = \int \left[\tau_t(\boldsymbol{x}_t) + s_t(\boldsymbol{x}_t, \boldsymbol{x}_{t+1})\right] \frac{p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t)p(\boldsymbol{x}_t|\boldsymbol{z}_{0:t})}{\int p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t)p(\boldsymbol{x}_t|\boldsymbol{z}_{0:t})d\boldsymbol{x}_{t+1}} d\boldsymbol{x}_t \tag{2.30}$$

Assuming that we have particle samples $\{\boldsymbol{x}_t^j, \boldsymbol{x}_{t+1}^j\}_{1\leq j\leq M}$ with associated normalized importance weights $\{W_t^j, W_{t+1}^j\}_{1\leq j\leq M}$ targeting the filter distribution and previous estimates $\{\hat{\tau}_t(\boldsymbol{x}_t^j)\}_{1\leq j\leq M}$ of $\{\tau_t(\boldsymbol{x}_t^j)\}_{1\leq j\leq M}$. Then we can form smoothed Monte Carlo estimates of additive functionals using the relationships expressed in equation (2.26) and (2.30) via

$$\hat{S}_t \approx \sum_{i=1}^{M} W_{t+1}^i \hat{\tau}_{t+1}(\boldsymbol{x}_{t+1}^i) \tag{2.31}$$

where

$$\hat{\tau}_{t+1}(\boldsymbol{x}_{t+1}^i) = \frac{\sum_{j=1}^{M} W_t^j p(\boldsymbol{x}_{t+1}^i|\boldsymbol{x}_t^j)\left[\hat{\tau}_t(\boldsymbol{x}_t^j) + s_{t+1}(\boldsymbol{x}_t^j, \boldsymbol{x}_{t+1}^i)\right]}{\sum_{l=1}^{M} W_t^l p(\boldsymbol{x}_{t+1}^i|\boldsymbol{x}_t^l)}. \tag{2.32}$$

This recursive formula is initialized by setting $\{\tau_0(\boldsymbol{x}_0^j)\}_{1\leq j\leq M} = 0$. The computational complexity of this algorithm is its native form is $\mathcal{O}(M^2)$. However according to [17], it is possible to reduce this to $\mathcal{O}(M \log M)$ using techniques discussed in [14]. The forward-only FFBSm algorithm used in this paper is given in Algorithm 4 in Appendix A.

2.8 Forward-filtering backward-simulation (FFBSi)

Similarly to the FFBSm algorithm presented in Section 2.6 the forward-filtering backward-simulation (FFBSi) algorithm [10] relies on first performing a forward filtering pass of the data and then a smoothing backwards pass. However, rather than reevaluating the importance weights the smoothing in the FFBSi algorithm is performed by reevaluating the particles. This is done by simulating path trajectories from the time-reversed discrete Markov chain with the $M \times M$ transition matrix $\{\Lambda_t^M(j,i)\}_{1\leq i\leq M, 1\leq j\leq M}$ given by

$$\Lambda_t^M(j,i) \coloneqq \frac{W_{t-1}^j p(\boldsymbol{x}_t^i|\boldsymbol{x}_{t-1}^j)}{\sum_{l=1}^M W_{t-1}^l p(\boldsymbol{x}_t^i|\boldsymbol{x}_{t-1}^l)}. \tag{2.33}$$

The Markov chain starts by drawing a particle index J_T such that $p(J_T = i) = W_T^i$. It proceeds by drawing J_t according to $p(J_t = j|J_{t+1} = i) = \Lambda_t^M(j,i)$ for $t \leq T-1$. Hence, the joint probability density function for the index sequence $J_{0:T} \coloneqq \{J_0, J_1, \ldots, J_T\}$ becomes

$$p(J_{0:T} = j_{0:T}) = W_T^{j_T} \prod_{t=0}^{T-1} \Lambda_t^M(j_t, j_{t+1}). \tag{2.34}$$

The particle path $\{\boldsymbol{x}_0^{j_0}, \boldsymbol{x}_1^{j_1}, \ldots, \boldsymbol{x}_T^{j_T}\}$ is then a realization of the joint distribution defined by $p(\boldsymbol{x}_{0:t}|\boldsymbol{z}_{0:t})$. Therefore, by simulating $\hat{M}$ such samples producing the index sequences $\{j_0^\eta, j_1^\eta, \ldots, j_T^\eta\}_{1\leq\eta\leq\hat{M}}$ we can form the following smoothed estimate for additive functionals

$$\hat{S}_T = \hat{M}^{-1} \sum_{\eta=1}^{\hat{M}} S_T(\boldsymbol{x}_0^{j_0^\eta}, \boldsymbol{x}_1^{j_1^\eta}, \ldots, \boldsymbol{x}_T^{j_T^\eta}). \tag{2.35}$$

FFBSi is not an online method as we require a complete sequence of particles and importance weights generated by the forward filtering process. The computational complexity of this method with respect to the particle number is $\mathcal{O}(M\hat{M})$. In the case that $\hat{M} = M$ the FFBSi algorithm will therefore have quadratic complexity.

2.8.1 Fast version of forward-filtering backward-simulation (fast FFBSi)

One mayor hindrance to lowering the complexity is the need for calculating the normalizing constant $\sum_{l=1}^{M} W_{t-1}^{l} p(\boldsymbol{x}_t^i|\boldsymbol{x}_{t-1}^l)$ present in equation (2.33) for every index i and time step. Following [5], this computation can be avoided by utilizing an accept-reject-based approach which yields a lowered computational complexity of, in best case, $\mathcal{O}(M)$.

In order to sample from the time-reversed transition matrix $\Lambda_t^M(j,i)$ without computing the normalizing constant, the accept-reject method described in Section 2.11 is utilized. This accept-reject-based approach relies on the assumption that there exists a number $\epsilon_+ \in \mathbf{R}_+$ such that for all possibles values of $(\boldsymbol{x}_t, \boldsymbol{x}_{t-1})$ we have that $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) \leq \epsilon_+$. Using similar notation as in Section 2.11, we draw a candidate proposal J_t^* from the proposal distribution defined by $p(J_t^* = j) = W_t^j$. This proposal is accepted with probability $p(\boldsymbol{x}_{t+1}^{J_{t+1}}, \boldsymbol{x}_t^{J_t^*})/\epsilon_+$. In the context of Section 2.11, this follows from letting $g(j) = W_t^j$ define our proposal distribution and $\pi(j) = W_t^j p(\boldsymbol{x}_{t+1}^{J_{t+1}}, \boldsymbol{x}_t^j)$ define our unnormalized target distribution

2.9 The particle based, rapid incremental smoother (PaRIS)

The PaRIS algorithm is an online smoothing algorithm based on the same decomposition and recursive formula as forward-only FFBSm. However, compared to the forward-only FFBSm algorithm, the PaRIS algorithm has lower computational complexity of $\mathcal{O}(M)$. This is achieved by implementing an accept-reject-based sampling method similar to that of the fast version of the FFBSi algorithm. Instead of computing the probability

$$\Lambda_t^M(j,i) \coloneqq \frac{W_{t-1}^j p(\boldsymbol{x}_t^i|\boldsymbol{x}_{t-1}^j)}{\sum_{l=1}^{M} W_{t-1}^l p(\boldsymbol{x}_t^i|\boldsymbol{x}_{t-1}^l)} \tag{2.36}$$

present in equation (2.32) directly we (given a previous estimate $\{\hat{\tau}_t(\boldsymbol{x}_t^j)\}_{1\leq j\leq M}$ of $\{\tau_t(\boldsymbol{x}_t^j)\}_{1\leq j\leq M}$) form the estimate

$$\hat{\tau}_{t+1}(\boldsymbol{x}_{t+1}^i) = \tilde{M}^{-1}\sum_{\eta=1}^{\tilde{M}}\left(\hat{\tau}_t(\boldsymbol{x}_t^{K_{t+1}^{i,\eta}}) + s_{t+1}(\boldsymbol{x}_t^{K_{t+1}^{i,\eta}}, \boldsymbol{x}_{t+1}^i)\right) \tag{2.37}$$

where $\{K_{t+1}^{i,\eta}\}_{1\leq i\leq M, 1\leq\eta\leq\tilde{M}}$ are particle indices drawn with a probability

$$p(K_{t+1}^{i,\eta} = j) = \Lambda_{t+1}^M(j,i). \tag{2.38}$$

This sampling of particle indices is done by using the accept-reject-based sampling method described in Section 2.11 in a similar manner as for fast FFBSi. In equation (2.37) $\tilde{M}$ is a design parameter determining the amount of backward samples used in the estimation. It is advised by the authors to let $\tilde{M} \geq 2$ in order to keep numerical stability (see page 13 in [18] for discussion as to why).

The accept-reject sampling algorithm in PaRIS also includes a threshold mechanic on the number of rejected samples. This is implemented in order to ensure more consistent computation times. This threshold is suggested to be set to $\sqrt{M}$, which is a rule of thumb value by the authors (page 22, [18]). Pseudocode for the accept-reject sampling algorithm and the PaRIS algorithm used in this paper is given in algorithm 5 and 6 respectively in appendix A.

2.10 Expectation-maximization (EM)

Expectation-maximization (EM) is an iterative method for finding maximum likelihood estimates of model parameters θ. It is divided into two steps, the expectation step (E-step) and the maximization step (M-step). In the E-step the *intermediate quantity* defined as

$$Q(\theta|\theta^i) \coloneqq \mathbb{E}_{\theta^i}[\log(L(\theta|\boldsymbol{X}_{0:T}, \boldsymbol{Z}_{0:T}))|\boldsymbol{Z}_{0:T}] \tag{2.39}$$

is calculated, where $\mathbb{E}_{\theta^i}$ specifies the expectation given a previous model parameter estimate θ^i and $L(\theta|\boldsymbol{X}_{0:T}, \boldsymbol{Z}_{0:T})$ is the *complete data likelihood function*. Note that θ can be a vector valued parameter, as is the case later in this paper. For an HMM model, the complete data likelihood function is given by

$$L(\theta|\boldsymbol{X}_{0:T}, \boldsymbol{Z}_{0:T}) = p_\theta(\boldsymbol{X}_0)p_\theta(\boldsymbol{Z}_0|\boldsymbol{X}_0)\prod_{t=1}^{T} p_\theta(\boldsymbol{X}_t|\boldsymbol{X}_{t-1})p_\theta(\boldsymbol{Z}_t|\boldsymbol{X}_t) \quad (2.40)$$

where $p_\theta(\cdot)$ denotes a probability density function under a model parameter θ. In the M-step the parameter value θ^{i+1} which maximizes Q given a previous estimate θ^i is calculated. In other words, one finds θ^{i+1} such that

$$\theta^{i+1} = \arg\max_\theta Q(\theta|\theta^i). \quad (2.41)$$

Of specially interest is the case that the complete data likelihood $L(\theta|\boldsymbol{X}_{0:T}, \boldsymbol{Z}_{0:T})$ is part of the exponential family of distributions. This means that the complete data likelihood function can be written as

$$L(\theta|\boldsymbol{X}_{0:T}, \boldsymbol{Z}_{0:T}) = h(\boldsymbol{x}_{0:T}) \exp(\langle\phi(\theta), S(\boldsymbol{x}_{0:T})\rangle - c(\theta)) \quad (2.42)$$

where $\langle\cdot,\cdot\rangle$ denotes the scalar product, $\phi(\theta)$ and $c(\theta)$ are known functions and $S(\boldsymbol{x}_{0:T})$ is a *sufficient statistic*. $S(\boldsymbol{x}_{0:T})$ being a sufficient statistic means that there is no other function of the data set $\boldsymbol{x}_{0:T}$ that provides additional information about the parameter θ. Notably, $S(\boldsymbol{x}_{0:T})$ can be of the form of an additive functional. Due to the structure of equation (2.42) the maximization performed in the M-step only requires that one maximizes

$$\langle\phi(\theta), S(\boldsymbol{x}_{0:T})\rangle - c(\theta) \quad (2.43)$$

with respect to θ. Conveniently, one can often find a closed form solution to the maximization of equation (2.43). If such a closed form solution does not exist or is not easily obtainable, then one may use numerical methods to perform the M-step or apply Taylor series approximations to force a closed solution. The E- and M-step are then repeated until convergence.

The version of the EM algorithm described in this section is sometimes referred to as the *batch*, or offline EM algorithm. This is because one needs to process all observational data $\boldsymbol{z}_{0:T}$ before each update of any parameters. Although not applied in this paper it is noted here that there is also an online version of the EM algorithm proposed in [4]. The batch EM algorithm used in the context of this paper is presented with pseudocode in Algorithm 7 in Appendix A.

2.11 Accept-reject sampling

Suppose that we want to sample from a distribution with probability density function $p(y)$ given by

$$p(y) = \frac{\pi(y)}{\int \pi(y)dy}. \tag{2.44}$$

However, as the integral may be hard or computationally expensive to determine, we want to circumvent a direct computation of the denominator. The accept-reject algorithms does so in the in the following way. Assume that we can sample from a distribution with proposal density $g(y)$ and let there be a finite constant ϵ_+ such that $0 \leq \pi(y) \leq \epsilon_+ g(y)$ for all values of y. Then we can generate samples from $p(y)$ using the following scheme.

1. Sample a proposal Y with density proportional to $g(y)$.
2. Sample a threshold value U from a uniform distribution stretching between 0 and 1.
3. If
$$U \leq \frac{\pi(Y)}{\epsilon_+ g(Y)} \tag{2.45}$$
then we accept the proposal, else one repeats the process until acceptance.

The accepted proposals will then be distributed according to $p(y)$ [9]. We note that it is possible to preform this sampling method by choosing ϵ_+ such that $\pi(y) < \epsilon_+ g(y)$. However, the closer the function $\epsilon_+ g(y)$ is to $\pi(y)$, the higher the probability of acceptance becomes and the faster a sample can be drawn. It is therefore advantageous to let ϵ_+ be as small as possible whilst still fulfilling the necessary condition.

Chapter 3

Methodology

This section seeks to present how the solution methods were structured and the necessary calculations made in order to utilizes them.

3.1 Definitions

We defined the following random variables and parameters.

- The state-space $\mathcal{X} \coloneqq \{(a, b, \psi) \in \mathbf{R}^3 | \psi \in [-\pi, \pi)\}$.
- The observable space $\mathcal{Z} \coloneqq \{(a, \psi) \in \mathbf{R}^2 | \psi \in [-\pi, \pi)\}$
- The set of random variables $\boldsymbol{X}_{0:T} \coloneqq \{\boldsymbol{X}_0, \boldsymbol{X}_1, \ldots, \boldsymbol{X}_T\}$ that represented the hidden states. For $t \in \{0, 1, \ldots, T\}$ we had that

$$\boldsymbol{X}_t = \begin{bmatrix} X_{t,1} \\ X_{t,2} \\ X_{t,3} \end{bmatrix} \in \mathcal{X} \tag{3.1}$$

  where $(X_{t,1}, X_{t,2})$ represented the robots position in the real plane and $X_{t,3} \in [-\pi, \pi)$ its orientation in in said plane.

- The set of landmarks $\boldsymbol{m} = \{\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}_N\}$. For $i \in \{0, 1, \ldots, N\}$ we had that

$$\boldsymbol{m}_i = \begin{bmatrix} m_{i,1} \\ m_{i,2} \end{bmatrix} \in \mathbf{R}^2 \tag{3.2}$$

  where $(m_{i,1}, m_{i,2})$ represented the i'th landmarks position.

- The sets of indexes of observed landmarks $\{\mathcal{A}_t\}_{0\leq t\leq T}$ such that if the landmark was observed at time t, then $i \in \mathcal{A}_t$ else $i \notin \mathcal{A}_t$.
- The set of random variables $\boldsymbol{Z}_{0:T} = \{\boldsymbol{Z}_{0,A_0}, \boldsymbol{Z}_{1,A_1}, \ldots, \boldsymbol{Z}_{T,\mathcal{A}_t}\}$ that represented the landmark observations such that $\boldsymbol{Z}_{t,\mathcal{A}_t} = (\boldsymbol{Z}_{t,i})_{i\in\mathcal{A}_t}$ where $\boldsymbol{Z}_{t,i}$ was an individual landmark observation at time t. In turn, we had that
$$\boldsymbol{Z}_{t,i} = \begin{bmatrix} Z_{t,i,1} \\ Z_{t,i,2} \end{bmatrix} \in \mathcal{Z} \tag{3.3}$$
where $Z_{t,i,1}$ was the observed distance to the landmark $\boldsymbol{m}_i$ and $Z_{t,i,2}$ was the observed relative angle between the robots bearing and the landmark $\boldsymbol{m}_i$.
- The set of control inputs $\boldsymbol{u}_{0:T} = \{\boldsymbol{u}_0, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_T\}$. For $t \in \{0, 1, \ldots T\}$ we had
$$\boldsymbol{u}_t = \begin{bmatrix} v_t\delta \\ \alpha_t \end{bmatrix} \tag{3.4}$$
where $v_t \in R$ was the travel velocity at time t, $\delta \in \{a \in \mathbf{R} | a > 0\}$ was the step size and $\alpha_t \in [-\pi, \pi)$ represented the desired change in orientation at time t. For clarification, the step size δ was a measure of time distinct from the time index t. Furthermore, $\boldsymbol{u}_{0:T}$ was considered a known constant.

3.2 Procedure

We formalized the SLAM problem using an HMM where the robots state sequence was modelled as a Markov chain $\{\boldsymbol{X}_t\}_{0\leq t\leq T}$ and the sequence of landmark observations were considered a realization of the observation variables $\boldsymbol{Z}_{0:T}$. As such, the initial distribution $p(\boldsymbol{x}_0)$, the transition probability $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{u}_t)$, the emission distribution defined by $p(\boldsymbol{z}_t|\boldsymbol{x}_t, \boldsymbol{m})$ and the sequence of control inputs $\boldsymbol{u}_{0:T}$ were all considered known. The map $\boldsymbol{m}$, however, was considered an unknown model parameter.

To receive test data we used the HMM to simulate an observation and a control sequence given a known map and trajectory. The aim was then to use particle-based methods to estimate the hidden state sequence $\{\boldsymbol{X}_t\}_{0\leq t\leq T}$ and the map $\boldsymbol{m}$ given only the observation sequence $\{\boldsymbol{z}_t\}_{0\leq t\leq T}$ and the control sequence $\{\boldsymbol{u}\}_{0\leq t\leq T}$. To gather filter estimates

of the state sequence $\boldsymbol{X}_{0:T}$ we used the *sequential importance sampling with resampling* SISR algorithm. To produce the map estimates we used the well-established offline EM algorithm. Hence we required smoothed estimates of sufficient statistics with respect to the map $\boldsymbol{m}$. These estimates were determined using three different smoothing algorithms. These were: the *fixed lag smoothing* (FLS) [19] algorithm, the *forward-only forward-filtering backward-smoothing* (forward-only FFBSm) [17] algorithm and the *particle-based, rapid incremental smoother* (PaRIS) algorithm [18]. Evaluations and comparison were then made regarding the effectiveness of these three approaches given a time budget.

A more in-depth description of the data generation process and the simulation results is presented in Section 4. The overall solution method is described in pseudocode in Algorithm 1 below and pseudocode for all used algorithms is presented in Appendix A.

Algorithm 1 Particle-based SLAM procedure

1: Assume that an observation sequence $\boldsymbol{z}_{0:T}$, a control sequence $\boldsymbol{u}_{0:T}$ and an initial map estimate $\boldsymbol{m}^0$ are available.
2: **for** $k = 1 \rightarrow K$
3: **for** $t = 0 \rightarrow T$
4: Given an observation $\boldsymbol{z}_t$ and a map estimate $\boldsymbol{m}^{k-1}$ generate a particle set $\{W_t^j, \boldsymbol{x}_t^j\}_{1 \leq j \leq M}$ and a filter estimate of the state $\hat{\boldsymbol{x}}_t$ using the SISR algorithm.
5: Given a particle set $\{W_t^j, \boldsymbol{x}_t^j\}_{1 \leq j \leq M}$ compute smoothed expectations of $S_t(\boldsymbol{X}_{0:t})$ using either FLS, forward-only FFBSm or PaRIS.
6: **end for**
7: Given a smoothed expectation of $S_T(\boldsymbol{X}_{0:T})$ produce a new map estimate $\boldsymbol{m}^{k+1}$ using the EM algorithm
8: **end for**
9: **return** the latest map $\boldsymbol{m}^K$ and trajectory $\hat{\boldsymbol{x}}_{0:T}$ estimates

3.3 Transition and observation model

The state sequence was initialized by setting $\boldsymbol{X}_0 = \vec{0}$ and state transition model was defined according to

$$\begin{bmatrix} X_{t,1} \\ X_{t,2} \\ X_{t,3} \end{bmatrix} = \begin{bmatrix} X_{t-1,1} \\ X_{t-1,2} \\ X_{t-1,3} \end{bmatrix} + \begin{bmatrix} v_t\delta\cos(X_{t-1,3}) \\ v_t\delta\sin(X_{t-1,3}) \\ \alpha_t \end{bmatrix} + \begin{bmatrix} \epsilon_{d_{t,1}} \\ \epsilon_{d_{t,2}} \\ \epsilon_{\alpha_t} \end{bmatrix} \tag{3.5}$$

where $\epsilon_{d_{t,1}}, \epsilon_{d_{t,2}}, \epsilon_{\alpha_t}$ are normally distributed noise variables with standard deviation $\sigma_{d_{t,1}}$, $\sigma_{d_{t,2}}$ and σ_{α_t} respectively.

The observation model was given by

$$\boldsymbol{Z}_{t,i} = \begin{bmatrix} Z_{t,i,1} \\ Z_{t,i,2} \end{bmatrix} = \begin{bmatrix} \rho^*(\boldsymbol{X}_t, \boldsymbol{m}_i) + \omega_\rho \\ \phi^*(\boldsymbol{X}_t, \boldsymbol{m}_i) + \omega_\phi \end{bmatrix} = \begin{bmatrix} \sqrt{(X_{t,1} - m_{i,1})^2 + (X_{t,2} - m_{i,2})^2} + \omega_\rho \\ \arctan\left(\frac{X_{t,2} - m_{i,2}}{X_{t,1} - m_{i,1}}\right) - X_{t,3} + \omega_\phi \end{bmatrix} \tag{3.6}$$

where ω_ρ and ω_ϕ are normally distributed noise variables with standard deviation σ_ρ and σ_ϕ respectively.

3.4 Analytical expression of the transition density

In order to utilize the forward-only FFBS algorithm we required the analytical expression of the transition density $p(\boldsymbol{X}_t|\boldsymbol{X}_{t-1} = \boldsymbol{x}_{t-1}, \boldsymbol{u}_t)$. We had that

$$p(\boldsymbol{X}_t|\boldsymbol{X}_{t-1}, \boldsymbol{u}_t) = p(X_{t,1}, X_{t,2}, X_{t,3}|X_{t-1,1}, X_{t-1,2}, X_{t-1,3}, \boldsymbol{u}_t). \tag{3.7}$$

Given the transition model presented in equation (3.5) we had due to conditional independence that

$$\begin{aligned} &p(X_{t,1}, X_{t,2}, X_{t,3}|X_{t-1,1}, X_{t-1,2}, X_{t-1,3}, \boldsymbol{u}_t) = \\ &\quad p(X_{t,1}|X_{t-1,1}, X_{t-1,3}, \boldsymbol{u}_t)p(X_{t,2}|X_{t-1,2}, X_{t-1,3}, \boldsymbol{u}_t)p(X_{t,3}|X_{t-1,3}, \boldsymbol{u}_t). \end{aligned} \tag{3.8}$$

It was clear from the model that

$$p(X_{t,1}|X_{t-1,1}, X_{t-1,3}, \boldsymbol{u}_t) = \frac{1}{\sqrt{2\pi\sigma^2_{d_{t,1}}}} \exp\left(-\frac{1}{2} \frac{(X_{t,1} - (X_{t-1,1} + v_t\delta\cos(X_{t-1,3})))^2}{\sigma^2_{d_{t,1}}} \right) \tag{3.9}$$

and

$$p(X_{t,2}|X_{t-1,2}, X_{t-1,3}, \boldsymbol{u}_t) = \frac{1}{\sqrt{2\pi\sigma^2_{d_{t,2}}}} \exp\left(-\frac{1}{2} \frac{(X_{t,2} - (X_{t-1,2} + v_t\delta\sin(X_{t-1,3})))^2}{\sigma^2_{d_{t,2}}} \right). \tag{3.10}$$

However as $\{X_{t,3}\}_{0\leq t\leq T} \in [-\pi, \pi)$ we had that $X_{t,3}|X_{t-1,3}, \boldsymbol{u}_t$ followed a wrapped normal distribution. Therefore

$$p(X_{t,3}|X_{t-1,3}, \boldsymbol{u}_t) = \frac{1}{\sqrt{2\pi\sigma^2_{\alpha_t}}} \sum_{k=-\infty}^{\infty} \exp\left(-\frac{1}{2} \frac{(X_{t,3} + 2\pi k - (X_{t-1,3} + \alpha_t))^2}{\sigma^2_{\alpha_t}} \right). \tag{3.11}$$

An approximate solution was thus given by

$$p(X_{t,3}|X_{t-1,3}, \boldsymbol{u}_t) = \frac{1}{\sqrt{2\pi\sigma^2_{\alpha_t}}} \sum_{k=-\kappa}^{\kappa} \exp\left(-\frac{1}{2} \frac{(X_{t,3} + 2\pi k - (X_{t-1,3} + \alpha_t))^2}{\sigma^2_{\alpha_t}} \right). \tag{3.12}$$

where κ is an integer. Using equations (3.8), (3.9), (3.10) and (3.12) we could acquire the following lengthy approximation of the transition density;

$$\begin{aligned} p(\boldsymbol{X}_t|\boldsymbol{X}_{t-1}, \boldsymbol{u}_t) = \frac{1}{2\pi\sigma_{d_{t,1}}\sigma_{d_{t,2}}} \exp\left(-\frac{1}{2}\Sigma^{-1}_{\boldsymbol{X}} \left(\begin{bmatrix} X_{t,1} \\ X_{t,2} \end{bmatrix} - \begin{bmatrix} X_{t-1,1} + v_t\delta\cos(X_{t-1,3}) \\ X_{t-1,2} + v_t\delta\sin(X_{t-1,3}) \end{bmatrix} \right)^2 \right) \\ \cdot \frac{1}{\sqrt{2\pi\sigma^2_{\alpha_t}}} \sum_{k=-\kappa}^{\kappa} \exp\left(-\frac{1}{2} \frac{(X_{t,3} + 2\pi k - (X_{t-1,3} + \alpha_t))^2}{\sigma^2_{\alpha_t}} \right) \end{aligned} \tag{3.13}$$

where $\Sigma_{\boldsymbol{X}} = \begin{bmatrix} \sigma^2_{d_{t,1}} & 0 \\ 0 & \sigma^2_{d_{t,2}} \end{bmatrix}$.

3.5 Selecting an upper bound for the transition density

In order to utilize the accept-reject sampling method as part of the PaRIS algorithm we needed to determine an upper bound to the transition density. Using the analytical expression of the transition density presented in equation (3.13) we derived that an upper bound was given by

$$\begin{aligned}\max_{\boldsymbol{x}_t} p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1},\boldsymbol{u}_t) &\approx \frac{1}{\sqrt{2\pi\sigma_{d_{t,1}}^2}}\frac{1}{\sqrt{2\pi\sigma_{d_{t,2}}^2}}\sum_{k=-\kappa}^{\kappa}\frac{\exp\left(-\frac{(2\pi k)^2}{2\sigma_{\alpha_t}^2}\right)}{\sqrt{2\pi\sigma_{\alpha_t}^2}} \\ &= \frac{1}{(2\pi)^{3/2}\sigma_{d_{t,1}}\sigma_{d_{t,2}}\sigma_{\alpha_t}}\sum_{k=-\kappa}^{\kappa}\exp\left(-\frac{(2\pi k)^2}{2\sigma_{\alpha_t}^2}\right) \end{aligned} \tag{3.14}$$

Hence the constant ϵ_+ bounding the transition density could be set to

$$\epsilon_+ = \frac{1}{(2\pi)^{3/2}\sigma_{d_{t,1}}\sigma_{d_{t,2}}\sigma_{\alpha_t}}\sum_{k=-\kappa}^{\kappa}\exp\left(-\frac{(2\pi k)^2}{2\sigma_{\alpha_t}^2}\right). \tag{3.15}$$

Note that this value is an underestimate and does not fulfill that $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1},\boldsymbol{u}_t) \leq \epsilon_+$ due to κ being a finite constant. However, this should not pose an issue for sufficiently large values of a.

3.6 Derivation of EM updating formula

As we aimed to utilize the EM algorithm presented in Section 2.10 we had to find a solution to the inherit maximization problem. Furthermore, we wished to find a closed form solution to ease implementation. Since the observations $\boldsymbol{Z}_{0:T}$ were considered conditionally independent of the control $\boldsymbol{u}_{0:T}$ given the state sequence $\boldsymbol{X}_{0:T}$ we had by Bayes theorem that

$$f(\boldsymbol{X}_{0:T},\boldsymbol{Z}_{0:T}|\boldsymbol{m},\boldsymbol{u}_{0:T}) = f(\boldsymbol{Z}_{0:T}|\boldsymbol{m},\boldsymbol{X}_{0:T}=\boldsymbol{x}_{0:T})f(\boldsymbol{X}_{0:T}|\boldsymbol{m},\boldsymbol{u}_{0:T}). \tag{3.16}$$

Given that $\boldsymbol{X}_{0:T}$ was assumed to be independent of the map $\boldsymbol{m}$ we had that $f(\boldsymbol{X}_{0:T}|\boldsymbol{m}, \boldsymbol{u}_t) = f(\boldsymbol{X}_{0:T}|\boldsymbol{u}_t)$ and thus

$$\arg\max_{\boldsymbol{m}} f(\boldsymbol{X}_{0:T}, \boldsymbol{Z}_{0:T}|\boldsymbol{m}, \boldsymbol{u}_t) = \arg\max_{\boldsymbol{m}} f(\boldsymbol{Z}_{0:T}|\boldsymbol{m}, \boldsymbol{X}_{0:T} = \boldsymbol{x}_{0:T}). \quad (3.17)$$

Since the observations at different times were assumed to be conditionally independent given the map and state we had that

$$f(\boldsymbol{Z}_{0:T}|\boldsymbol{m}, \boldsymbol{X}_{0:T} = \boldsymbol{x}_{0:T}) = \prod_{t=0}^{T} f(\boldsymbol{Z}_{t,\mathcal{A}_t}|\boldsymbol{m}, \boldsymbol{X}_t = \boldsymbol{x}_t). \quad (3.18)$$

This could be factorized further under the assumption that the different landmark observations were conditionally independent given the map and the state. Hence

$$\prod_{t=0}^{T} f(\boldsymbol{Z}_{t,\mathcal{A}_t}|\boldsymbol{m}, \boldsymbol{X}_t = \boldsymbol{x}_t) = \prod_{i=1}^{N}\prod_{t=0}^{T} \mathbb{1}_{i\in\mathcal{A}_t} f(\boldsymbol{Z}_{t,i}|\boldsymbol{m}_i, \boldsymbol{X}_t = \boldsymbol{x}_t) \quad (3.19)$$

where $\mathbb{1}$ denotes the indicator function. Using our observation model we had that

$$\begin{aligned} f(\boldsymbol{Z}_{t,i}|\boldsymbol{m}_i, \boldsymbol{X}_t = \boldsymbol{x}_t) = \\ &= \frac{1}{\sqrt{2\pi}\sigma_\rho\sigma_\phi} \exp\left(-\frac{1}{2}\Sigma_{\boldsymbol{Z}}^{-1}\left(\boldsymbol{Z}_{t,i} - \begin{bmatrix}\rho^*(\boldsymbol{x}_t, \boldsymbol{m}_i)\\ \phi^*(\boldsymbol{x}_t, \boldsymbol{m}_i)\end{bmatrix}\right)^2\right) \\ &\propto \exp\left(-\frac{1}{2}\Sigma_{\boldsymbol{Z}}^{-1}\left(\boldsymbol{Z}_{t,i} - \begin{bmatrix}\rho^*(\boldsymbol{x}_t, \boldsymbol{m}_i)\\ \phi^*(\boldsymbol{x}_t, \boldsymbol{m}_i)\end{bmatrix}\right)^2\right) \end{aligned} \quad (3.20)$$

where $\Sigma_{\boldsymbol{Z}} = \begin{bmatrix}\sigma_\rho^2 & 0\\ 0 & \sigma_\phi^2\end{bmatrix}$. As the logarithm of a function is strictly increasing we had that

$$\begin{aligned} \arg\max_{\boldsymbol{m}_i} f(\boldsymbol{Z}_{t,i}|\boldsymbol{m}_i, \boldsymbol{X}_t = \boldsymbol{x}_t) &= \arg\max_{\boldsymbol{m}_i} \log\left(f(\boldsymbol{Z}_{t,i}|\boldsymbol{m}_i, \boldsymbol{X}_t = \boldsymbol{x}_t)\right) \\ &= \arg\max_{\boldsymbol{m}_i} -\frac{1}{2}\Sigma_{\boldsymbol{Z}}^{-1}\left(\boldsymbol{Z}_{t,i} - \begin{bmatrix}\rho^*(\boldsymbol{x}_t, \boldsymbol{m}_i)\\ \phi^*(\boldsymbol{x}_t, \boldsymbol{m}_i)\end{bmatrix}\right)^2 \\ &= \arg\max_{\boldsymbol{m}_i} -\Sigma_{\boldsymbol{Z}}^{-1}\left(\boldsymbol{Z}_{t,i} - \begin{bmatrix}\rho^*(\boldsymbol{x}_t, \boldsymbol{m}_i)\\ \phi^*(\boldsymbol{x}_t, \boldsymbol{m}_i)\end{bmatrix}\right)^2. \end{aligned} \quad (3.21)$$

The probability density $f(\boldsymbol{Z}_{t,i}|\boldsymbol{m}_i, \boldsymbol{X}_t = \boldsymbol{x}_t)$ is not part of the exponential family, partly due to the trigonometric expression in equation (3.6)

and does not offer a closed form solution the problem of maximization. As a remedy we approximated $\rho^*(\boldsymbol{x}_t, \boldsymbol{m}_i)$ and $\phi^*(\boldsymbol{x}_t, \boldsymbol{m}_i)$ by the first order Taylor series expansion. Hence given a previous map estimate $\boldsymbol{m}'_i$ we had the linear estimates

$$\begin{aligned}
\rho^*(\boldsymbol{x}_t, \boldsymbol{m}_i) \approx \rho^*(\boldsymbol{x}_t, \boldsymbol{m}'_i) + \frac{\partial \rho^*(\boldsymbol{x}_t, \boldsymbol{m}'_i)}{\partial m_{i,1}}(m_{i,1} - m'_{i,1}) \\
+ \frac{\partial \rho^*(\boldsymbol{x}_t, \boldsymbol{m}'_i)}{\partial m_{i,2}}(m_{i,2} - m'_{t,2}) \coloneqq \hat{\rho}^*(\boldsymbol{x}_t, \boldsymbol{m}'_i, \boldsymbol{m}_i) \\
\phi^*(\boldsymbol{x}_t, \boldsymbol{m}_i) \approx \phi^*(\boldsymbol{x}_t, \boldsymbol{m}'_i) + \frac{\partial \phi^*(\boldsymbol{x}_t, \boldsymbol{m}'_i)}{\partial m_{i,1}}(m_{i,1} - m'_{i,1}) \\
+ \frac{\partial \phi^*(\boldsymbol{x}_t, \boldsymbol{m}'_i)}{\partial m_{i,2}}(m_{i,2} - m'_{i,2}) \coloneqq \hat{\phi}^*(\boldsymbol{x}_t, \boldsymbol{m}'_i, \boldsymbol{m}_i).
\end{aligned} \tag{3.22}$$

Inserting (3.22) in (3.21) yielded the intermediate quantity as $Q(m|m') = \sum_{t=0}^{T} \sum_{i=1}^{q} Q_{t,i}(m|m')$ where

$$\begin{aligned}
&Q_{t,i}(m|m') = \\
&= \mathbb{E}\left[-\Sigma_{\boldsymbol{Z}}^{-1} \left(\boldsymbol{Z}_{t,i} - \begin{bmatrix} \hat{\rho}^*(\boldsymbol{x}_t, \boldsymbol{m}'_i, \boldsymbol{m}_i) \\ \hat{\phi}^*(\boldsymbol{x}_t, \boldsymbol{m}'_i, \boldsymbol{m}_i) \end{bmatrix} \right)^2 \middle| \boldsymbol{m}'_i, \rho_i, \phi_i \right].
\end{aligned} \tag{3.23}$$

Since $\hat{\rho}^*(\boldsymbol{x}_t, \boldsymbol{m}'_i, \boldsymbol{m}_i)$ and $\hat{\phi}^*(\boldsymbol{x}_t, \boldsymbol{m}'_i, \boldsymbol{m}_i)$ are linear functions with respect to m_i we have that $Q(m|m')$ is a quadratic function with respect to m_i. Hence a maximum of $Q(m|m')$ was easily obtained by solving for $\frac{\partial Q(m|m')}{\partial m_i} = 0$ which resulted in the following updating formulas

$$\begin{aligned}
m_{i,1} &= \frac{d \cdot c - e \cdot b}{a \cdot d - b^2} \\
m_{i,2} &= \frac{e \cdot a - b \cdot c}{a \cdot d - b^2}
\end{aligned} \tag{3.24}$$

where $a - e$ were defined as follows

$$
\begin{aligned}
a &= \sum_{t=0}^{T} \frac{1}{\sigma_\rho^2} \frac{\partial \rho^*(\boldsymbol{x}_t, \boldsymbol{m}_i')}{\partial m_{i,1}}^2 + \frac{1}{\sigma_\phi^2} \frac{\partial \phi^*(\boldsymbol{x}_t, \boldsymbol{m}_i')}{\partial m_{i,1}}^2 \\
b &= \sum_{t=0}^{T} \frac{1}{\sigma_\rho^2} \frac{\partial \rho^*(\boldsymbol{x}_t, \boldsymbol{m}_i')}{\partial m_{i,1}} \frac{\partial \rho^*(\boldsymbol{x}_t, \boldsymbol{m}_i')}{\partial m_{i,2}} + \frac{1}{\sigma_\phi^2} \frac{\partial \phi^*(\boldsymbol{x}_t, \boldsymbol{m}_i')}{\partial m_{i,1}} \frac{\partial \phi^*(\boldsymbol{x}_t, \boldsymbol{m}_i')}{\partial m_{i,2}} \\
c &= \sum_{t=0}^{T} \frac{1}{\sigma_\phi^2} \frac{\partial \phi^*(\boldsymbol{x}_t, \boldsymbol{m}_i')}{\partial m_{i,1}} \left(Z_{t,i,2} - \phi^*(\boldsymbol{x}_t, \boldsymbol{m}_i') + \frac{\partial \phi^*(\boldsymbol{x}_t, \boldsymbol{m}_i')}{\partial m_{i,1}} m_{i,1}' + \frac{\partial \phi^*(\boldsymbol{x}_t, \boldsymbol{m}_i')}{\partial m_{i,2}} m_{i,2}' \right) \\
&\quad + \frac{1}{\sigma_\rho^2} \frac{\partial \rho^*(\boldsymbol{x}_t, \boldsymbol{m}_i')}{\partial m_{i,1}} \left(Z_{t,i,1} - \rho^*(\boldsymbol{x}_t, \boldsymbol{m}_i') + \frac{\partial \rho^*(\boldsymbol{x}_t, \boldsymbol{m}_i')}{\partial m_{i,1}} m_{i,1}' + \frac{\partial \rho^*(\boldsymbol{x}_t, \boldsymbol{m}_i')}{\partial m_{i,2}} m_{i,2}' \right) \\
d &= \sum_{t=0}^{T} \frac{1}{\sigma_\rho^2} \frac{\partial \rho^*(\boldsymbol{x}_t, \boldsymbol{m}_i')}{\partial m_{i,2}}^2 + \frac{1}{\sigma_\phi^2} \frac{\partial \phi^*(\boldsymbol{x}_t, \boldsymbol{m}_i')}{\partial m_{i,2}}^2 \\
e &= \sum_{t=0}^{T} \frac{1}{\sigma_\phi^2} \frac{\partial \phi^*(\boldsymbol{x}_t, \boldsymbol{m}_i')}{\partial m_{i,2}} \left(Z_{t,i,2} - \phi^*(\boldsymbol{x}_t, \boldsymbol{m}_i') + \frac{\partial \phi^*(\boldsymbol{x}_t, \boldsymbol{m}_i')}{\partial m_{i,1}} m_{i,1}' + \frac{\partial \phi^*(\boldsymbol{x}_t, \boldsymbol{m}_i')}{\partial m_{i,2}} m_{i,2}' \right) \\
&\quad + \frac{1}{\sigma_\rho^2} \frac{\partial \rho^*(\boldsymbol{x}_t, \boldsymbol{m}_i')}{\partial m_{i,2}} \left(Z_{t,i,1} - \rho^*(\boldsymbol{x}_t, \boldsymbol{m}_i') + \frac{\partial \rho^*(\boldsymbol{x}_t, \boldsymbol{m}_i')}{\partial m_{i,1}} m_{i,1}' + \frac{\partial \rho^*(\boldsymbol{x}_t, \boldsymbol{m}_i')}{\partial m_{i,2}} m_{i,2}' \right).
\end{aligned}
\tag{3.25}
$$

Although quite cumbersome as an expression we were only required to compute the following sufficient statistics in order to utilize the up-

dating formulas given by equation (3.24):

$$
\begin{aligned}
S_{T,1}(\boldsymbol{X}_{0:T}, \boldsymbol{m}_i') &\coloneqq \sum_{t=0}^{T} \frac{\partial \rho^*(\boldsymbol{x}_t, \boldsymbol{m}_i')}{\partial m_{i,1}}^2, & S_{T,2}(\boldsymbol{X}_{0:T}, \boldsymbol{m}_i') &\coloneqq \sum_{t=0}^{T} \frac{\partial \rho^*(\boldsymbol{x}_t, \boldsymbol{m}_i')}{\partial m_{i,2}}^2 \\
S_{T,3}(\boldsymbol{X}_{0:T}, \boldsymbol{m}_i') &\coloneqq \sum_{t=0}^{T} \frac{\partial \phi^*(\boldsymbol{x}_t, \boldsymbol{m}_i')}{\partial m_{i,1}}^2, & S_{T,4}(\boldsymbol{X}_{0:T}, \boldsymbol{m}_i') &\coloneqq \sum_{t=0}^{T} \frac{\partial \phi^*(\boldsymbol{x}_t, \boldsymbol{m}_i')}{\partial m_{i,2}}^2 \\
S_{T,5}(\boldsymbol{X}_{0:T}, \boldsymbol{m}_i') &\coloneqq \sum_{t=0}^{T} \rho^* \frac{\partial \rho^*(\boldsymbol{x}_t, \boldsymbol{m}_i')}{\partial m_{i,1}}, & S_{T,6}(\boldsymbol{X}_{0:T}, \boldsymbol{m}_i') &\coloneqq \sum_{t=0}^{T} \rho^* \frac{\partial \rho^*(\boldsymbol{x}_t, \boldsymbol{m}_i')}{\partial m_{i,2}} \\
S_{T,7}(\boldsymbol{X}_{0:T}, \boldsymbol{m}_i') &\coloneqq \sum_{t=0}^{T} \phi^* \frac{\partial \phi^*(\boldsymbol{x}_t, \boldsymbol{m}_i')}{\partial m_{i,1}}, & S_{T,8}(\boldsymbol{X}_{0:T}, \boldsymbol{m}_i') &\coloneqq \sum_{t=0}^{T} \phi^* \frac{\partial \phi^*(\boldsymbol{x}_t, \boldsymbol{m}_i')}{\partial m_{i,2}} \\
S_{T,9}(\boldsymbol{X}_{0:T}, \boldsymbol{m}_i') &\coloneqq \sum_{t=0}^{T} Z_{t,i,1} \frac{\partial \rho^*(\boldsymbol{x}_t, \boldsymbol{m}_i')}{\partial m_{i,1}}, & S_{T,10}(\boldsymbol{X}_{0:T}, \boldsymbol{m}_i') &\coloneqq \sum_{t=0}^{T} Z_{t,i,1} \frac{\partial \rho^*(\boldsymbol{x}_t, \boldsymbol{m}_i')}{\partial m_{i,2}} \\
S_{T,11}(\boldsymbol{X}_{0:T}, \boldsymbol{m}_i') &\coloneqq \sum_{t=0}^{T} Z_{t,i,2} \frac{\partial \phi^*(\boldsymbol{x}_t, \boldsymbol{m}_i')}{\partial m_{i,1}}, & S_{T,12}(\boldsymbol{X}_{0:T}, \boldsymbol{m}_i') &\coloneqq \sum_{t=0}^{T} Z_{t,i,2} \frac{\partial \phi^*(\boldsymbol{x}_t, \boldsymbol{m}_i')}{\partial m_{i,2}}.
\end{aligned}
\tag{3.26}
$$

Note that the expressions above are all additive functionals on the form mentioned in equation (2.2).

Chapter 4

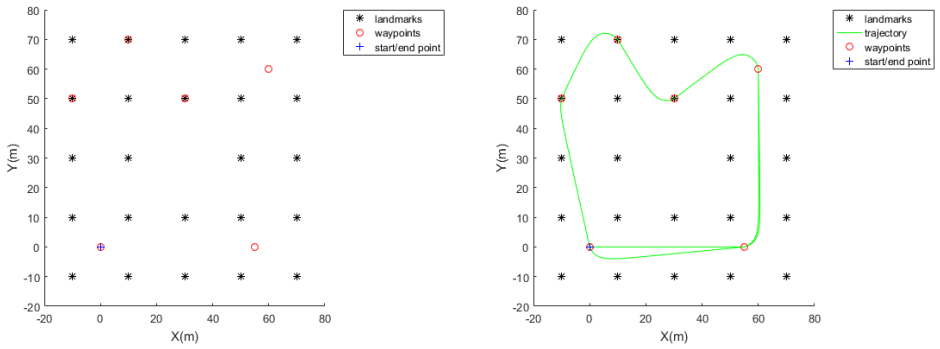
Simulations

This section aims to provide a clear depiction of the performed experiments and report the performance of the employed SLAM solution methods.

4.1 Generation of observation and control sequences

In order to create fair comparisons between the algorithms we required the inputted realizations of the observation sequence, $\boldsymbol{z}_{0:T}$ and the control sequence $\boldsymbol{u}_{0:T}$ to be the fixed and generated beforehand. Furthermore, as the SLAM problem is prevalent in a physical environment we introduced units to make the results easily graspable.

The map of landmarks used is presented in Figure 4.1a together with the waypoints, which the robot aims to traverse through. For the generated data set used, the robot looped through these waypoints three times in an anti-clockwise fashion starting at the origin. The robot had a top speed of $3\,\mathrm{m/s}$ and would take readings and update the control input every $0.1\,\mathrm{s}$ resulting in a total of 2771 readings. The robot had a field of view of $30\,\mathrm{m}$ and 180 degrees. Figure 4.1b shows the robot trajectory generated in this setting.



(a) The map with waypoints.

(b) The map with the simulated robot trajectory. The robot does three laps anticlockwise starting at the origin

Figure 4.1: The map with and without the robot trajectory.

4.2 Design parameters

We considered the model described in Section 3.3 with noise standard deviations set to $(\sigma_\rho, \sigma_\phi, \sigma_{d_{t,1}}, \sigma_{d_{t,2}}, \sigma_{\alpha_t}) = (0.25\text{m}, 5\pi/180, 0.05\text{m}, 0.05\text{m}, 0.0175)$. The three smoothing algorithms for computing the sufficient statistics were restricted to an average runtime close to 277 seconds, which resulted in the choices of design parameters presented in Table 4.1. The runtime was determined such that the computations would keep up with the data gathering pace required in an online setting.

Table 4.1: Design parameters with time restriction of 55 seconds

	FLS	FFBSM	PaRIS
number of particles, M	3970	40	513
lag, Δ_T	20	-	-
number of samples, $\tilde{M}$	-	-	2

The integer κ used to approximate the maximum bound ϵ_+ of the transition density was set to $\kappa = 1$ (see Section 3.5). The difference between the approximate maximum bound for $\kappa = 1$ and $\kappa = 2$ was very low (well below 10^{-100}). Hence the approximation was very close to the true value.

4.3 Map adjustment

As the estimated maps often are translated and/or rotated it can be hard to assess and compare results. In order to ease interpretation the maps were translated and rotated to best fit the maximum likelihood map. The rotation was calculated by minimizing the mean adjusted function

$$\arg\min_{\gamma} \sum_{i=1}^{N} || \left(\hat{\boldsymbol{m}}_{\boldsymbol{i}} - \frac{\sum_{j=1}^{N} \hat{\boldsymbol{m}}_j}{N} \right) - \left(\boldsymbol{m}^{ML} - \frac{\sum_{j=1}^{N} \boldsymbol{m}_j^{ML}}{N} \right) ||^2 \tag{4.1}$$

where $\boldsymbol{m}^{ML}$ is the maximum likelihood map and γ is the angle related to the polar coordinate representation of the estimated map $\hat{\boldsymbol{m}}$ such that

$$\hat{\boldsymbol{m}}_i = \begin{bmatrix} \hat{m}_{i,1} \\ \hat{m}_{i,2} \end{bmatrix} = \begin{bmatrix} \hat{r}_i \cos(\gamma) \\ \hat{r}_i \sin(\gamma) \end{bmatrix}. \tag{4.2}$$

This least squares minimization problem was solved numerically.

4.3.1 Maximum likelihood map versus true map

As all three algorithms produce maximum-likelihood estimations of the map and trajectory it is reasonable to compare the results to that of the maximum-likelihood map and trajectory. The then difference between the true map and trajectory and the maximum-likelihood map and trajectory is then another, separate comparison to make. Figure 4.2 shows the maximum-likelihood map with the true map. The maximum-likelihood map was generated using the PaRIS algorithm with $M = 1000$ particles, $\tilde{M} = 3$ backward samples and 100 iterations of the EM algorithm.

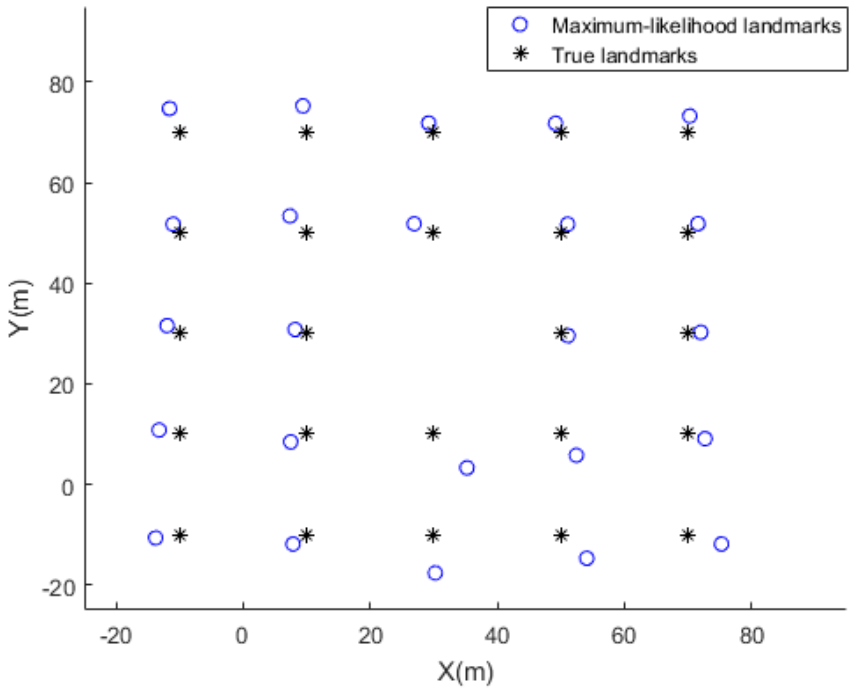
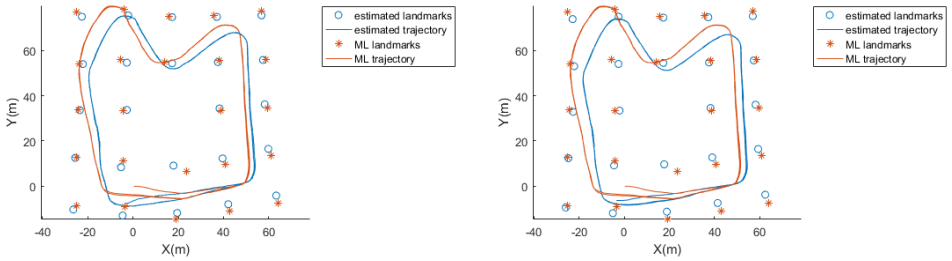


Figure 4.2: The maximum-likelihood map versus the true map.

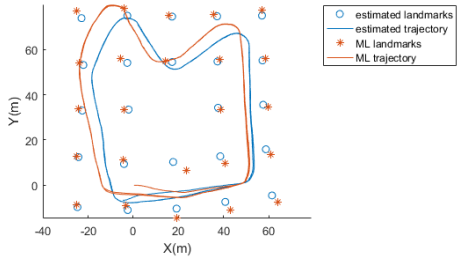
4.4 Simulation results

Figure 4.3 shows map and trajectory estimates for all smoothing algorithms. The map estimates were taken after 10 iterations of the EM algorithm using the three smoothing methods with the design parameters presented in Table 4.1.



(a) FLS

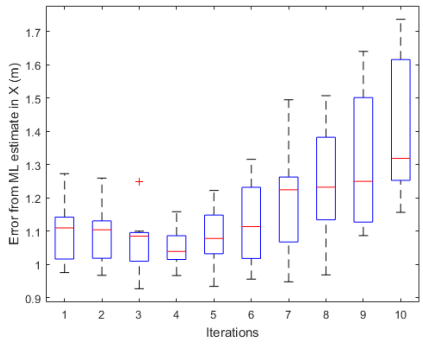
(b) FFBSm



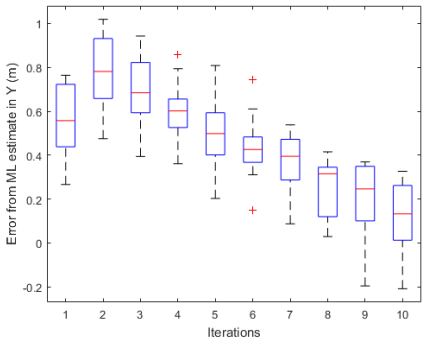
(c) PaRIS

Figure 4.3: The maximum-likelihood map and trajectory and the estimated maps and trajectories computed using different smoothing techniques.

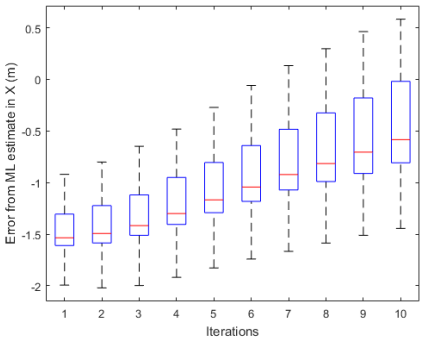
Figures 4.4, 4.5, 4.6 and 4.7 presents boxplots related to the signed error of landmarks estimates in relation to the maximum likelihood estimate. The two targeted landmarks are placed at $(-10, 50)$ and $(70, -10)$ (see Figure 4.2) and are the most and the least observed landmarks respectively. The landmark estimates were generated from 10 simulations based on the same observation and control data. The algorithms all performed 10 iterations with design parameters described in Section 4.2. The landmark placements were adjusted as described in Section 4.3.



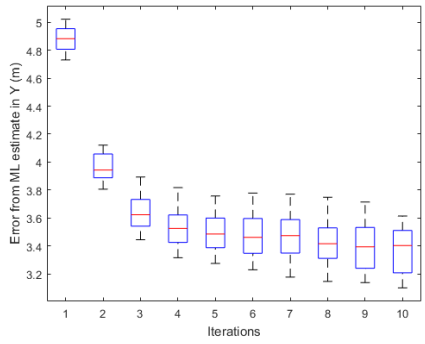
(a) Signed error of estimated landmarks position in relation to the maximum likelihood (ML) estimate in the X coordinate. The targeted landmark was the most observed landmark with 450 observations.



(b) Signed error of estimated landmarks position in relation to the maximum likelihood (ML) estimate in the Y coordinate. The targeted landmark was the most observed landmark with 450 observations.

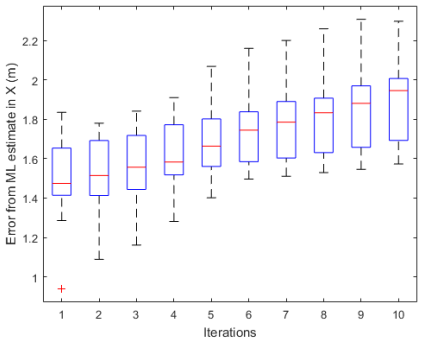


(c) Signed error of estimated landmarks position in relation to the maximum likelihood (ML) estimate in the X coordinate. The targeted landmark was the least observed landmark with 180 observations.

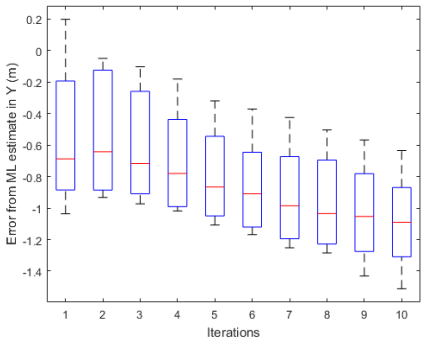


(d) Signed error of estimated landmarks position in relation to the maximum likelihood (ML) estimate in the Y coordinate. The targeted landmark was the least observed landmark with 180 observations.

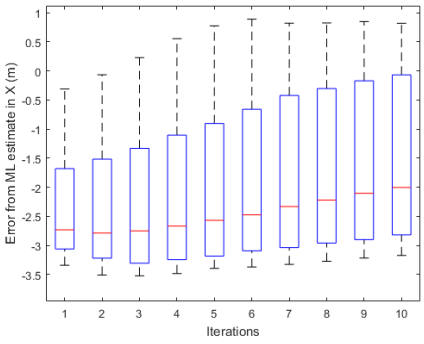
Figure 4.4: Box plots related to the error of landmark estimates obtained using the FLS technique.



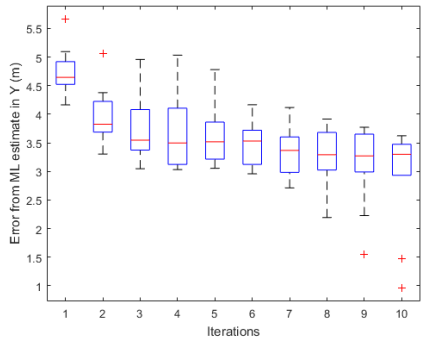
(a) Signed error of estimated landmarks position in relation to the maximum likelihood (ML) estimate in the X coordinate. The targeted landmark was the most observed landmark with 450 observations.



(b) Signed error of estimated landmarks position in relation to the maximum likelihood (ML) estimate in the Y coordinate. The targeted landmark was the most observed landmark with 450 observations.

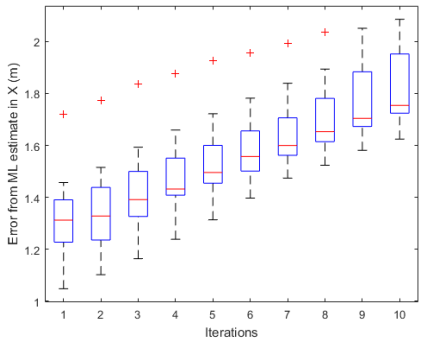


(c) Signed error of estimated landmarks position in relation to the maximum likelihood (ML) estimate in the X coordinate. The targeted landmark was the least observed landmark with 180 observations.

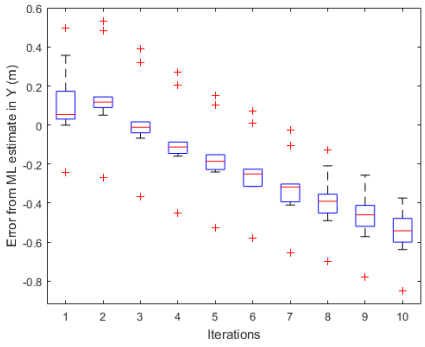


(d) Signed error of estimated landmarks position in relation to the maximum likelihood (ML) estimate in the Y coordinate. The targeted landmark was the least observed landmark with 180 observations.

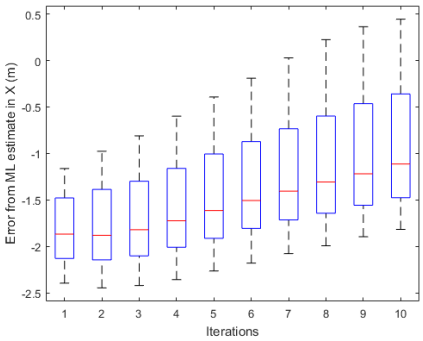
Figure 4.5: Box plots related to the error of landmark estimates obtained using forward-only FFBSM.



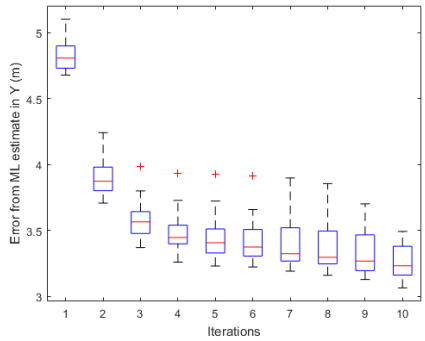
(a) Signed error of estimated landmarks position in relation to the maximum likelihood (ML) estimate in the X coordinate. The targeted landmark was the most observed landmark with 450 observations.



(b) Signed error of estimated landmarks position in relation to the maximum likelihood (ML) estimate in the Y coordinate. The targeted landmark was the most observed landmark with 450 observations.



(c) Signed error of estimated landmarks position in relation to the maximum likelihood (ML) estimate in the X coordinate. The targeted landmark was the least observed landmark with 180 observations.



(d) Signed error of estimated landmarks position in relation to the maximum likelihood (ML) estimate in the Y coordinate. The targeted landmark was the least observed landmark with 180 observations.

Figure 4.6: Box plots related to the error of landmark estimates obtained using PaRIS.

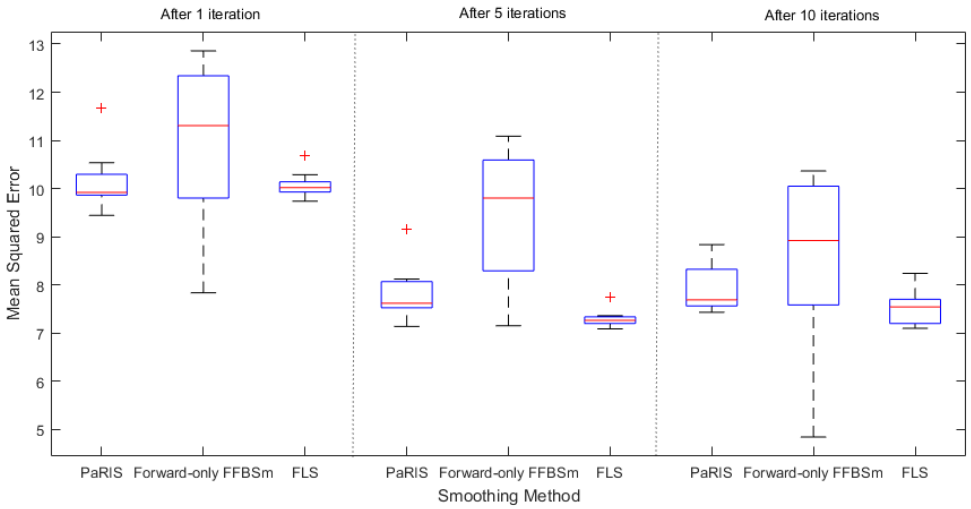
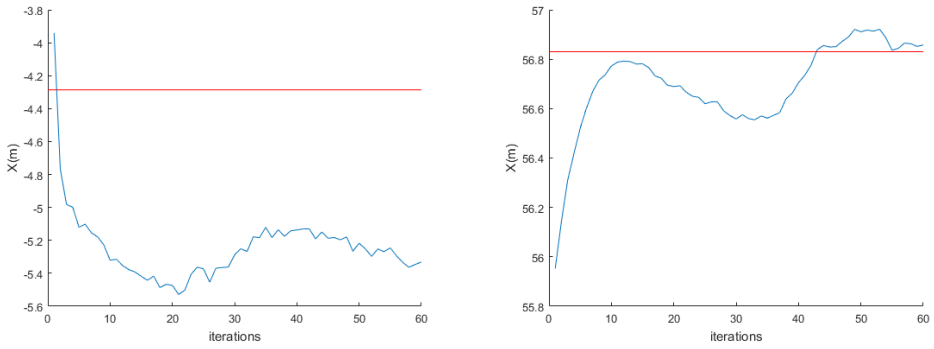


Figure 4.7: The overall landmark mean squared error for different smoothing methods and iterations. The boxplots are based on 10 simulations. Notably forward-only FFBSm has larger and more spread mean squared error for all iterations.

Figures 4.8, 4.10 and 4.12 presents plots related to the convergence of landmark estimates using FLS, forward-only FFBSm and PaRIS respectively. The number of iterations was set to 60 and all algorithms used the same input data set. The landmark placement was again adjusted to best fit the map for all iterations.



(a) Estimates of the x position as a function of iterations for the landmark located at $(10, 10)$. The blue line is the estimated position and the red line is the maximum-likelihood position.

(b) Estimates of the x position as a function of iterations for the landmark located at $(70, 70)$. The blue line is the estimated position and the red line is the maximum-likelihood position.

Figure 4.8: Landmark estimates in the X-position obtained using the FLS and EM algorithm as a function of iterations.

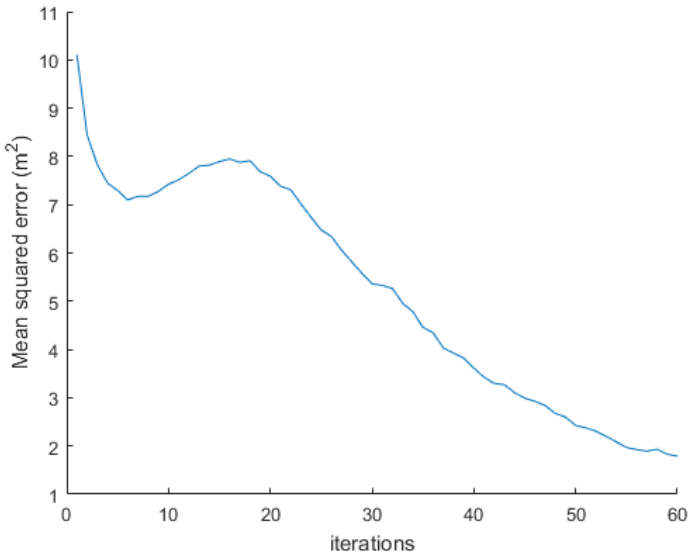
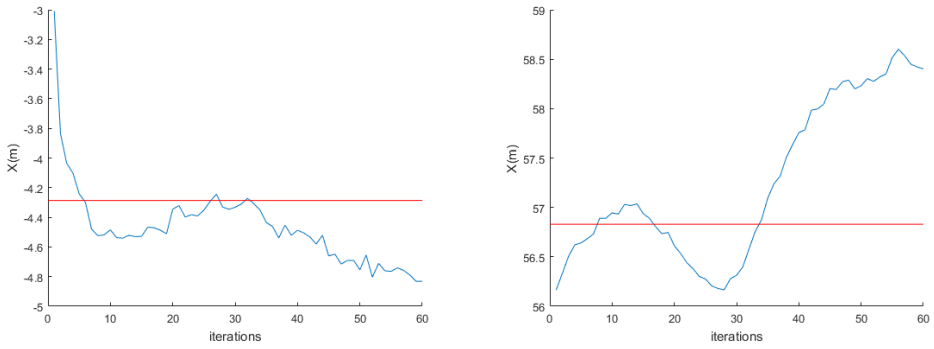


Figure 4.9: Plot of the mean squared error for estimated map in relation to the maximum-likelihood map versus iterations of the FLS and EM algorithm. There is a clear downward trend in mean squared error apart from an increase at between around 10 and 20 iterations.



(a) Estimates of the x position as a function of iterations for the landmark located at $(10, 10)$. The blue line is the estimated position and the red line is the maximum-likelihood position.

(b) Estimates of the x position as a function of iterations for the landmark located at $(70, 70)$. The blue line is the estimated position and the red line is the maximum-likelihood position.

Figure 4.10: Landmark estimates in the X-position obtained using the forward-only FFBSm and EM algorithm as a function of iterations.

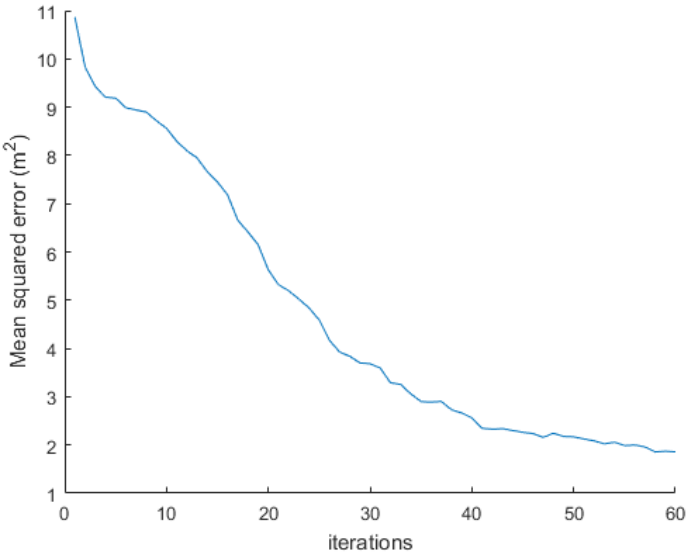
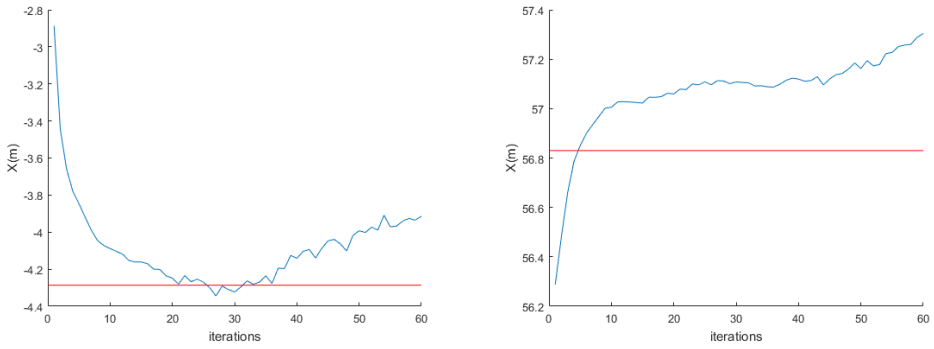


Figure 4.11: Plot of the mean squared error for estimated map in relation to the maximum-likelihood map versus iterations of the forward-only FFBSm and EM algorithm. There is a clear downward trend in mean squared error.



(a) Estimates of the x position as a function of iterations for the landmark located at $(10, 10)$. The blue line is the estimated position and the red line is the maximum-likelihood position.

(b) Estimates of the x position as a function of iterations for the landmark located at $(70, 70)$. The blue line is the estimated position and the red line is the maximum-likelihood position.

Figure 4.12: Landmark estimates in the X-position obtained using the PaRIS and EM algorithm as a function of iterations.

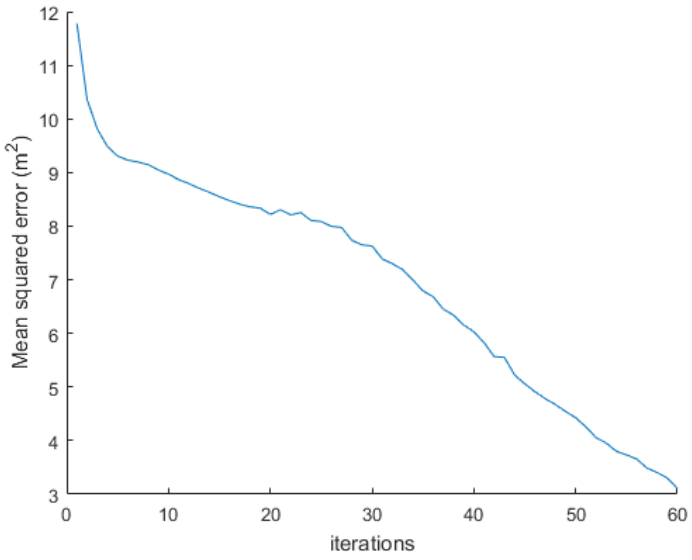


Figure 4.13: Plot of the mean squared error for estimated map in relation to the maximum-likelihood map versus iterations of the PaRIS and EM algorithm. There is a clear downward trend in mean squared error.

Chapter 5

Discussion

The individual landmark estimates did not strictly converge to the maximum likelihood placement, see for example Figure 4.4a or 4.5b. Even the most observed landmark exhibited this behaviour, seemingly independently of the method used (see for example Figure 4.4a, 4.5a,4.6a). Although this can be somewhat alarming it is important to note that the overall mean square error for all the landmarks is decreasing under the same iteration period, see Figure 4.7. This is not a surprising result as we perform maximum-likelihood estimation with regards to the whole map. However, there is an interesting increase in the mean squared error for the FLS-based algorithm in the same figure. This might be due to the process not "forgetting" previous states fast enough for the approximation presented in equation (2.13) to be good. It may also be due to the first order Taylor approximation present in the EM algorithm or due to the filter and smoothing estimating processes being stochastic in nature, thus creating some variation in the map estimation process.

The in-sample variance was not strictly decreasing for the two chosen landmarks as can be seen in Figure 4.4. This held true for all methods. Overall, the most observed landmark (see Figure 4.4 (a)-(b), 4.5 (a)-(b) and 4.6 (a)-(b)) had lower variance than the least observed landmark (see Figure 4.4 (c)-(d), 4.5 (c)-(d) and 4.6 (c)-(d)) which is not a surprising result. The spread in overall landmark mean square error was highest for forward-only FFBSm, and lowest for FLS as visible in figure 4.7. This may be expected as when the number of particles used (see

Table 4.1) increases, the consistency in the estimation process should too. The reason for this is that having fewer particles would imply that the state space is less explored, thus reducing consistency in the filter estimation process of the robot trajectory.

As visible in Figure 4.9, 4.11 and 4.13 all algorithms converged towards the maximum likelihood estimate with less clear convergence for individual landmarks (see Figure 4.8,4.10 and 4.12). There were mostly small deviations from the downward trend of the means squared error for all methods, likely due to variation in the smoothing process between EM iterations. However, again for the FLS method there was a clear increase between the 10 and 20 iterations. As this increase was much larger than for the other methods (which utilizes fewer number of particles) it raises the likelihood that this is a phenomena specific to the FLS process. However, it is noteworthy that this is based on single, large runs of the algorithms. This behaviour might therefore be universal for all algorithms and should probably be studied further before drawing a clear conclusion.

Chapter 6

Conclusion

The aim of this paper was to explore new, and compare solutions to the back-end part of SLAM based on particle methods. As such, three different particle-based SLAM algorithms were implemented. The key difference between the employed algorithms were the smoothing techniques used. These smoothing techniques were fixed lag smoothing (FLS) [19], forward-only forward-filtering backward-smoothing (forward-only FFBSm) [17] and the particle-based, rapid incremental smoother (PaRIS) algorithm [18]. All three methods were successful in constructing maximum likelihood estimates of the map and robot trajectory. Notably, with a runtime restriction of 277 seconds per smoothing iteration, the forward-only FFBSm performed the worst with lower consistency than both the other algorithms. This is likely because of the lower particle number available due to the quadratic complexity of the employed method. The most consistent estimator was the FLS-based algorithm. However it also saw an increase in the mean squared error of the map estimates, possibly due to biases present in the smoothing method. In the window of 10 iterations, the FLS- and the PaRIS-based algorithm performed considerably better than the forward-only FFBSm algorithm. Although more research is required, both the FLS- and the PaRIS-based algorithm shows some promising results for the application of particle-based methods to the SLAM problem.

6.1 Improvements

The SLAM model and estimation algorithms presented in this paper has a high degree of freedom. There is therefore many possible aspect to consider when comparing algorithms. Different algorithm-unique design parameters such as the lag for the FLS algorithm could be optimized further. The forward-only FFBSm method used had quadratic complexity, whereas it is possible to reduce this to $\mathcal{O}(M \log M)$ [17]. Using the lower complexity version of forward-only FFBSm could remedy the experienced relatively high in-sample variance.

Due to the large simulation times, only 10 samples from each algorithm was used. A larger sample size would be appropriate to get more easily interpreted results. It could also bring insight to whether the increasing mean squared error is specific to the FLS-based method.

For the used model presented in Section 3.3 it is possible for the robot to detect landmarks outside its field of view. This is because the maximum detectable range and angle is determined as a realization of a Gaussian centered around the cut-off range and angle. As this might not be possible in a real-world application the model can be improved by more strictly enforcing the cut-off values such as by limiting the observation distribution.

6.2 Further research

This paper was limited to investigating the performance of the methods compared to each other As such, there is room to compare these techniques to other methods for solving the SLAM problem such as graph-based SLAM methods or the FastSLAM algorithm [16]. In this paper, we explored exclusively offline implementations. However, in many practical applications of SLAM systems we require online smoothing and estimation. Both the forward-only FFBSm and the PaRIS algorithm are directly applicable in this setting and an online-version of the FLS algorithm can also be constructed. Hence performance of these algorithms in an online setting is a possible topic for further research.

Bibliography

[1] T. Bailey, J. Nieto, and E. Nebot. "Consistency of the FastSLAM algorithm". In: *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* May 2006, pp. 424–429. DOI: 10.1109/ROBOT.2006.1641748.

[2] T. Bailey et al. "Consistency of the EKF-SLAM Algorithm". In: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Oct. 2006, pp. 3562–3568. DOI: 10.1109/IROS.2006.281644.

[3] C. Cadena et al. "Past, Present, and Future of Simultaneous Localization And Mapping: Towards the Robust-Perception Age". In: *IEEE Transactions on Robotics* 32.6 (2016), pp. 1309–1332.

[4] Olivier Cappé and Eric Moulines. "On-line expectation–maximization algorithm for latent data models". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 71.3 (2009), pp. 593–613. ISSN: 1467-9868. DOI: 10.1111/j.1467-9868.2009.00698.x. URL: http://dx.doi.org/10.1111/j.1467-9868.2009.00698.x.

[5] Randal Douc et al. "SEQUENTIAL MONTE CARLO SMOOTHING FOR GENERAL STATE SPACE HIDDEN MARKOV MODELS". In: *The Annals of Applied Probability* 21.6 (2011), pp. 2109–2145. ISSN: 10505164. URL: http://www.jstor.org/stable/41408080.

[6] Arnaud Doucet, Nando de Freitas, and Neil Gordon. "An Introduction to Sequential Monte Carlo Methods". In: *Sequential Monte Carlo Methods in Practice*. Ed. by Arnaud Doucet, Nando de Freitas, and Neil Gordon. New York, NY: Springer New York, 2001, pp. 178–195. ISBN: 978-1-4757-3437-9. DOI: 10.1007/978-1-4757-3437-9_1. URL: http://dx.doi.org/10.1007/978-1-4757-3437-9_1.

[7] Arnaud Doucet, Simon Godsill, and Christophe Andrieu. "On sequential Monte Carlo sampling methods for Bayesian filtering". In: *Statistics and Computing* 10.3 (July 2000), pp. 197–208. ISSN: 1573-1375. DOI: 10.1023/A:1008935410038. URL: http://dx.doi.org/10.1023/A:1008935410038.

[8] H. Durrant-Whyte and T. Bailey. "Simultaneous localization and mapping: part I". In: *IEEE Robotics Automation Magazine* 13.2 (June 2006), pp. 99–110. ISSN: 1070-9932. DOI: 10.1109/MRA.2006.1638022.

[9] Gunnar Englund. *Datorintensiva metoder i matematisk statistik*. KTH Royal Institute of Technology, Department of Mathmatics, pp. 256–258.

[10] Simon J Godsill, Arnaud Doucet, and Mike West. "Monte Carlo Smoothing for Nonlinear Time Series". In: *Journal of the American Statistical Association* 99.465 (2004), pp. 156–168. DOI: 10.1198/016214504000000151. eprint: http://dx.doi.org/10.1198/016214504000000151. URL: http://dx.doi.org/10.1198/016214504000000151.

[11] N.J. Gordon, D.J. Salmond, and A.F.M. Smith. "Novel approach to nonlinear/non-Gaussian Bayesian state estimation". English. In: *IEEE Proceedings F (Radar and Signal Processing)* 140 (2 Apr. 1993), 107–113(6). ISSN: 0956-375X. URL: http://digital-library.theiet.org/content/journals/10.1049/ip-f-2.1993.0015.

[12] M. Kaess, A. Ranganathan, and F. Dellaert. "iSAM: Incremental Smoothing and Mapping". In: *IEEE Transactions on Robotics* 24.6 (Dec. 2008), pp. 1365–1378. ISSN: 1552-3098. DOI: 10.1109/TRO.2008.2006706.

[13] R. E. Kalman. "A New Approach to Linear Filtering and Prediction Problems". In: *Journal of Basic Engineering* 82.1 (1960), pp. 35–45. DOI: 10.1115/1.3662552. URL: http://dx.doi.org/10.1115/1.3662552.

[14] Mike Klaas, Nando de Freitas, and Arnaud Doucet. "Toward Practical N2 Monte Carlo: The Marginal Particle Filter". In: *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*. UAI'05. Edinburgh, Scotland: AUAI Press, 2005, pp. 308–315. ISBN: 0-9749039-1-4. URL: http://dl.acm.org/citation.cfm?id=3020336.3020375.

[15] Augustine Kong, Jun S. Liu, and Wing Hung Wong. "Sequential Imputations and Bayesian Missing Data Problems". In: *Journal of the American Statistical Association* 89.425 (1994), pp. 278–288. ISSN: 01621459. URL: http://www.jstor.org/stable/2291224.

[16] Michael Montemerlo et al. "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem". In: *In Proceedings of the AAAI National Conference on Artificial Intelligence*. AAAI, 2002, pp. 593–598.

[17] P.D. Moral, A. Doucet, and S.S. Singh. *Forward smoothing using sequential Monte Carlo*. CUED/F-INFENG/TR. University of Cambridge, Department of Engineering, 2009. URL: https://books.google.se/books?id=jgFxmwEACAAJ.

[18] Jimmy Olsson and Johan Westerborn. "Efficient particle-based online smoothing in general hidden Markov models: The PaRIS algorithm". In: *Bernoulli* 23.3 (Aug. 2017), pp. 1951–1996. DOI: 10.3150/16-BEJ801. URL: http://dx.doi.org/10.3150/16-BEJ801.

[19] Jimmy Olsson et al. "Sequential Monte Carlo smoothing with application to parameter estimation in nonlinear state space models". In: *Bernoulli* 14.1 (Feb. 2008), pp. 155–179. DOI: 10.3150/07-BEJ6150. URL: http://dx.doi.org/10.3150/07-BEJ6150.

[20] R. Smith, M. Self, and P. Cheeseman. "Estimating uncertain spatial relationships in robotics". In: *Proceedings. 1987 IEEE International Conference on Robotics and Automation*. Vol. 4. Mar. 1987, pp. 850–850. DOI: 10.1109/ROBOT.1987.1087846.

[21] Sebastian Thrun and Michael Montemerlo. "The GraphSLAM algorithm with applications to large-scale mapping of urban structures". In: *INTERNATIONAL JOURNAL ON ROBOTICS RESEARCH* 25.5 (2006), pp. 403–430.

Appendix A

Algorithms

This appendix presents in broad strokes the algorithms used in the simulations. To make this appendix more clear the following random variables and parameters are loosely specified.

- The set of state parameters $\boldsymbol{X}_{0:T} = \{\boldsymbol{X}_0, \boldsymbol{X}_1, \ldots, \boldsymbol{X}_T\}$
- The set of landmarks $\boldsymbol{m} = \{\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}_N\}$.
- The set of indexes of observed landmarks $\mathcal{A}_t, t \in \{0 : T\}$. Where if the landmark was observed at time t, then $i \in \mathcal{A}_t$ else $i \notin \mathcal{A}_t$.
- The set of landmark observations $\boldsymbol{Z}_{0:T} = \{\boldsymbol{Z}_{0,A_0}, \boldsymbol{Z}_{1,A_1}, \ldots, \boldsymbol{Z}_{T,\mathcal{A}_t}\}$ where $\boldsymbol{Z}_{t,\mathcal{A}_t} = (\boldsymbol{Z}_{t,i})_{i \in \mathcal{A}_t}$
- The set of control inputs $\boldsymbol{u}_{0:T} = \{\boldsymbol{u}_0, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_T\}$.

A.1 Sequential importance sampling (SISR)

Algorithm 2 SISR

1: Initialize all particles $\{\boldsymbol{x}_0^j\}_{1\leq j\leq M}$ and assign weights $\{\omega_0^j\}_{1\leq j\leq M}$
2: **for** each time step t **do**
3: **for** each particle j **do**
4: draw new particles $\tilde{\boldsymbol{x}}_t^j$ with probability $\{W_t^j\}_{1\leq j\leq M} = \dfrac{\omega_t^j}{\Omega_t}$
where $\Omega_t = \sum_{j=1}^{M} w_t^j$.
5: propagate by sampling $\boldsymbol{x}_{t+1}^i$ from $p(\boldsymbol{X}_{t+1}|\boldsymbol{X}_t = \tilde{\boldsymbol{x}}_t^j, \boldsymbol{u}_t)$
6: update weights $\omega_{t+1}^j = p(\boldsymbol{Z}_{t+1,A_{t+1}}|\boldsymbol{X}_{t+1} = \boldsymbol{x}_{t+1}^j, \boldsymbol{m})$
7: **end for**
8: estimate the state $\hat{\boldsymbol{x}}_t$ by

$$\hat{\boldsymbol{x}}_t = \sum_{j=1}^{M} W_t^j \boldsymbol{x}_t^j(k) \tag{A.1}$$

.
9: **end for**

The algorithm is initialized by drawing M particles $\{\boldsymbol{x}_0^j\}_{1\leq j\leq M}$ from the initial distribution $p(\boldsymbol{X}_0)$ and weighing the particles by $\omega_0^j = p(\boldsymbol{z}_0|\boldsymbol{x}_0^j, \boldsymbol{m})$. Apart from providing filter estimates of the state it also provides particles and associated weights which the smoothing algorithms are dependent upon.

A.2 Fixed lag smoothing (FLS)

Algorithm 3 FLS

1: Let $h(\Delta_T) = \min(t + \Delta_T, T)$ where Δ_T is an integer and a design parameter. Assume at time $t-1$ that the particle trajectories $\{\boldsymbol{x}^j_{0:h(\Delta_T)}\}_{1\leq j\leq M}$ and associated importance weights $\{\omega^j_{h(\Delta_T)}\}_{1\leq j\leq M}$ computed from the SISR algorithm are available.

2: compute the particle filter approximations $\{W^j_t, \boldsymbol{x}^j_t\}_{1\leq j\leq M}$ using SISR.

3: compute smoothed expectations of the underlying state

$$\hat{\boldsymbol{x}}_t \coloneqq \mathbb{E}\left[\boldsymbol{X}_t | \boldsymbol{Z}_{0:T}\right] \approx \frac{\sum_{j=1}^{M} \omega^j_{h(\Delta_T)} \boldsymbol{x}^j_{0:h(\Delta_T)}(t)}{\sum_{j=1}^{M} \omega^j_{h(\Delta_T)}} \tag{A.2}$$

where $\boldsymbol{x}^j_{0:h(\Delta_T)}(t)$ is the state at time t for the particle trajectory $\boldsymbol{x}^j_{0:h(\Delta_T)}$.

4: At any time t smoothed expectations of additive functionals can estimated using

$$\mathbb{E}\left[S_t(\boldsymbol{X}_{0:t}) | \boldsymbol{Z}_{0:t} = \boldsymbol{z}_{0:t}\right] \approx S_t(\hat{\boldsymbol{x}}_0, \hat{\boldsymbol{x}}_1, \ldots, \hat{\boldsymbol{x}}_t) \tag{A.3}$$

The FLS algorithm above is based on the simpler,but more memory demanding approach where the additive functionals are not computed for each iteration. Instead smoothed expectations of the underling state sequence are computed and stored. These are later used to directly compute smoothed estimates of the additive functionals.

A.3 Forward-only forward-filtering backward-smoothing (Forward-only FFBSm)

Algorithm 4 Forward-only FFBSm

1: Assume at time $t-1$ that particle filter approximations $\{W_{t-1}^j, \boldsymbol{x}_{t-1}^j\}_{1\leq j\leq M}$ of $p(d\boldsymbol{x}_{t-1}|\boldsymbol{z}_{0:t-1}, \boldsymbol{m}, \boldsymbol{u}_{0:t-1})$ and $\{\hat{\tau}_{t-1}^m(\boldsymbol{x}_{t-1}^j)\}_{1\leq j\leq M}$ of $\{\tau_{t-1}^m(\boldsymbol{x}_{t-1}^j)\}_{1\leq j\leq M}$ are available.
2: compute the particle filter approximations $\{W_t^j, \boldsymbol{x}_t^j\}_{1\leq j\leq M}$ of $p(d\boldsymbol{x}_t|\boldsymbol{z}_{0:t}, \boldsymbol{m}, \boldsymbol{u}_{0:t})$ using SISR.
3: **for** $j = 1 \to M$ **do**
4: compute

$$\hat{\tau}_t^m(\boldsymbol{x}_t^j) = \frac{\sum_{k=1}^M W_{t-1}^k p(\boldsymbol{X}_t^j|\boldsymbol{X}_{t-1}^k = \boldsymbol{x}_{t-1}^k, \boldsymbol{u}_t)\left\{\hat{\tau}_{t-1}^m(\boldsymbol{x}_{t-1}^k) + s_t(\boldsymbol{x}_{t-1}^k, \boldsymbol{x}_t^j)\right\}}{\sum_{k=1}^M W_{t-1}^k p(\boldsymbol{X}_t^j|\boldsymbol{X}_{t-1}^k = \boldsymbol{x}_{t-1}^k, \boldsymbol{u}_t)} \tag{A.4}$$

5: **end for**
6: estimate

$$\hat{S}_t^m = \sum_{j=1}^M W_t^j \hat{\tau}_t^m(\boldsymbol{x}_t^j) \tag{A.5}$$

The algorithm is initialized by setting $\{\tau_0^m(\boldsymbol{x}_0^j)\}_{1\leq j\leq M} = 0$. The computational complexity of this algorithm is this form is $\mathcal{O}(M^2)$.

A.4 Particle-based, rapid incremental smoother (PaRIS)

Algorithm 5 Accept-Reject for PaRIS

Assume at time $t-1$ that particle filter approximations $\{W_{t-1}^j, \boldsymbol{x}_{t-1}^j\}_{1\leq j\leq M}$ and $\{W_t^j, \boldsymbol{x}_t^j\}_{1\leq j\leq M}$ are available and that there exists $\epsilon_+ \in R_+^*$ such that $p(\boldsymbol{X}_t|\boldsymbol{X}_{t-1} = \boldsymbol{x}_{t-1}, \boldsymbol{u}_t) \leq \epsilon_+$ for all possible realizations of $\boldsymbol{X}_t, \boldsymbol{X}_{t-1}$ and values of $\boldsymbol{u}_t$.
for $j = 1 \to \tilde{M}$ **do**
  set $L \leftarrow [\![1, M]\!]$
  set tries $\leftarrow 0$
  while $L \neq \emptyset$ **and** tries $\leq \sqrt{M}$ **do**
    set $n \leftarrow \#L$
    draw $(I_1, \ldots, I_M) \sim p(\{w_{t-1}^i\}_{i=1}^M)$
    draw $(U_1, \ldots, U_M) \sim U(0, 1)$
    set $L_n \leftarrow \emptyset$
    for $k = 1 \to n$ **do**
      if $U_k \leq p(\boldsymbol{X}_t^{L(k)}|\boldsymbol{X}_{t-1}^{I_k} = \boldsymbol{x}_{t-1}^{I_k}, \boldsymbol{u}_t)/\epsilon_+$ **then**
        set $K_t^{L(k),j} \leftarrow I_k$
      else
        set $L_n \leftarrow L_n \cup \{L(k)\}$
      end if
    end for
    set $L \leftarrow L_n$
    set tries $\leftarrow$ tries $+ 1$
  end while
  if $L \neq \emptyset$ **do**
    set $n \leftarrow \#L$
    for $k = 1 \to n$ **do**
      draw $K_t^{L(k),j} \sim p(\{\Lambda_t^M(L(k), i)\}_{i=1}^M)$ directly
    end for
  end if
end for
return $\{K_t^{i,j} : (i, j) \in [\![1, M]\!] \times [\![1, \tilde{M}]\!]\}$

The accept-reject algorithm above utilizes a threshold mechanic in or-

der to avoid getting stuck. If the accept-reject sampling is not able to produce a sample in $\sqrt{M}$ attempts, the sample is instead drawn directly from the original distribution.

Algorithm 6 PaRIS

1: Assume at time $t-1$ that particle filter approximations $\{W_{t-1}^j, \boldsymbol{x}_{t-1}^j\}_{1\leq j\leq M}$ of $p(d\boldsymbol{x}_{t-1}|\boldsymbol{z}_{0:t-1}, \boldsymbol{m}, \boldsymbol{u}_{0:t-1})$ and $\{\hat{\tau}_{t-1}^m(\boldsymbol{x}_{t-1}^j)\}_{1\leq j\leq M}$ of $\{\tau_{t-1}^m(\boldsymbol{x}_{t-1}^j)\}_{1\leq j\leq M}$ are available.
2: compute the particle filter approximations $\{W_t^j, \boldsymbol{x}_t^j\}_{1\leq j\leq M}$ of $p(d\boldsymbol{x}_t|\boldsymbol{z}_{0:t}, \boldsymbol{m}, \boldsymbol{u}_{0:t})$.
3: **for** each particle j **do**
4: **for** $\eta = 1 \to \tilde{M}$
5: draw $K_{t+1}^{j,\eta} \sim p(\frac{W_{t-1}^k p(\boldsymbol{X}_t^j|\boldsymbol{X}_{t-1}^k=\boldsymbol{x}_{t-1}^k, \boldsymbol{u}_t)}{\sum_{l=1}^M W_{t-1}^l p(\boldsymbol{X}_t^j|\boldsymbol{X}_{t-1}^l=\boldsymbol{x}_{t-1}^l, \boldsymbol{u}_t)})$ using accept-reject sampling method presented in algorithm 5.
6: **end for**
7: compute

$$\hat{\tau}_t^m(\boldsymbol{x}_t^j) = \frac{\sum_{\eta=1}^{\tilde{M}} \left(\hat{\tau}_{t-1}^m(\boldsymbol{x}_{t-1}^{K_t^{j,\eta}}) + s_t(\boldsymbol{x}_{t-1}^{K_t^{j,\eta}}, \boldsymbol{x}_t^j)\right)}{\tilde{M}} \tag{A.6}$$

8: **end for**
9: estimate

$$\hat{S}_t^m = \sum_{j=1}^{M} W_t^j \hat{\tau}_t^m(\boldsymbol{x}_t^j) \tag{A.7}$$

The algorithm is initialized by setting $\{\tau_0^m(\boldsymbol{x}_0^j)\}_{1\leq j\leq M} = 0$. The computational complexity of this algorithm is this form is $\mathcal{O}(M)$.

A.5 Expectation-maximization (EM)

Algorithm 7 Expectation Maximization (EM)

1: Assume that at iteration k previous map estimate $\boldsymbol{m}^{k-1}$ and relevant sufficient statistics are available .
2: update the map estimate according to

$$\boldsymbol{m}^k = \arg\max_{\boldsymbol{m}} Q(\boldsymbol{m}|\boldsymbol{m}^{k-1}) \tag{A.8}$$

where

$$Q(\boldsymbol{m}|\boldsymbol{m}^{k-1}) = \mathbb{E}\left[\log(L(\boldsymbol{m}, \boldsymbol{u}_{0:T}|\boldsymbol{X}_{0:T}, \boldsymbol{Z}_{0:T}))|\boldsymbol{Z}_{0:T}, \boldsymbol{u}_{0:T}, \boldsymbol{m}^{\boldsymbol{k-1}}\right] \tag{A.9}$$

and $L(\boldsymbol{m}, \boldsymbol{u}_{0:T}|\boldsymbol{X}_{0:T}, \boldsymbol{Z}_{0:T}))$ is the complete data likelihood function.

TRITA -MAT-E 2017:52
ISRN -KTH/MAT/E--17/52--SE