Simultaneous Localization and Mapping Techniques

Andrew Hogue

Oral Exam: June 20, 2005

Contents

Chapter 1

Introduction

Simultaneous Localization and Mapping (SLAM) addresses two interelated problems in mobile robotics concurrently. The first is localization; answering the question “Where am I?” given knowledge of the environment. The second problem is mapping; answering the question “What does the world look like?” At first glance, solving these two problems concurrently appears intractable since *mapping* requires the solution to localization but solving *localization* requires the solution to mapping. To estimate a map, the robot receives sensor measurements, possibly relative distances to particular landmarks relative to its position. Thus in order to determine a map of the environment, these measurements must be converted into a globally consistent world frame thus requiring the robot location within the world frame. Similarly, estimating the robot’s position requires a map simply because it has to be defined in relation to some consistent world frame, the map. This ‘chicken-and-egg’ relationship can be addressed by thinking of the problem in terms of uncertainties or probabilities. Within a probabilistic framework the above questions are combined into the single question “Where am I likely to be in the most likely version of the world that I have sensed so far?” Answering this question involves estimating the map while simultaneously estimating the robot location. Absolute knowledge of the world is unrealistic in practice, thus uncertainty plays a key role and must be modeled appropriately. Developing a solution within a probabilistic framework provides a way to solve SLAM in the presence of noisy sensors and an uncertain world model.

This report provides a survey of techniques that have been used to solve SLAM, and an overview of open problems in this area of research. A short introduction to the necessary Bayesian framework is presented followed by a derivation of SLAM in probabilistic terms. Particle filters and FastSLAM are highly active research topics and are described in detail. The final section of this report presents a survey of some open problems and issues with existing SLAM solutions.

1.1 Why is SLAM necessary?

Robot autonomy is perhaps the ‘holy grail’ of mobile robotics. Developing fully automatic robot platforms to operate in dangerous environments such as active or abandoned mines, volcanos, underwater reefs, or even nuclear power plants, would be beneficial for humanity. Replacing humans in such dangerous situations is a common application of robotic technology since it allows humans to work in a safer environment. A commonality to such robotic applications is the necessity for the robot to collect sensorial data about its environment and present this material to a human-operator in an easy-to-understand manner. Types of information that might be collected include air temperature and quality, and the physical appearance of particular locations. If the air quality is insufficient or hazardous, unprotected humans should not be placed in this environment. The robot might present the data to the human as a map, or model of the environment along with a path representing the robot’s trajectory. Overlaying sensor information on the map allows a human to understand the sensor data more easily. Removing the human from dangerous environments has provided the impetus to solve the SLAM problem effectively and efficiently. The difficulty of manually constructing a highly accurate map has motivated the research community to develop autonomous and inexpensive solutions for robotic mapping.

Tracking the position and orientation (pose) of a robot has a mulititude of applications. The robot needs

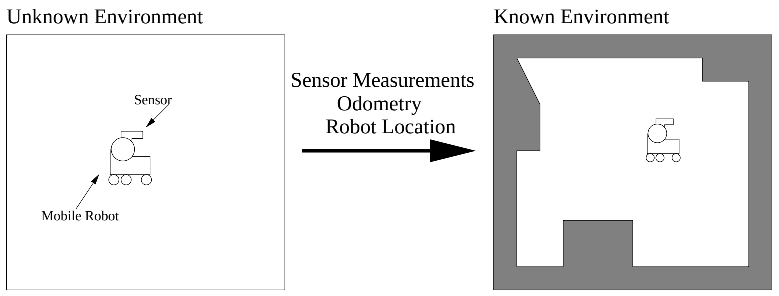
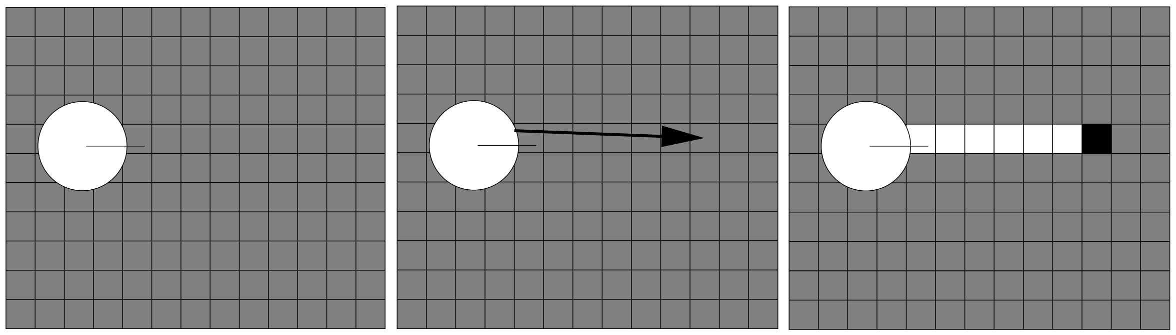


Figure 1.1: Mapping without Localization. The algorithm takes inputs from sensors and known robot pose and generates a detailed map of the environment.

to have some representation of its pose within the world for successful navigation. Absolute pose in world coordinates is not always necessary, however an accurate pose relative to the starting location is required for many applications. It becomes possible to navigate unknown terrain by enabling the robot to estimate its pose. This notion has been popularized with the Mars Rovers from NASA which are equipped with sensing technologies and algorithms capable of tracking the robot's pose relative to known landmarks. Navigation through unknown terrain presents many problems for robotic systems. The robot must be able to not only estimate its own pose but also the orientation, size and position of potential obstacles in its path. This is not an easy task. Also, in order to re-trace its steps if necessary the robot should keep track of its trajectory as well as other sensor readings (possibly in the form of a map) to allow accurate path planning. Mapping the environment allows the robot to *know where it is* relative to where it began its trek. It also enables the robot to recalibrate its sensors when it revisits the same area multiple times.

1.2 A Brief History of SLAM

When mapping and localization were introduced by researchers in the early '80s, the work focused on solving the two problems of mapping and localization independently. This section provides a brief overview of the literature and how it relates to current work in SLAM. An excellent review of the area can also be found in [75] and [76].



(a) Initial Occupancy Grid. Each cell is initialized with equal probability, i.e. Unknown state

(b) Robot Sensor reading. The range to a particular cell is sensed.

(c) Each cell is updated appropriately. The white cells are set to empty state (low probability), the black cell is a high probability or occupied state, the rest of the cells are still unknown and not updated.

Figure 1.2: Principle of the Occupancy Grid

1.2.1 Mapping without Localization

Robotic mapping is the problem of constructing an accurate map of the environment given accurate knowledge of the robot position and motion (see Figure 1.1). The early work in robotic mapping typically assumed that the robot location in the environment was known with 100% certainty and focused mainly on incorporating sensor measurements into different map representations of the environment. Metric maps, such as the occupancy grid introduced by Elfes and Moravec [55, 19, 18], allowed for the creation of geometrically accurate maps of the environment. In this approach, they represented the world as a fine-grained grid where each cell is in an occupied, empty, or unknown state. In his thesis [18], Elfes describes the process of updating the occupancy grid as going through the following stages

1. The sensor reading is converted into a probability distribution using the stochastic sensor model
2. Bayes rule is applied to update the cell probabilities
3. If desired, the Maximum a posteriori (MAP) decision rule is applied to label the cell as being either occupied, empty, or unknown

Each cell maintains the degree of certainty that it is occupied. As range data is gathered from the sonar sensors, empty and occupied areas in the grid representation are identified. The cells between the current robot location and the sensed range are set to an *empty* state due to the line-of-sight properties of sonar technology. Similarly, the cells at the sensed range are updated to an *occupied* state (see Figure 1.2). Most robotic tasks are capable of using the probabilistic occupancy grid representation, so it is rare that the occupied, empty, or unknown labels are used. This approach has received much success in the robotic community [3, 35, 4, 77, 49] and is still used at the core of new algorithms such as DP-SLAM [20, 21] which is discussed in more detail later in § 3.3.3. The topological map was another popular approach [43, 69, 7, 74]. Topological maps represent the environment as a graph with a connected set of nodes and edges. Each node represents significant locations in the environment and the set of edges represent traversable paths between nodes. In [43], a node is defined as a place that is locally distinctive within its immediate neighbourhood. For instance corners in laser scans are distinctive. They also define a *signature* of a distinctive place to be a subset of features that distinctively define the particular location. This signature is then used to recognize distinctive places as the robot moves. For example, a set of corners and planar surfaces could be a possible signature for a particular location. As the robot explores the world using some exploration strategy, distinctive nodes are identified and the signatures are added to the topological map. The map defines traversibility between the nodes and can be used as part of a high-level control scheme later on. A geometric map can be extracted if other data information is stored within the nodes and links, i.e. if the odometry is stored, then approximate registration between range information can be extracted in a post-processing scheme.

1.2.2 Localization without Mapping

Much work has also been done to estimate and maintain the robot position and orientation with an existing complete representation of the environment. In this situation, it is typically assumed that the map is known

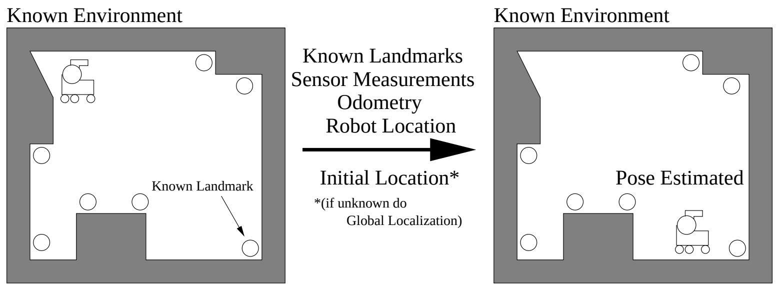


Figure 1.3: Localization without Mapping. The algorithm takes sensor measurements of known landmarks in the environment and estimates the robot's current pose.

with 100% certainty and is usually assumed to be static (for example, chairs and doors do not move or change state). These algorithms require an *a priori* map of the environment to define the reference frame and the structure of the world. Generally, localization algorithms take as input (i) a geometric map of the environment, (ii) initial estimate of the robot pose, (iii) odometry, and (iv) sensor readings. A successful algorithm uses these inputs and generates the best estimate of the robot pose within the environment (see Figure 1.3). Knowledge of the environment can be in many forms, either a full geometric map representation of the world, or the knowledge of particular landmarks and their locations. Typically, the sensors used can measure distance from the robot to any landmark in its line-of-sight.

Leonard and Durrant-Whyte "solved" SLAM using an Extended Kalman Filter (EKF) framework in [44]. They used an EKF for robot pose estimation through the observation of known geometric-beacons in the environment. Cox [8] used range data from sonars and matched sensor readings to an a priori map with an iterative process. This allowed the robot to align its version of the world with the known environment, thus determining its position and orientation. Map-matching (or scan matching) is a widely used iterative algorithm in robotics for global-localization. This technique was made popular by Lu and Milios in [48] and Scheile [66] used map matching on edge segments within an occupancy grid to self-localize.

More recent approaches to localization involve Monte-carlo sampling techniques, such as Particle-filtering made popular by Fox and Thrun in [24] and [79].

1.2.3 The Beginnings of Simultaneity

Chatila and Laumond in [6] first developed the principle of localizing a robot while creating a topological map of entities. This solution did not provide a probabilistic framework but rather the two problems were interleaved. Building on these principles, Smith, Self, and Cheeseman [70, 71, 72] along with Csorba [10] introduced the idea of solving both of the above problems, localization and mapping, simultaneously. They developed a probabilistic method of explicitly modeling the spatial-relationships between landmarks in an environment while simultaneously estimating the robot's pose. The map was represented as a set of landmark positions and a covariance matrix was used to model the uncertainty. The framework utilized a Kalman filter to estimate the mean position of each landmark from sensor readings. This constituted the first use of a probabilistic framework for estimating a *Stochastic Map* of the environment with modeled uncertainty, while simultaneously estimating the robot location. The introduction of probabilistic methods for robot localization and map creation stimulated a considerable amount of research in this area. The method was coined *SLAM* or *Simultaneous Localization And Mapping* by Durrant-Whyte and colleagues [45] and *CML* or *Concurrent Mapping and Localization* [78] by others. Csorba [10] examined the theoretical framework surrounding solutions to the SLAM problem. This work detailed how correlations arise between errors in the map estimates and the vehicle uncertainty which he argues are of fundamental importance in SLAM. Csorba proved theoretically that it is possible to build a map of an unknown environment accurately through the simultaneous estimation of the robot's pose and the map.

Since the early days of SLAM, a probabilistic approach has become the de-facto standard way of modeling the SLAM problem. The different ways in which the probabilistic density functions are represented constitute the differences in each approach. Many issues associated with the Kalman filtering approach have been identified and improved upon using Particle-filtering techniques and a theoretical analysis of the SLAM problem has also been performed. Probabalistic methods are fundamental to solving SLAM because of the inherent uncertainty in the sensor measurements and robot odometry. The next chapter introduces and explores the mathematical preliminaries necessary for solving SLAM in a probabilistic framework.

Chapter 2

Mathematical Preliminaries

Dealing with uncertainty is fundamental to any SLAM algorithm. Sensors never provide perfect data. Sensor measurements are corrupted by noise and have intrinsic characteristics that must be appropriately modeled. For instance, laser range sensors provide highly accurate data in one direction for a single point in space, whereas a sonar sensor provides range data of a point within a cone of uncertainty. Odometry sensors provide a good estimate of robot motion, however wheel slippage and external forces in the environment can cause the estimate to drift increasing the amount of uncertainty in the robot pose. Different sensors have different accuracy and noise characteristics that must be taken into account. This can be accomplished by using stochastic, or probabilistic, models and Bayesian inference mechanisms.

The following sections introduce the reader to probability and probabilistic estimation techniques in the context of SLAM. The notion of a random variable, prior and conditional probabilities, and functions to manipulate entire distributions are described. Bayes rule is discussed and the derivation of Bayesian estimation through recursive filtering leads into the explanation of the Kalman filter; a popular tool used in the SLAM literature. Particle filtering techniques for robotic mapping have gained popularity in favour of Kalman filtering approaches and their implementation and properties are discussed.

Note that the following sections provide a review of concepts and techniques that are important for

SLAM. An exhaustive review of Probability theory is beyond the scope of this report and the interested reader should look at [22, 73]. See Kalman's seminal paper [41] or Welch and Bishop's introductory paper [83] (available electronically at http://www.cs.unc.edu/∼welch/kalman) for a complete theoretical and practical explanation of Kalman filtering. Also, see [15] for a good introduction to Monte Carlo and Particle Filtering techniques.

2.1 Probability

The notion of probability began in the sixteenth century by mathematicians Blaise Pascal and Pierre de Fermat. Since then, probability theory has formed a large area of mathematics used in many applications including game theory, quantum mechanics, computer vision and robotics. As discussed previously, the use of a Stochastic framework to address the SLAM problem is of foremost importance. It is with the use of probabilitistic techniques that simultaneously estimating a map of the environment and the pose of the robot is possible. Thus, the notion of probability must be defined. Much of the discussion in this section can be found in any introductory textbook on probability and statistics such as [73] or [28]. This section will briefly introduce the constructs necessary to derive the SLAM problem from a stochastic point of view.

2.1.1 Discrete Probability Distributions

In order to discuss probabilities, a Random Variable (RV) must be defined. Imagine that you roll a die. The possible outcomes of rolling the die are 1,2,3,4,5,6 depending on which side of the die turns up. A mathematical expression for expressing this is possible by representing each outcome as one of $X_1, X_2, X_3, X_4, X_5, X_6$ or $X_i, i = 1 \ldots 6$. Each X_i is called a *Random Variable* and represents a particular outcome of an experiment. More importantly, we need a function $m(X_i)$ of the variable that lets us assign probabilities to each possible outcome. This function is called the *distribution function*. For the example of a roll of a die, we would assign the same value to each of the possible outcomes of the roll. Then, the distribution would be uniform and each outcome would have a probability of $\frac{1}{6}$. Now, you could ask the question "What is the probability of rolling a number less than or equal to 4?". This could be computed using the notation

$$P(X \leq 4) = P(X_1) + P(X_2) + P(X_3) + P(X_4) = \frac{2}{3} \tag{2.1}$$

meaning that the probability that a number less than 4 will be the outcome of the experiment of rolling a single die would be two-thirds.

More formally, the set of possible outcomes is called the Sample Space denoted as Ω. Any experiment must produce an outcome that is contained within Ω. In a discrete sense, we consider only experiments that have finitely many possible outcomes, or that Ω is finite.

Given an experiment with an outcome that is random (depends upon chance), the outcome of the experiment is denoted with a capital letter such as X called a random variable. The sample space Ω is the set of all possible outcomes of the experiement. Another definition that is used is the idea of an *event* which is any subset of outcomes of Ω. Wikipedia[1] defines a Random Variable as:

> A random variable can be thought of as the numeric result of operating a non-deterministic mechanism or performing a non-deterministic experiment to generate a random result. For example, a random variable can be used to describe the process of rolling a fair die and the possible outcomes 1, 2, 3, 4, 5, 6 . Another random variable might describe the possible outcomes of picking a random person and measuring their height.

and it is important to note that a RV is not a variable in the traditional sense of the word. You cannot define the value of the random variable, it is a statistical construct used to denote the possible outcomes

[1]http://en.wikipedia.org/wiki/Random_variable

of some event in terms of real numbers. More formally, the definition of a random variable paraphrased from PlanetMath[2] is

> Let $\mathcal{A}$ be a σ-algebra and Ω be the sample space or space of events related to an experiment we wish to discuss. A function $X : (\Omega, \mathcal{A}) \rightarrow \Re$ is a random variable if for each subset $A_r = \{\omega : X(\omega) \leq r\}, r \in \Re$ the condition $A_r \in \mathcal{A}$ is satisfied. A random variable X is *discrete* if the set $\{X(\omega) : \omega \in \Omega\}$ is finite or countable. A random variable X is continuous if it has a cumulative distribution function which is continuous.

A distribution function for the random variable X is a real-valued function $m(\cdot)$ whose domain is Ω and satisfies

$$m(\omega) \geq 0 \text{ , } \forall \omega \in \Omega \tag{2.2}$$

$$\sum_{w \in \Omega} m(\omega) = 1 \tag{2.3}$$

For any event E (a subset of Ω), the probability of E, $P(E)$, is defined as

$$P(E) = \sum_{\omega \in E} m(\omega) \tag{2.4}$$

The properties or discrete probability rules can be stated as

1. $P(E) \geq 0$ for every $E \in \Omega$
2. $P(\Omega) = 1$
3. $P(E) \leq P(F)$ where $E \subset F \subset \Omega$
4. $P(A \cup B) = P(A) + P(B)$ if and only if A and B are disjoint subsets of Ω

[2]http://planetmath.org/?op=getobj&from=objects&id=485

5. $P(\bar{A}) = 1 - P(A)$ for every $A \subset \Omega$

A uniform distribution on sample space Ω with n elements is the function

$$m(\omega) = \frac{1}{n}\,,\ \forall \omega \in \Omega \tag{2.5}$$

2.1.2 Continuous Probability Distributions

For many applications, such as SLAM, we require that the experiments can take on an entire range of possible outcomes as opposed to a particular set of discrete elements. Generally, if the sample space Ω is a general subset of $\mathcal{R}^n$ then Ω is called a *continuous sample space*. Thus, any random variable X representing the outcome of an experiment defined over Ω is considered a *Continuous Random Variable*.

It is interesting to note that as before, the probability of a particular event occurring is the sum of values in a particular range defined over some real-valued function. In a continuous space, this sum becomes an integral and the function being integrated is called the *density function* since it represents the density of probabilities over the sample space. The defining property of a density function is that the area under the curve and over an interval corresponds to a probability of some event occurring.

Thus, the probabilistic density function (pdf), $f(x)$ can be defined as a real-valued function that satisfies

$$P(a \le X \le b) = \int_a^b f(x)dx\,,\ \forall a, b \in \mathbf{R} \tag{2.6}$$

Note that the density function contains all probability information since the probability of any event can be computed from it but that the value of $f(x)$ for outcome x is not the probability of x occurring and generally the value of the density for the function is not a probability at all. The density function however does allow us to determine which events are more likely to occur. Where the density is large, the events are more likely to occur, and where the density is smaller, the events are less likely to occur.

Another important function for using probabilities is the *Cumulative Distribution Function* or commonly just called the *distribution function* and are closely related to the density functions above.

Let X be a real-valued continuous random variable, then the distribution function is defined as

$$F_X(x) = P(X \leq x) \tag{2.7}$$

or more concretely if $f(x)$ is the density function of random variable X then

$$F(x) = \int_{-\infty}^{x} f(t)dt \tag{2.8}$$

is the cumulative distribution function of X. This by definition means that

$$\frac{d}{dx}F_X(x) = f_X(x) \tag{2.9}$$

It is useful to note the following properties of the cumulative distribution function

- Boundary cases: $F_X(\infty) = 1, F_X(-\infty) = 0$
- $F_X(x)$ is a Non-decreasing function: $x_1 \leq x_2 \rightarrow F_X(x_1) \leq F_X(x_2)$
- $F_X(x)$ is continuous, namely that $F_X(x) = \lim_{\varepsilon \to 0} F_X(x+\varepsilon)$ where $\varepsilon > 0$
- The value at any point in the distribution is the cumulative sum of all previous probabilities,

$$F_X(x) = P(X < x) = \int_{-\infty}^{x} f_X(x)dx \tag{2.10}$$

- The probability of the random variable taking on a value in a particular range $[a,b]$ is

$$F_X([a,b]) = P(a \leq X \leq b) \int_a^b f_X(x)dx \tag{2.11}$$

2.1.3 Manipulating Probabilities

Now that the notion of a probability is given, we can discuss how to manipulate them and the very important idea of joint and conditional probability.

The **joint** probability of two events describes the probability of both events occuring and is denoted $P(X,Y)$ or $P(X \cap Y)$. If X and Y are independent events ($X \cap Y = \emptyset$) then the joint probability is $P(X,Y) = P(X)P(Y)$.

The **conditional** probability is the probability of one event occurring given knowledge that another event has already occurred. This is denoted $P(X|Y)$ and is the probability of event X conditional on event Y happening. If both events are independent of each other then the conditional probability is $P(X|Y) = P(X)$. If X and Y are not independent events, then the joint probability can be defined using the notion of conditional probabilities as $P(X,Y) = P(X|Y)P(Y)$ or equivalently $P(X,Y) = P(Y|X)P(X)$.

The theorem of total probability is defined in terms of marginal probabilities for Discrete random variables

$$\begin{aligned} P(A,C) &= P(A|B_1,C)P(B_1,C) + P(A|B_2,C)p(B_2,C) + \cdots + P(A|B_n,C) && (2.12) \\ &= \sum_{i=0}^{n} P(A|B_i,C)P(B_i,C) \text{ , where } \cup_i B_i = \Omega \text{ , } B_i, i = 1 \ldots n \text{ partition } \Omega && (2.13) \end{aligned}$$

Generally, given two random variables, X and Y, we define the notion of conditional probability as $P(X|Y)$ meaning that we wish to compute the probability of X with the knowledge of Y's state. Following from the definition of joint probability, the following are equivalent

$$P(X \cap Y) = P(X|Y)P(Y) \tag{2.14}$$

$$P(X \cap Y) = P(Y|X)P(X) \tag{2.15}$$

the conditional probability term can be solved for and the equation is rearranged to form

$$P(X|Y) = \frac{P(X \cap Y)}{P(Y)}\ , P(Y) \neq 0 \tag{2.16}$$

$$P(Y|X) = \frac{P(X \cap Y)}{P(X)}\ , P(X) \neq 0 \tag{2.17}$$

Note that $P(X \cap Y)$ is also written as $P(X,Y)$.

The equation above forms the basis of Bayes for discrete events rule[22, 73].

$$P(X,Y) = P(X|Y)P(Y) = P(Y|X)P(X) \tag{2.18}$$

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)}\ , P(Y) \neq 0 \tag{2.19}$$

The denomenator can be re-written using the marginal distribution of Y over the variable X, then Bayes rule becomes

$$P(X_j|Y) = \frac{P(Y|X_j)P(X_j)}{\sum\limits_{i=0}^{n} P(Y|X_i)P(X_i)} \tag{2.20}$$

Similarly, Bayes rule can be defined for over continuous probability densities. Let x, y denote random variables and $f(x)$ be a density function. Then Bayes rule is described as

$$f(x|y) = \frac{f(y|x)f(x)}{f(y)} \tag{2.21}$$

and using the total probability theorem gives the following form

$$f(x|y) = \frac{f(y|x)f(x)}{\int\limits_{-\infty}^{+\infty} f(y|x)f(x)dx} \tag{2.22}$$

The probability $P(X_j)$ in Equation 2.20 or $f(x)$ in Equation **??** is known as the **prior** probability (or a priori density) of the event X occurring. A prior probability represents the degree of certainty in a random variable before any evidence is gathered about the state of that variable. If no knowledge is available about the process being estimated, then typically the variable is assigned a uniform prior probability; each event is assigned an equal prior probability.

Bayes rule (also known as Bayes Theorem) is the basis of many statistical inference techniques (also known as Bayesian Inference). It provides a method for calculating a conditional probability given some evidence about the world and the prior probability distributions. The conditional probability after applying Bayes rule is also called a **posterior**, or *a posteriori* probability. For robot localization, Bayes rule supplies an inference mechanism for calculating a probability of the robot location given information about the world in the form of sensor measurements.

For the remainder of this report, we will be using the notation of capital $P(X)$ as a discrete probability of the random variable X and we will use lowercase $p(x)$ as the continuous probability density function of the random variable x.

2.2 Expected Value and Moments

When large sets of values are to be considered, we are generally not interested in any particular value but would rather like to look at certain quantities that describe the distribution fully. This is also true for probabilities since if we had to evaluate each outcome of a continuous pdf, it would take an infinite amount of time to do so. Two such descriptive quantities are the *Expected value* and the *variance* of the distribution.

Estimating the *expected value* or *mean* of a probability density function is fundamental to using probability in any state estimation algorithm. The standard way of calculating the sample mean of N numbers is statisically

$$\bar{x} = \frac{1}{N}\sum_{i=0}^{N} x_i \tag{2.23}$$

For probability density functions, each value x has an associated probability density $p(x)$. With no prior knowledge about the pdf of a random variable, a uniform pdf is typically assumed. That is, the random variable x can take on any value in the sample space with equal probability. In this case, the expected value is simply the mean (μ) of the distribution. In a situation where we have obtained knowledge about the probability distribution, we need to weight each event with its associated probability. The mean can be defined as the **Expected value** of the random variable. For continous domains, the expected value is

$$E[x] = \int_{-\infty}^{+\infty} x p(x) dx \tag{2.24}$$

and for discrete random variables, the mean is denoted

$$E[x] = \sum_{x \in \Omega} x P(x) dx \tag{2.25}$$

The expected value is also sometimes called the first moment of the distribution. Every random variable can be described by a set of moments that describe the behaviour of the pdf. A moment represents a particular characteristic of the density function. Given enough knowledge of these moments, the pdf can be fully reconstructed.

In general, the r-th moment, ξ_r, of a probability distribution function is given by

$$\xi_r \triangleq \sum_i x_i^r P(x_i) \tag{2.26}$$

and its central moment (centered around the mean) can be described as

$$E\left[[x - E[x]]^r\right] \triangleq \sum_i (x_i - E[x])^r P(x_i) \tag{2.27}$$

These notions are expressed similarly for continuous domains

$$E\left[[x - E[x]]^r\right] = \int_{-\infty}^{+\infty} (x - E[x])^r p(x) dx \tag{2.28}$$

For instance the equation of the normal (Gaussian) distribution is

$$N(x;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \tag{2.29}$$

This can be fully represented by its first moment, the mean, and its second central moment, the variance. More generally, a Gaussian with dimensionality d is represented by its mean vector $\vec{\mu}$ and a covariance matrix Σ.

$$N_d(\vec{x};\vec{\mu},\Sigma) = \frac{1}{\sqrt{(2\pi)^d}} |\Sigma|^{-\frac{1}{2}} e^{-\frac{1}{2}(\vec{x}-\vec{\mu})^T \Sigma^{-1} (\vec{x}-\vec{\mu})} \tag{2.30}$$

To manipulate expected values of random variables, there are only a few simple rules that have to be used

1. $E[c] = c$ where c is a constant.

2. $E[cx] = cE[x]$ where c is a constant.

3. $E[E[x]] = E[x]$.

4. $E[x+y] = E[x] + E[y]$.

5. $E[xy] = E[x]E[y]$ if and only if x and y are independent random variables.

2.3 Bayes Filters

Bayes Rule provides a framework for inferring a new probability density function as new evidence in the form of sensor measurements becomes available. A Bayes filter is a probabilistic tool for computing a posterior probability density in a computationally tractable form. Bayesian filtering is the basis of all successful SLAM algorithms and thus must be introduced appropriately. First the notion of a Markov-Chain process and the Markov assumption is introduced which leads into the explanation of Recursive Bayesian Filtering. This type of filter is implemented, with some assumptions about the probability distributions, in the Kalman filtering framework which is discussed in detail. Finally, the Particle-filter is an important recursive Bayes filtering technique that has recently become popular in the robotics community.

2.3.1 Markov-Chain Processes

A Markov-chain process is defined In a Markov-chain process the current state is dependent only upon the previous state [42, 39, 15]. This allows for a huge reduction in complexity, both in time and space, and provides a computationally feasible way to approach many problems. This can be described notationally as

$$p(x_{t+1}|x_0, x_1, \ldots, x_{t-1}, x_t) = p(x_{t+1}|x_t) \tag{2.31}$$

A consequence of the Markov-chain assumption is that the probability of the variable at time $t+1$ can be computed as

$$p(x_{t+1}) = \int p(x_{t+1}|x_t)p(x_t)dx_t \tag{2.32}$$

2.3.2 Recursive Bayesian Filtering

Using Bayesian updating rules at each step of a Markov-chain process is called *recursive Bayesian filtering*. Recursive Bayesian filtering addresses the problem of estimating a random vector using sensor measurements or other data pertaining to the state. Basically, it estimates the posterior probability density function of the state conditioned by the data collected so far. In the robot localization scenario, we are given some measurements of the world (i.e. range from the robot to known landmarks) as the robot moves and we wish to estimate the pose of the robot. Thus, we need to estimate

$$p(s_t|z^t, u^t)\ ,\ z^t = \{z_1, z_2, \ldots, z_t\}\ ,\ u^t = \{u_1, u_2, \ldots, u_t\} \tag{2.33}$$

where s_t is the robot's pose at time t, z^t are the measurements until now, and u^t are the control inputs (i.e. commanded motion of the robot) until now. This can be described using Bayes rule as

$$p(s_t|z^t, u^t) = \frac{p(z_t|s_t, z^{t-1}, u^t)p(s_t|z^{t-1}, u^t)}{\int p(z_t|s_t, z^{t-1}, u^t)p(s_t|z^{t-1}, u^t)ds_t} = \eta^{-1}p(z_t|s_t, z^{t-1}, u^t)p(s_t|z^{t-1}, u^t) \tag{2.34}$$

where $\eta^{-1} = \int p(z_t|s_t, z^{t-1}, u^t)p(s_t|z^{t-1}, u^t)ds_t$ is usually viewed as a normalizing constant that ensures the result is a probability density function.

The first term, $p(z_t|s_t, z^{t-1}, u^t)$, can be simplified by assuming that each measurement is independent of each other and only dependent upon the robot pose, or that $p(z_t|s_t, z^{t-1}, u^t) = p(z_t|s_t)$.

$p(z_t|s_t)$ is known as the probabilistic measurement model. The second term, $p(s_t|z^{t-1}, u^t)$, is the prior pdf of the robot pose given all measurements and control inputs until now. Since the robot pose is dependent only upon the pose and control inputs but not on the actual measurements (i.e. the measurement itself does not influence how the pose changes over time), then $p(s_t|z^{t-1}, u^t) = p(s_t|u^t)$ and Equation 2.34 may be simplified to

$$p(s_t|z^t, u^t) = \eta \underbrace{p(z_t|s_t)}_{\text{Measurement Model}} \underbrace{p(s_t|u^t)}_{\text{Pose Prior}} \tag{2.35}$$

Using the Markov assumption (see § 2.3.1), we marginalize the pose prior distribution on the previous robot position, namely we can write $p(s_t|u^t) = \int p(s_t|s_{t-1}, u_t)p(s_{t-1}|u^{t-1})ds_{t-1}$ to get

$$p(s_t|z^t, u^t) = \eta p(z_t|s_t) \int \underbrace{p(s_t|s_{t-1}, u_t)}_{\text{Motion Model}} p(s_{t-1}|u^{t-1})ds_{t-1} \tag{2.36}$$

The simplest way to think about a recursive Bayesian filter is by evaluating Equation 2.36 in two stages. In order to estimate the posterior density given sensor measurements, first a prediction stage is performed which estimates the pdf $p^-(s_t|u^t)$ defined as

$$p^-(s_t|u^t) = \int p(s_t|s_{t-1}, u_t)p(s_{t-1}|u^{t-1})ds_{t-1} \tag{2.37}$$

This describes a process of predicting what the state should be at the current time step given that we know how the system evolves and the value of the state the previous time step. The term $p(s_t|s_{t-1}, u_t)$ represents the system motion model. In a moving robot example, this could be a linear function that uses the current velocity of the robot to predict where it should be, or the function could very well be a nonlinear function that integrates angular and linear velocities to predict the next pose. The term $p(s_{t-1}|u^{t-1})$ is the probability of being in the previous state s_{t-1} given the control inputs until time $t-1$.

The second stage of recursive Bayesian filtering is to incorporate the measurement and probabilistic measurement model into the estimate of the posterior pdf. This stage is typically known as the measurement update and is described in notation as

$$p^*(s_t|u^t) \leftarrow p(z_t|s_t)p^-(s_t|u^t) \tag{2.38}$$

p^* is then re-normalized to become a proper pdf

$$p(s_t|u^t) = \eta_t p^*(s_t|u^t) \tag{2.39}$$

$$p^*(s_t|u^t) = \int p(s_t|u^t)p(s_t)ds_t \tag{2.40}$$

This equation describes how to correct the probability density function for the current time $p(s_t|u^t)$, using the predicted density function $p^-(s_t|u^t)$ and the measurement data.

Of course, the above formulation provides a very generalized probabilistic framework. To implement such a scheme requires the specification of a system dynamics model $p(s_t|s_{t-1},u_t)$, a sensor model $p(z_t|s_t)$, and the representation of the density function $p(s_t|u^t)$. This is non-trivial and highly problem specific. It is in the differences in the representation of these quantities that differentiate the various Bayesian filtering schemes in the literature.

2.4 Kalman Filtering

Perhaps the most common application of Bayesian filtering (Equation 2.36) is the Kalman filter [41, 83]. This is a recursive Bayes filter for linear systems. The assumption within the Kalman filter is that the underlying probability density function for each of the above distributions can be modeled using a Gaussian distribution. This provides a simple way of estimating the pdf since a Gaussian can be fully represented in closed-form by its variance, σ^2, and mean, μ. In a multi-dimensional situation, this is easily generalized by using a mean vector $\vec{\mu}$ and a covariance matrix P that represents how each element of the state vector varies with every other element.

The Kalman filter is used as a state or parameter estimator. It uses the Bayesian framework to update the mean and covariance of a Gaussian probability distribution. The state to be estimated, $\hat{x}$, is a multi-dimensional vector of random variables that we wish to estimate. The derivation of the Kalman filter equations is beyond the scope of this report but is included in the Appendix for the interested reader. The final Kalman filtering algorithm can be stated (see Table 2.1 for the notation used).

Time Update Step

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k \tag{2.41}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{2.42}$$

Measurement Update Step

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{2.43}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{2.44}$$

$$P_k = (I - K_k H)P_k^- \tag{2.45}$$

2.4.1 Extended Kalman Filtering

The linear Kalman filter formulated as above has been proven in [41], in a least squares sense, to be an optimal estimation filter for discrete linear systems. It is rare in computer vision, graphics and robotics, that the system dynamics of the process to be estimated is in fact linear. Most systems do not exhibit linear system dynamics. Adding a rotation term to the system dynamics (e.g. if the robot rotates in the plane)

Notation	Size	Description
$\hat{x}_k$	$n \times 1$	State vector of variables, estimated at time step k
z_k	$m \times 1$	Measurement vector at time step k
A	$n \times n$	System dynamics matrix
u_k	$l \times 1$	Optional control input at time step k
B	$n \times l$	Matrix that relates control input to the state
P_k	$n \times n$	State covariance matrix at time step k
Q	$n \times n$	Process noise covariance matrix
R	$m \times m$	Measurement noise covariance matrix
K	$n \times m$	Kalman gain matrix
H	$m \times n$	Matrix relating state to measurement
I	$n \times n$	Identity matrix
$\hat{x}_k^-$	$n \times 1$	Predicted state
P_k^-	$n \times n$	Predicted covariance

Table 2.1: The notation used in the form of the Kalman Filter discussed here. Note that n is the dimension of the state vector, m is the dimension of the measurement vector, and l is the dimension of the control input.

adds an inherent non-linearity to the system. Several different techniques have been employed to model the non-linearity in the Kalman filtering framework. The most popular implementations of Kalman filters for nonlinear systems are the extended Kalman filter (EKF) [83] and the Unscented Kalman filter (UKF) [40, 81].

The EKF assumes that the system dynamics model

$$\hat{x}_t = f_X(x_{t-1}, u_{t-1}, v_{t-1}) \tag{2.46}$$

is governed by a nonlinear process with additive zero-mean Gaussian noise. Similarly, the measurement model is also nonlinear

$$z_t = h(x_t, v_t) \tag{2.47}$$

The EKF solution to using nonlinear system and measurement models involves truncating the Taylor series of the nonlinear systems model. It linearizes the nonlinear model by using only the first two terms of the Taylor series.

The notation used in the extended Kalman filter is summarized in Table 2.2.

Notation	Size	Description
$\hat{x}_t$	$n \times 1$	State vector of variables, estimated at time step t
z_t	$m \times 1$	Measurement vector at time step t
u_t	$l \times 1$	Optional control input at time step t
P_t	$n \times n$	State covariance matrix at time step t
Q	$n \times n$	Process noise covariance matrix
R	$m \times m$	Measurement noise covariance matrix
K	$n \times m$	Kalman gain matrix
I	$n \times n$	Identity matrix
$f(\cdot)$		The motion model.
$h(\cdot)$		The measurement model.
A	$n \times n$	Jacobian Matrix of Partial derivatives of $f(\cdot)$ with respect to x
		$A_{[i,j]} = \frac{\partial f_{[i]}}{\partial x_{[j]}}(\hat{x}_{t-1}, u_t, 0)$
W	$n \times n$	Jacobian Matrix of Partial derivatives of $f(\cdot)$ with respect to noise w
		$W_{[i,j]} = \frac{\partial f_{[i]}}{\partial w_{[j]}}(\hat{x}_{t-1}, u_t, 0)$
H	$m \times n$	Jacobian Matrix of Partial derivatives of $h(\cdot)$ with respect to x
		$H_{[i,j]} = \frac{\partial h_{[i]}}{\partial x_{[j]}}(\hat{x}_t, 0)$
V	$m \times m$	Jacobian Matrix of Partial derivatives of $h(\cdot)$ with respect to noise v
		$H_{[i,j]} = \frac{\partial h_{[i]}}{\partial v_{[j]}}(\hat{x}_t, 0)$
$\hat{x}_t^-$	$n \times 1$	Predicted state
P_t^-	$n \times n$	Predicted Covariance

Table 2.2: The notation used in the form of the extended Kalman filter discussed here.

The *time-update* equations in this framework now become

$$\hat{x}_t^- = f(\hat{x}_{t-1}, u_t, 0) \tag{2.48}$$

$$P_t^- = A_t P_{t-1} A_t^T + W_t Q_{t-1} W_t^T \tag{2.49}$$

and the *measurement-update* equations are now

$$K_t = P_t^- H_t^T (H_t P_t^- H_t^T + V_t R_t V_t^T)^{-1} \tag{2.50}$$

$$\hat{x}_t = \hat{x}_{t-1}^- + K_t (z_t - h(\hat{x}_t^-, 0)) \tag{2.51}$$

$$P_t = (I - K_t H_t) P_t^- \tag{2.52}$$

Note that the Jacobians A, W, H, V are different at each timestep and must be recalculated. The Jacobian H_t is important since it propagates only the relevant component of the measurement information. It magnifies only the portion of the residual that affects the state.

Time Update Step
$\hat{x}_t^- = f(\hat{x}_{t-1}, u_t, 0)$ $P_t^- = A_t P_{t-1} A_t^T + W_t Q_{t-1} W_t^T$
Measurement Update Step
$K_t = P_t^- H_t^T (H_t P_t^- H_t^T + V_t R_t V_t^T)^{-1}$ $\hat{x}_t = \hat{x}_{t-1}^- + K_t (z_t - h(\hat{x}_t^-, 0))$ $P_t = (I - K_t H_t) P_t^-$

Table 2.3: Summary of the Extended Kalman Filter Equations

2.4.2 Extended Information Filters

The Extended Information Filter (EIF) [50, 80] is highly related to the EKF. It is essentially the EKF framework re-expressed in *information form*. The difference is that the EKF maintains a covariance matrix, while the EIF maintains the inverse of the covariance matrix, typically called the *information matrix* [58, 50]. The EKF formulation above represents the estimated posterior density function as a multivariate

Gaussian $N(x_t : \mu_t, \Sigma_t)$:

$$p(x_t|z^t, u^t) \propto e^{-\frac{1}{2}(x_t-\mu_t)^T \Sigma_t (x_t-\mu_t)} \tag{2.53}$$

$$= e^{-\frac{1}{2}x_t^T \Sigma_t^{-1} x_t + \mu_t^T \Sigma_t^{-1} x_t - \frac{1}{2}\mu_t^T \Sigma_t^{-1} \mu_t} \tag{2.54}$$

The last term does not contain the random variable x_t, thus it can be subsumed into the normalizing constant and we get

$$p(x_t|z^t, u^t) \propto e^{-\frac{1}{2}x_t^T \underbrace{\Sigma_t^{-1}}_{H_t} x_t + \underbrace{\mu_t^T \Sigma_t^{-1}}_{b_t} x_t} \tag{2.55}$$

$$H_t = \Sigma_t^{-1} \tag{2.56}$$

$$b_t = \mu_t^T \Sigma_t^{-1} \tag{2.57}$$

where H_t is defined to be the information matrix and b_t is the information vector. The EIF maintains the following posterior density function

$$p(x_t|z^t, u^t) \propto e^{-\frac{1}{2}x_t^T H_t x_t + b_t x_t} \tag{2.58}$$

Solutions in information space have advantages over state space when using multiple robots to explore the space. The total map or information of the state is simply the sum of the contributions from each robot and the cross information between robots is zero. This can make simplifications when designing SLAM-like algorithms. However, the main disadvantage of information space versus state-space algorithms is that the state is typically required for making decisions about the world. In the information formulation, the state is hidden through a matrix inversion. In SLAM, the information matrix represents links between map features, however if a geometric map is required, the information matrix must be inverted which incurs a high computational cost.

2.4.3 Unscented Kalman Filtering

The Unscented Kalman filter (UKF) [40, 81] addresses the issue of nonlinearity in the system model by using the non-linear system motion model directly instead of approximating it with Jacobians. It must be noted that the UKF overcomes the issues with nonlinearity of the Kalman filter, however it still assumes the models are fully represented by a unimodal Gaussian distribution (as oppposed to a mixture of Gaussians) with additive zero-mean Gaussian noise. At each time-update step, the UKF samples the state around the current mean estimate. The number of samples is deterministic and provably requires at most $2L+1$ samples where L is the dimension of the state vector. Each sample is temporally propagated with the actual non-linear dynamics model and a new mean is calculated. This new mean is considered to be the predicted state at that time in the future. Since no Jacobians are calculated as in the EKF, the computational cost of the UKF algorithm is of the same order as the EKF [81]. The UKF requires very few samples in comparison to Monte-Carlo Particle Filtering techniques (see next section) which typically requires hundreds if not thousands of particles to represent the distribution properly (depending on the complexity of the distribution). The UKF requires far fewer numbers of samples than a Particle filter since it samples in a deterministic fashion around the covariance of its current estimate correctly. It has been shown in [81] that the UKF approximation is accurate to the third order of the Taylor series for Gaussian motion models, and also accurate to the second order for non-Gaussian motions.

2.5 Particle Filtering

The Kalman filter is provably optimal in a least squares sense but only for a system with linear dynamics, linear measurement models, and both processes are adequately represented by a unimodal Gaussian density function and have additive zero-mean Gaussian noise [41]. This is restrictive as such constraints are unrealistic for many applications. The EKF and UKF addresses nonlinear system models, however these algorithms still are only applicable to problems with unimodal Gaussian distributions. When the system dynamics are highly nonlinear, or the sample rate is too low, the EKF gives a poor approximation to the dynamics and has a tendency to diverge. This is because the EKF provides only a first-order approximation to the nonlinear function being estimated. The UKF performs much better than the EKF and is accurate to the second order of nonlinearity in general but is accurate to the third order for Gaussian systems. The UKF is still not-applicable to systems that exhibit non-Gaussian noise or have multi-modal dynamics.

Particle filters have become a popular solution to the issues of Kalman filtering and have recently been applied to robot localization [25, 64]. At the core of the particle filtering algorithm is a collection of samples (particles) rather than assuming a closed-form representation of the pdf (i.e. representing the pdf via a Gaussian). Each individual particle represents a single sample of the pdf. If enough samples are used, the distribution is represented in full. For the interested reader, an excellent discussion and tutorial on Particle filters can be found in [65].

Particle-filtering algorithms represent a probability density through a weighted set of N samples; $p(x_t)$ is represented as $p(x_t) = \{x^{(i)}, w^{(i)}\}_{i=1\ldots N}$. Here, $x^{(i)}$ is a particle that samples the pdf $p(x_t)$ and $w^{(i)}$ is a weighting or *importance factor* denoting the importance of each particle (typically $\sum w^{(i)} = 1$). The initial set of samples of the pdf is drawn according to a uniform prior probability density; each particle is assigned an equal weighting of $\frac{1}{N}$.

The particle update procedure can be described by five stages.

1. Sample a state x_{t-1} from the pdf $p(x_{t-1})$ by randomly drawing a sample x from the set of particles according to the importance distribution (determined by the weights).

2. Use the sample x_{t-1} and the control input u_{t-1} to sample a new pdf, namely $p(x_t|x_{t-1}, u_{t-1})$. This

is described as predicting the motion of the vehicle using the motion model. The predicted pdf can be described by $p(x_t|x_{t-1},u_{t-1})p(x_{t-1})$.

3. Compute the new weight for the particle given a measurement and the predicted motion probability density. This is accomplished by computing the likelihood of the sample $x_{t-1}^{(i)}$ given the measurement z_t. Weight the particle by the non-normalized likelihood that the measurement could produce this sample using the measurement model $p(z_t|x_{t-1}^{(i)})$. These three steps must be performed on a per particle basis for each time step.

4. Re-normalize the weights $w^{(i)}$. This is done after all M particles have been sampled and re-weighted. After this step the particles represent a probability density function. Note that this procedure implements the posterior $p(x_t) = \eta p(z_t|x_t) \int p(x_t|x_{t-1},u_{t-1})p(x_{t-1})dx_{t-1}$.

5. Re-sample the posterior. The goal is to sample the posterior so that only the most important particles remain in the set for the next iteration.

There are many ways to implement the resampling stage but a widely used method is described by Liu in [46]. The process (replacing stages 4. and 5. above) is described in general as given a set of random samples $S_t = \{x^{(i)}, w^{(i)}\}_{i=1}^{N}$, the set S_t is treated as a discrete set of samples and another discrete set, $\tilde{S}_t = \{\tilde{x}^{(i)}, \tilde{w}^{(i)}\}_{i=1}^{N}$, is produced as follows

- For $i = 1 \ldots N$
  - compute $a^{(i)} = \sqrt{w^{(i)}}$
  - if $a^{(i)} \geq 1$
    * retain $k_i = \lfloor a^{(i)} \rfloor$ copies of the sample $x^{(i)}$
    * assign new weight $\tilde{w}^{(i)} = \frac{w^{(i)}}{k_i}$ to each copy
  - if $a^{(i)} < 1$

  - Remove the sample with probability proportional to $1 - a^{(i)}$
  - assign new weight $\tilde{w}^{(i)} = \frac{w^{(i)}}{a^{(i)}}$ if sample survives

Liu notes that resampling is necessary for several reasons. Notably that

1. resampling can prune away bad samples
2. resampling can produce multiple copies of good samples which allow the next sampling stage to produce better future samples.

Grisetti et al [29] demonstrate that the effective number of particles N_{eff} can be used as an indication on how well the particle system estimates the posterior pdf of the robot location. This is computed as

$$N_{eff} = \frac{1}{\sum\limits_{i=1}^{N} \left(w^i\right)^2} \tag{2.59}$$

and if N_{eff} falls below a specified threshold, then this measure indicates that the variance of the weights is high and thus the particles are dispersed too much and the set should be resampled. By resampling only when necessary, the particle filter has a much improved performance which is necessary for real-time applications.

2.6 Summary

In this section we have explored the necessary mathematics and mechanisms that make it possible to "solve" SLAM in a probabilistic framework. We first introduced the notion of probabilities which led into the idea of Bayesian estimation. Most algorithms work on the assumption that the current state estimate subsumes the history of its past states hence we introduced the Markov assumption. This was followed by a definition of Recursive estimation which brings these notions into a framework that can be subsequently implemented. The Kalman filter and some of its variants were discussed in detail which aids in the understanding of typical SLAM schemes (since the most common implementations of SLAM use Kalman filtering at their core). Finally, Particle-filtering was introduced which provides a way of estimating any generalized probability density function instead of the assumption that the density functions can be fully described by a Gaussian. The next chapter shows how this mathematics can be used to solve SLAM and how it has been done in the literature.

Chapter 3

Solutions to the SLAM problem

To date, the most common solution to SLAM has been with the use of an extended Kalman filter formulation. However, there has recently been a shift of interest towards the use of Monte-Carlo particle filtering techniques for computing the SLAM posterior to overcome the limitations of the EKF approach. These approaches constitute the most successful solutions to SLAM to date, however many other approaches exist. This chapter will first place SLAM into a probabilistic framework by using the tools described in the previous chapter to derive a Bayesian formulation. This formulation serves as the basis of the solutions to SLAM and the remainder of the chapter will discuss the common approaches to estimating the SLAM posterior.

3.1 Probabilistic derivation of SLAM

The goal of SLAM is to simulataneously localize a robot and determine an accurate map of the environment. This can be formalized in a Bayesian probabilistic framework by representing each of the robot position and map locations as probabilistic density functions. In essence, it is necessary to estimate the posterior density of maps Θ and poses s_t given that you know the observations $z^t = \{z_1, z_2, \ldots, z_t\}$, the control input $u^t = \{u_1, u_2, \ldots, u_t\}$ and data associations $n^t = \{n_1, n_2, \ldots, n_t\}$ which represent the mapping between map points in Θ and observations in z^t. The SLAM posterior, as defined in Montemerlo[52], is

$$p(s_t, \Theta | z^t, u^t, n^t) \tag{3.1}$$

The system dynamics motion model assumes the Markov property

$$p(s_t | s_{t-1}, u_t) \tag{3.2}$$

and the perceptual measurement model is

$$p(z_t | s_t, \Theta, n^t) \tag{3.3}$$

The derivation of the above posterior is detailed in [52] but it is worthwhile to repeat here in detail since it provides insight into the SLAM problem and how successful solutions are modeled.

The first step in deriving the SLAM posterior is to apply Bayes rule. Remembering Equation 2.20, the posterior probability can be defined as:

$$p(A|B,C) = \eta p(B|A,C) p(A,C) \tag{3.4}$$

and by using the substitutions $A = \{s_t, \Theta\}$, $B = z_t$, and $C = \{z^{t-1}, u^t, n^t\}$, Equation 3.1 becomes

$$p(s_t, \Theta | z^t, u^t, n^t) = \eta p(z_t | s_t, \Theta, z^{t-1}, u^t, n^t) p(s_t, \Theta | z^{t-1}, u^t, n^t) \tag{3.5}$$

The first term η is a normalizing value that ensures the posterior is a probability and is within the range

$[0,1]$. The second term can be simplified since each measurement is assumed to be independent of each other, thus z_t is independent of z_{t-1} and all other measurements. The measurement at this time is also independent of the control input. Thus, the term can be simplified to

$$p(z_t|s_t,\Theta,z^{t-1},u^t,n^t) \rightarrow p(z_t|s_t,\Theta,n^t) \tag{3.6}$$

and the SLAM posterior (Equation 3.5) can be re-written as

$$p(s_t,\Theta|z^t,u^t,n^t) = \eta \underbrace{p(z_t|s_t,\Theta,n^t)}_{\text{Measurement Model}} p(s_t,\Theta|z^{t-1},u^t,n^t) \tag{3.7}$$

The next step is to evaluate the last factor in Equation 3.7, $p(s_t,\Theta|z^{t-1},u^t,n^t)$. Using the theorem of total probability (Equation 2.13) along with the Markov Assumption enables us to condition the pdf on the previous state s_{t-1}

$$p(s_t,\Theta|z^{t-1},u^t,n^t) \rightarrow \int p(s_t,\Theta|s_{t-1},z^{t-1},u^t,n^t)p(s_{t-1}|z^{t-1},u^t,n^t)ds_{t-1} \tag{3.8}$$

and the SLAM posterior can now be re-written as

$$p(s_t,\Theta|z^t,u^t,n^t) = \eta p(z_t|s_t,\Theta,n^t)\int p(s_t,\Theta|s_{t-1},z^{t-1},u^t,n^t)p(s_{t-1}|z^{t-1},u^t,n^t)ds_{t-1} \tag{3.9}$$

If the map is allowed to be dynamic then a new map is possible at each time step as well which is denoted with the subscripted Θ_t. Thus, the SLAM posterior to be estimated would then be

$$p(s_t,\Theta_t|z^{t-1},u^t,n^t) = \eta p(z_t|s_t,\Theta,n^t)\iint p(s_t,\Theta_t|u_t,s_{t-1},\Theta_{t-1},n^t)p(s_{t-1},\Theta_{t-1}|z^{t-1},u^{t-1},n^{t-1})ds_{t-1}d\Theta_{t-1} \tag{3.10}$$

This is computationally very difficult, so the most common assumption in SLAM algorithms is a static map and Equation 3.9 is used.

Using the definition of conditional probability, the first term in the integral of Equation 3.9 can be split into

$$p(s_t,\Theta|s_{t-1},z^{t-1},u^t,n^t) \longrightarrow p(\Theta|s_{t-1},z^{t-1},u^t,n^t)p(s_t|s_{t-1},\Theta,z^{t-1},u^t,n^t) \tag{3.11}$$

and the SLAM posterior becomes

$$p(s_t,\Theta|z^t,u^t,n^t) = \eta p(z_t|s_t,\Theta,n^t)\int p(\Theta|s_{t-1},z^{t-1},u^t,n^t)p(s_t|s_{t-1},\Theta,z^{t-1},u^t,n^t)p(s_{t-1}|z^{t-1},u^t,n^t)ds_{t-1} \tag{3.12}$$

Using the Markov assumption, the robot pose at time t (s_t) can be assumed to be independent upon all variables except its own pose in the previous time step s_{t-1} and the control input given at time t (u_t). Thus, the following substitution is used

$$p(s_t|s_{t-1},\Theta,z^{t-1},u^t,n^t) \longrightarrow p(s_t|s_{t-1},u^t) \tag{3.13}$$

which is our motion model. Thus SLAM can be re-expressed as

$$p(s_t,\Theta|z^t,u^t,n^t) = \eta p(z_t|s_t,\Theta,n^t)\int \underbrace{p(s_t|s_{t-1},u^t)}_{\text{Motion Model}} p(\Theta|s_{t-1},z^{t-1},u^t,n^t)p(s_{t-1}|z^{t-1},u^t,n^t)ds_{t-1} \tag{3.14}$$

The final two terms in the integral can be combined into a single term using the definition of conditional probability

$$p(\Theta|s_{t-1},z^{t-1},u^t,n^t)p(s_{t-1}|z^{t-1},u^t,n^t) \longrightarrow p(s_{t-1},\Theta|z^{t-1},u^t,n^t) \tag{3.15}$$

Also, note that since $u^t = \{u_{0:t-1},u_t\}$ and $n^t = \{n_{0:t-1},n_t\}$ and u_t,n_t do not influence the previous measurement and pose, z_{t-1}, s_{t-1}, they can be eliminated. Finally the Bayesian filtering equation for SLAM can be written as

$$p(s_t, \Theta | z^t, u^t, n^t) = \eta \underbrace{p(z_t | s_t, \Theta, n^t)}_{\text{Measurement Model}} \int \underbrace{p(s_t | s_{t-1}, u^t)}_{\text{Motion Model}} p(s_{t-1}, \Theta | z^{t-1}, u^{t-1}, n^{t-1}) ds_{t-1} \tag{3.16}$$

As can be seen in Equation 3.16, it incorporates the perceptual model $p(z_t | s_t, \Theta, n^t)$ and the system dynamics motion model $p(s_t | s_{t-1}, u^t)$ of the system into a single expression.

It is in the evaluation of this posterior that differentiates the various SLAM algorithms. The following sections explore a number of these algorithms in detail.

3.2 Kalman filtering approaches

Kalman filtering approaches to SLAM go back to the seminal paper by Smith, Self, and Cheeseman [70, 71, 72]. They estimated the spatial-relationships between landmarks using an EKF framework. Much work using an EKF framework can be found in [45, 13, 17].

In order to implement Equation 3.16 in the Bayes filter framework, the representation of the measurement model, the motion model and the posterior pdf must be defined. In the EKF approach, all probability density functions are assumed to be Gaussian and can thus be fully represented with a multivariate Gaussian distribution.

$$p(s_t, \Theta | z^t, u^t, n^t) \sim N(\mu_t, \Sigma_t) \tag{3.17}$$

Since the Gaussian distribution can be fully reconstructed by its mean and covariance, this is what is required to be estimated. The mean vector μ_t of the Gaussian represents the current state of the world, i.e.

the current robot pose and the current positions of the map elements. Thus,

$$x = [s_t, \theta_0, \theta_1, \ldots, \theta_n] \tag{3.18}$$

where s_t represents the robot pose in vector form, and the map point positions are represented as $\Theta = \{\theta_0, \theta_1, \ldots, \theta_n\}$. An uncertain spatial relationship can be represented over the random variable x as $P(x) = p(x)dx$ where $p(x)$ is the probability density function that assigns a probability to x. To estimate this probability distribution function using a Gaussian requires us to define the estimated mean $\hat{x}$ and covariance $\Sigma(x)$ of the distribution, thus we define

$$\hat{x} = E[x] \tag{3.19}$$

$$\tilde{x} = x - \hat{x} \tag{3.20}$$

$$\Sigma(x) = E[\tilde{x}\tilde{x}^T] \tag{3.21}$$

The two steps in the Bayes filter, namely prediction followed by correction (observation), can be implemented by updating the mean and covariance of a Gaussian distribution.

The prediction stage predicts the new probability distribution given knowledge of the previous probability distribution, namely it needs to implement

$$\int p(s_t|s_{t-1}, u^t)p(s_{t-1}, \Theta|z^{t-1}, u^{t-1}, n^{t-1})ds_{t-1} \tag{3.22}$$

This is implemented through the Extended Kalman filter update equations for prediction or time update:

$$\hat{x}_t^- = f(\hat{x}_{t-1}, u_t, 0) \tag{3.23}$$

$$P_t^- = A_t P_{t-1} A_t^T + W_t Q_{t-1} W_t^T \tag{3.24}$$

Equivalently we can note the following:

$$\int p(s_t|s_{t-1}, u^t) p(s_{t-1}, \Theta|z^{t-1}, u^{t-1}, n^{t-1}) ds_{t-1} \sim N(\hat{x}_t^-, P_t^-) \tag{3.25}$$

Similarly, the next step in the Bayes filter is to perform a correction or measurement update, thus we need to represent the probability density function $p(z_t|s_t, \Theta, n^t)$, or rather provide a mechanism to update the Gaussian distribution using the current mean position, map, and data associations. This is accomplished using the EKF measurement update equations

$$K_t = P_t^- H_t^T (H_t P_t^- H_t^T + V_t R_t V_t^T)^{-1} \tag{3.26}$$

$$\hat{x}_t = \hat{x}_{t-1}^- + K_t(z_t - h(\hat{x}_t^-, 0)) \tag{3.27}$$

$$P_t = (I - K_t H_t) P_t^- \tag{3.28}$$

This actually does two steps, it uses the measurement model $h(\hat{x}_t^-, 0)$ as an error metric for the least squares formulation, then it uses this to weight the predicted mean hence correcting it appropriately. The EKF measurement update equations implement the Bayes filter steps

$$p(z_t|s_t, \Theta, n^t) p^-(s_t, \Theta|u^t, n^t) \tag{3.29}$$

After a succession of time-update and measurement update stages, the current mean and covariance reflect the parameters of the posterior probability density function over robot pose and map point positions.

The disadvantages of the EKF formulation are threefold

- **Linearized motion model**. The assumption that the motion of the vehicle is close to linear can become fatal for all EKF formulations. If the system exhibits highly non-linear motions, then the EKF can diverge. However, if the motion is close to being linear or the time step between filter updates is small then this is a good assumption and works well in many situations.

- **Quadratic Complexity**. The filter update procedure requires a quadratic (in the number of landmarks) number of operations. The covariance matrix P_t is size $(2N+M)\times(2N+M)$ where M is the number of variables being estimated for the robot pose. Computationally, the most number of operations occur when computing the Kalman gain

$$K_t = Pt^{-}H_t^T(H_tP_t^{-}H_t^T + V_tR_tV_t^T)^{-1} \tag{3.30}$$

  The measurement jacobian H_t is of size $(2N+M)\times L$ where L is the dimension of the measurement vector. The term $H_tP_t^{-}H_t^T$ requires that the covariance matrix be first inverted (traditionally matrix inversion is an $O(N^3)$ algorithm, strictly speacking theoretically the time complexity of matrix inversion is $O(N^{\log_2 7})$[63], either way it is cubic or quadratic (at best) in N) and then multiplied by a matrix of size $(2N+M)\times L$. Thus, computing the Kalman gain requires a quadratic number of multiplications in the number of landmarks. This makes it computationally difficult to estimate an increasing number of landmarks within the EKF itself.

- **Single-Hypothesis Data Association**. When a sensor reading of the environment is available, it must be associated with a map point to update the EKF appropriately. This is typically done through a maximum likelihood heuristic. The problem with this heuristic, is that if the data association is incorrect, the filter is updated incorrectly. If this happens for too many map features often, the EKF will diverge since the effect can never be corrected (see [14]).

A useful extension to the extended Kalman filtering framework is the *Compressed Extended Kalman Filter* (CEKF) introduced by Guivant [31, 32, 30]. This is essentially a hierarchical approach to Kalman filtering implementing a very local update scheme with a global update once in a while. The CEKF exploits the fact that not all landmarks need to be present in the state update equations. The filter produces an identical state estimate to the EKF but at a lower computational cost. This algorithm works well when the robot is exploring a very local area but when the vehicle transitions to a different area a full SLAM update must be done; practically this happens infrequently. The idea of the CEKF algorithm is to run the standard EKF algorithm on a smaller set of data in a local area of the map. This allows for an increase in performance since the entire state and covariance need not be used. However, to be consistent when the robot moves outside of this local area, the entire state and covariance must be updated to reflect the changes that have been made locally. To do this, an additional set of state and covariance matrices are maintained during the local updates and are used in the postponed global system update.

3.3 Particle filtering approaches

3.3.1 Rao-Blackwellised Particle Filters

The Rao-Blackwell theorem [] shows how to improve on any estimator under every convex loss function [5]. This theorem has been extended to Markov chain Monte Carlo methods by Gelfand and Smite [] and Liu, Wong and Kong[]. Using this to improve variance allows the algorithm to integrate out ancillary variables and create an estimator provably superior to the original estimator. A Rao-Blackwellised particle filter provides a method of simplifying a probability estimation scheme by partitioning the state space into independent variables and marginalizing out one or more of the components of the partition, see [5, 16] for further discussion on the topic of Rao-Blackwellisation and Rao-Blackwellised Particle Filters (RBPFs). The most common realisation of Rao-Blackwellisation for Particle Filtering is outlined by Murphy in [56, 57]. Here, they use Dynamic Bayesian Network theory to show that it is possible to separate the SLAM posterior into estimating two independent posteriors. One posterior over maps, and the other posterior over robot *paths* or entire *trajectories* instead of pose. FastSLAM [52] is a popular implementation of this type of Rao-Blackwellised Particle Filter framework. Grisetti showed improvements in the basic RBPF framework in [29] by estimating an improved proposal distribution (sampling the predictive motion model), however this work assumes a normalized Gaussian motion model but seems to work well in practice.

3.3.2 FastSLAM

Most solutions to SLAM attempt to estimate the posterior over robot *pose*, a single pose of the robot. The FastSLAM approach, introduced by Montemerlo in [52], takes a slightly different approach to solving SLAM. FastSLAM estimates a different posterior

$$p(s^t, \Theta | z^t, u^t, n^t) \tag{3.31}$$

where $s^t = s_1, s_2, \ldots, s_t$ is a robot path or trajectory. The algorithm estimates the posterior over maps and robot *paths*. As shown in [51], this allows the SLAM problem to be factored into a product of simpler terms. By using ideas from Dynamic Bayes Network theory, Montemerlo observed that if the robot path is known then the observation of each landmark does not provide information about other landmarks. Thus, given the robot path, each landmark is independent from the rest of the landmarks. This enabled

Montemerlo to factor the SLAM posterior into the following form

$$p(s^t, \Theta | z^t, u^t, n^t) = p(s^t | z^t, u^t, n^t) \prod_{n=1}^{N} p(\Theta_n | s^t, z^t, u^t, n^t) \tag{3.32}$$

This states that SLAM can be decomposed into estimating the product of a posterior over robot paths and N landmark posteriors given knowledge of the robot's path.

Due to the separation between the pose and the landmarks, the estimation can be slightly decoupled into a filter that estimates the robot pose, followed by N-landmark estimators (1 per landmark). In [51], Montemerlo describes his implementation as partitioning the SLAM problem into a localization problem and a mapping problem. He solves localization through the use of a particle filter and the construction of a map is performed using a set of independent EKFs, one per landmark. The original paper [52], describes the algorithm assuming that the data association is known. This becomes an issue in practice since the data association is almost never known. In [60], Nieto shows how the data association problem for FastSLAM can be handled using *Multiple Hypothesis Tracking* (MHT) algorithms. This is implemented by splitting each particle in the FastSLAM filter into $n+2$ particles, where n is the number of hypotheses to be tracked. Thus, there are n hypothesis particles containing a different data association for the observation, there is 1 additional particle for the non-association hypothesis (i.e. to ensure that it is possible to not have a data association at all) and another particle for a new landmark hypothesis. As new measurements are incorporated into the filter, the wrong data associations are less likely to effectively describe the robot motion and will thus be pruned out in the particle resampling stage.

3.3.3 DP-SLAM

DP-SLAM [20, 21] is a pure particle-filtering approach to solving SLAM. It uses a particle filter to maintain a joint probability density over robot positions as well as the possible map configurations. This approach removes the need to maintain separate EKFs for the landmarks as in FastSLAM. DP stands for *Distributed Particle* mapping and allows for efficient maintenance of hundreds of candidate maps and robot poses. The data structure used in DP-SLAM plays a key role in the efficiency of this algorithm. Each particle in essence keeps its own version of the map represented as an occupancy grid. In each grid-cell, a balanced tree is kept of all of the particles that have updated this cell. A second data structure called the ancestry tree per particle is also maintained which describes the relationships between particles at time t and the sampled particles at time $t+1$. Instead of associating maps with particles, the algorithm associates particles with maps. An update consists of the following; when a particle makes an observation about a particular grid cell, the ID of that particle is inserted into the balanced data structure stored in the cell. In order to check the occupancy state of that cell, the particle looks into its ancestry tree and finds an ancestor that has updated this cell previously. If no ancestor is found then the state of the cell is currently unknown to this particle. In this way, each particle is able to efficiently maintain its own version of the map (i.e. thus multiple maps are estimated). An important advantage of this algorithm over most SLAM solutions is that there is no need for explicit data association or external loop-closing (§ 4.5). Since the algorithm maintains multiple maps and robot locations the proper data associations are estimated and loops are automatically closed. The algorithmic complexity of this algorithm was analyzed to be log-quadratic in the number of particles.

Subsequently, an addendum to the original DP-SLAM algorithm called DP-SLAM 2.0 [21] was developed. The new algorithm makes improvements in the laser uncertainty model. In the original algorithm there was an inconsistency with the laser model since they assumed it was perfect for the mapping problem, but assumed it was noisy for the localization problem. DP-SLAM 2.0 addressed this issue by using a probabilistic laser penetration model that is dependent upon the distance the laser has traveled through each grid cell. Using this method, the occupancy grid was improved from storing binary states (*occupied* or *empty*) to a more stochastic representation. Each cell has a probability associated with the laser penetration model, the higher the probability the more likely the cell is to be occupied. The search method for the localization stage was also improved. For each particle, when an update occurs it is required to search through its ancestry to determine if any of its ancestors have updated this cell before. The new method uses a more efficient search method using a batch search of all ancestors at the same time. Simplifications can then be made to the algorithm using sorting methods and the overall time complexity is reduced from log-quadratic to simply quadratic in the number of particles.

3.4 Other notable approaches

3.4.1 Sparse Extended Information Filters

Recently, there has been a thrust in developing SLAM solutions using information theory. Most notably is the work by Nettleton [50, 58] using Extended Information Filters. The use of information filters in SLAM was proposed by Frese in [26], implemented by Thrun in [80], and related to the work of Lu and Milios [49]. The idea of using information filters for SLAM is to represent maps of the environment by a graphical network of locally interconnected features. Each link in the network represents relative information between pairs of features in a local neighbourhood. The link also encodes information of the robot pose relative to the map. At each step, weak off-diagonal elements are removed from the information matrix to guarantee sparsity. An interesting result of Thrun's work is that by exploiting the sparsity of the information matrix, the algorithm complexity is reduced from the EKF's $O(n^3)$ to $O(n)$, the most important update equations can be performed in constant time; In other words, the time complexity for each update is independent of the number of landmarks being estimated in the map. However, even though the algorithm requires constant time for a measurement update, the relaxation process used introduces small errors in the resulting map that negatively affects the overall map fidelity.

3.4.2 Thin Junction Tree Filters

Paskin [61, 62] proposed the *Thin Junction Tree Filter* to solve SLAM. The idea here is to view the SLAM problem as a graphical problem and notice that if a node separates the graph into two partitions, these partitions are conditionally independent of one another given that you have an estimate for the nodes that create the partitions. The junction tree of these separating nodes is maintained which grows with the measurement and motion updates. To keep the computational complexity to a minimum, the tree is thinned, or equivalently weak edges are removed from the graphical model. This is essentially the same goal as the Sparse Extended Information Filter above and shows promise for solving SLAM in practice.

3.4.3 Submap Approaches

Submap approaches are another popular approach to solving SLAM. In this paradigm, small independent maps are estimated. This allows the algorithm to work only on a small amount of data at a time, thus an update can take constant time. The issue with these algorithms is that it is difficult to align the submaps together.

One implementation of the submap approach is that of Atlas [2]. Atlas merges the ideas of metric and topological map building into a single framework. The algorithm maintains a graph where each node in the graph represents a local coordinate frame and each edge represents the transformation between these frames. A metric map is maintained per node thus merging the ideas of metric and topological mapping. Each submap is small and of constant size and is estimated using a Kalman filter, thus covariance estimates of each map point are also estimated along with the robot pose relative to this local frame. This allows for a constant time updating scheme. However, even though updating a submap can be performed in constant time, a map matching algorithm must be employed to close loops and identify submaps that have been visited before. Even though an efficient algorithm has been implemented, the time complexity of the

Map Matching is still $O(n^2 \log n)$. During the Map Matching process, edges are created between submaps that are *close*. This enables the algorithm to identify possible loops in the graph. One issue with Atlas' approach is that since only local submaps are maintained, a global pose estimate is unavailable. Plus, since the map matching process adds edges between nodes, there is a discrepancy between the global pose of this submap determined through different paths through the graph. To obtain a global map of the envrironment from the topological map requires the estimation of this discrepancy. This is posed as a non-linear least squares optimization problem to find the global arrangement of map poses of all frames that minimize the discrepancy error over all edges.

Another approach related to submap algorithms is the Hybrid-Metric Maps (HYMMs) algorithm from Nieto and colleagues [59]. Their solution uses an occupancy grid as their map representation and combines feature maps with other sensory information. The feature maps are used to partition the explored environment into smaller regions. Each feature position defines a coordinate frame for this region and a global dense map consists of many smaller submaps defined relative to the coordinate frame in their region. When the position estimate of the features change, the dense representation of the area is automatically updated since they are defined relative to the local region frame. This algorithm is similar to Atlas' described above in that it uses a hybrid mapping approach that combines metric information (the dense sensory information) with topological information (relationships between salient features).

3.5 Summary

This chapter introduced different approaches to solving the SLAM problem. The most standard approach is the Kalman filter based approach where both the map and pose are encoded in the state vector. The state is defined to be the mean of the unimodal Gaussian probability distribution and a covariance matrix encodes the interactions between elements in the state vector.

Lately, Particle-filtering based approaches have become successful in solving SLAM and it seems that the research community is leaning towards these types of solutions rather than the Kalman filter based approach. This is due to the fact that Particle-filter-based solutions address the core issues in the Kalman filter, namely that a Gaussian model does not necessarily fit the SLAM problem well. Particle-filtering can represent any probability distribution given enough samples. A popular algorithm in use is FastSLAM, and not just for its catchy name, but also because it works very well in practice. FastSLAM estimates the posterior distribution over robot paths (trajectories) and landmarks. The FastSLAM algorithm utilizes a separate Extended Kalman Filter per landmark and a particle filter that samples the robot trajectory. The novelty of the FastSLAM algorithm is that Montemerlo showed that it was possible to accurately solve SLAM using Particle-filters and that it was possible to separate the two problems of mapping and localization. Subsequently, FastSLAM has data association issues which Nieto solved using a multiple hypothesis tracking algorithm that exploited the particle-resampling phase to pick the proper data association.

FastSLAM works well in practice, however there are drawbacks to this approach. As Paskin points out in [62] and [61], since the data association problem must be solved on a per-particle basis, each particle contains a different representation of the environment. Merging these multiple maps is nontrivial and computationally expensive. Also, if landmarks are sparse or measurements have high amounts of noise then the FastSLAM algorithm is prone to diverge. This occurs because of the dimensionality of the problem. Since the probability distribution that is estimated is the posterior over robot *paths*, as time progresses the dimensionality increases linearly. The use of particle-filters requires that the particle population is large enough to sample the region of “high-probability” in the sample space. If too-few particles exist, then the filter essentially ignores hypotheses in the hidden state. Thus, to avoid divergence, FastSLAM requires frequent updates with informative observations, i.e. when closing a loop. As a loop is closed, the particles that represent this change are weighted much higher than other particles and are less-likely to be pruned out in the resampling phase. Thus, only the particles that are consistent with closing the loop remain. From that point onwards, the particles begin to spread out until another loop is closed. Thus, in situations where extremely large loops exist or if there are no loops at all, then the particles will continue to diverge and spread out from one another. Effectively, this is the core issue of FastSLAM. In order to maintain a high degree of accuracy, many more particles are required to sample the robot path space which increases the computational requirements of the algorithm.

FastSLAM is a hybrid particle-filtering and Kalman filtering algorithm and contains elements of both including their strengths and weaknesses. A full Particle-filtering SLAM algorithm known as DP-SLAM was discussed which shows promise in solving SLAM effectively in practice. Since it is a particle-filtering approach, it suffers from the need to sample the space with an appropriate number of particles, however unlike FastSLAM the dimensionality of the problem is constant over time since it estimates the posterior over robot *poses* instead of paths. One issue with DP-SLAM is that even though for simple environments the algorithm can function well with in near-real time with 3000 particles, the results shown in their paper required around 24 hours of computation time since they were extremely complex environments.

Other approaches to solving SLAM were also discussed in this chapter. These involve submap approaches such as Atlas which separates the problem of mapping into a very local problem that can be updated in a constant amount of time. This is an interesting approach which is capable of mapping very large areas effectively. The drawback to this approach is the lack of a good mechanism to merge local maps into a global world representation. This is a complex algorithm that requires computational time quadratic in the number of local maps. In related work, the goal of the Hybrid Metric Maps algorithm is to acquire a dense representation of the environment. They do not limit themselves to a particular type of sensor. Many local feature maps are used to partition the envrionment into smaller regions. A topological map of coordinate frames are estimated and a dense map representation is attached to each frame. This approach seems promising in acquiring a dense representation of the environment however a full computational complexity examination of this algorithm has not yet been produced.

Recently, the Sparse Extended Information Filter (SEIF) [80], approach is gaining in popularity. It uses the EKF framework to maintain an Information matrix, namely the inverse of the covariance matrix. The fundamental assumption in this algorithm is that if the covariance matrix is dense, then the information matrix will be sparse or very close to being sparse. This is an appropriate assumption since work of Newman and Csorba proved how as time progresses the covariance matrix becomes dense, that is errors in each landmark become fully correlated with each other. The interesting point about the SEIF approach is that the computational complexity of the update procedure is constant. The sparsity of the information matrix is exploited allowing the algorithm to update only the neighbouring nodes. However, even though the algorithm update requires constant time, the map itself is inaccessible at any point in the algorithm. In order to extract a map from the information filter, the information matrix must be inverted which is typically a $O(n^3)$ algorithm, quadratic at best.

Thin Junction Tree Filters (TJTF) were also discussed. This solution seems promising and addresses some issues in the FastSLAM approach. Initial experiments indicate that the TJTF algorithm seems to work well in closing large loops, something that FastSLAM has difficulty with. The authors state that the algorithm is expected to be less susceptible to divergence than FastSLAM however a proof of this is not given. However, as Thrun points out in [80] this method presently does not address the data association problem efficiently.

Chapter 4

Key Issues and Open Problems

In this chapter, I will discuss some issues with developing a solution to SLAM, i.e. things that one must take into account when developing a SLAM algorithm. Identifying the core issues reveals areas that are still immature and some of the open problems become apparent. Key issues in SLAM include

- Representational issues
- Unstructured 3D Environments
- Dynamic Environments
- Data Association issues
- Loop Closing

4.1 Representational issues

One of the key issues in developing any solution to SLAM is representation. That is, there are questions that need to be asked about the representation of the map, robot pose, and the probability distributions. Should the map be composed of a sparse set of landmarks (i.e. salient image features), or should it be composed of a dense set of data (i.e. dense laser scan data or dense depth from stereo imagery)? What form of the probability distribution functions should be employed?

In an EKF-based solution, a Gaussian form is assumed for all pdf's. This has its limitations but provides a relatively simple way of proving optimality and convergence properties. Particle-filter based approaches do not use a closed-form but rather use sampling to estimate a generalized pdf. This is useful in the sense that it can represent any pdf, however many samples are required for more general pdf's. Are other representation formalisms more effective or useful for solving SLAM?

4.2 Unstructured 3D Environments

A mostly unexplored area is large unstructured 3D environments. Many applications require a robot to operate autonomously for hours, possibly even days while travelling many kilometers from its starting position. Underwater exploration applications, for example, may require the robot to collect 3D data for large sections of ocean-floors or the entire lake-bed. City exploration, i.e. being able to accurately and automatically construct a 3D map of an entire city could be beneficial for urban planning applications.

There have been several papers on SLAM in large unstructured environments, (e.g. [33, 82]), however these results still provide a two dimensional map of the environment as output and are tested only on standard datasets from fairly structured environments. Some work has been done in 3D EKF-based SLAM using a single camera [11] and also using a wide-angle camera [12]. Davison uses a single camera and an EKF to estimate the pose of the camera with respect to static features that are tracked in real-time. Here, the static world assumption is abundant and the algorithm fails if too many features are improperly tracked, i.e. the object moves while other features stay stationary.

One interesting approach to 3D SLAM is presented in [67, 68]. Here, a stereo camera is used to extract a dense 3D mesh of the world, and scale-invariant features (SIFT features [47]) are used to estimate the camera's ego-motion. The SIFT features are used as tie points between the 3D mesh sets to compute the registration between point clouds. The dense 3D depth information is registered using the computed transformation and a 3D surface model is extracted. This algorithm seems to work well in unstructured 6DOF domains, however it assumes the presence of valid SIFT features in the environment. These are typically locally planar patches that have high gradient response in multiple levels in an image pyramid. Their algorithm does not exploit the dense data in the registration process and thus errors can be accumulated. Also, there is no automatic method of closing loops while estimating the pose. This algorithm treats the SLAM problem as two interleaved independent problems, estimating a map from known pose and then estimating a pose from known map points.

4.3 Dynamic Environments

It is important to address the issue of dynamic environments when developing a solution to the SLAM problem. The world we live in, and the one in which a robot will be navigating, is not static. The state of the world is constantly changing, objects are moved and doors are opened. Moving objects or animals move in and out of the navigated area constantly in a real world application. In an indoor environment typically people move in front of the robot, underwater applications have fish swimming around the scene. These play havoc on most algorithms since each frame is considered individually and assumed static. The effect that this has on the global map is tremendous. First, the possibility of divergence increases since map registration is highly dependent upon data association. The accuracy of the registration is degraded when spurious dynamic objects are improperly treated as static objects since their motion will be estimated and not just the motion of the robot. Second, even if the map registration converges the accuracy of the map will contain elongated objects due to their motion.

SLAM within highly dynamic environments have not been addressed fully in the literature. There is currently no fully encompassing framework that solves SLAM in the presence of dynamic environments. There has been work in this field however. Current work in navigating in the presence of moving objects has been looked at in [54, 4, 37, 38, 84]. In [38], an expectation maximization algorithm has been developed which uses laser range data as input and updates the global map of the world in two steps. The first step (Expectation step) estimates probabilistically which measurements are associated with static objects and which ones are associated with dynamically moving objects. The second step (Maximization step) uses these estimates to localize the robot within the map. Finally, these steps are iterated until no improvement in the map is made. The results from this algorithm are promising and is capable of extracting 3D information of people walking in a static environment.

In another approach to dynamic environments [84] two occupancy grids are estimated, one for the static portions of the map and the other models the dynamic portion. A complete representation of the environment can be produced through the union of both grids. A third map is also maintained which represents static landmarks detected in the environment which provides the robot with a way to localize itself. A probabilistic framework is used to update the occupancy grids based upon range data. In this context, "dynamic" denotes areas that change occupancy state in their static grid. Thus, if it sees an area which is occupied and we had previously observed that it was static moments prior, then this is designated as a dynamic portion in the environment and the dynamic occupancy grid is updated appropriately. The features used to localize the robot are corners in the range data and are separately found and used to estimate the robot pose. This method provides another heuristic method of coping and identifying dynamic portions of the environment.

4.4 Data Association issues

Data association is the problem of associating observations from sensors to a particular landmark or area of the map being estimated within the SLAM algorithm. Robust data association is necessary for any SLAM algorithm. In an EKF SLAM algorithm, it is necessary to have correct data associations, if not then the filter will diverge. FastSLAM also requires good data associations but is more tolerant to spurious data due to the particle resampling phase. However, if too many spurious measurements are associated improperly, then the filter will not produce a correct map/pose estimate. Nieto [60] proposed a multi-hypothesis scheme to solve the data association problem with FastSLAM by using the particle resampling phase as the mechanism to prune incorrect associations. A review of statistical techniques for data association for motion correspondence has been looked at by Cox in [9]. Another highly used algorithm for fitting models to data is RANSAC [23] which could be used to determine a set of samples that fit a minimal model well. This algorithm could be used to help in robustness for determining data associations.

Difficulties for associating measurements with map elements includes imprecision, noise and clutter. If the sensor is not precise or noisy, then the probability of associating each measurement with an incorrect map measurement is high. Some sensors produce very ordered data sets (such as laser scanners) which have low clutter and are precise. Other sensors such as stereo vision produce a tremendous amount of data points but are highly cluttered. High clutter increases the probability of incorrect association without another mechanism. Using features, or small sets of salient observations, is useful for determining matches.

A typical approach to matching multiple sets of 3D points is the iterative closest point algorithm [1]. In this algorithm, it is assumed that the point sets are overlapping. The distance between closest points are minimized by determining the “best” (in a least squares sense) rotation matrix and translation vector. The final rotation and translation are considered to be the registration transformation between the two point clouds that aligns them properly.

Some issues concerning the data association problem are:

1. Noisy measurements.

   Each sensor has noise characteristics that must be modeled. This is typically done offline but is necessary for the system to converge appropriately. Additive noise to the measurements makes it much more difficult to associate measurements since the same static point in the world is not sampled identically due to noisy sensors.

2. Precision of the sensors.

   Cameras have a limited resolution. Sonars are precise within a known distance range. Sensor characteristics limit the precision of the generated localization and map. Each sensor used must be carefully modeled and the precision must be taken into account. Sampling intervals must be monitored and the algorithm should not attempt to generate a model that is more precise than the data given by the sensors. As Nyquist said, you can only accurately reconstruct a signal with a sampling rate of twice the highest frequency which also limits the precision of the reconstruction or map that theoretically can be generated with a particular set of sensors.

3. Dynamically moving objects.

   Typically SLAM implementations use a static world assumption to avoid having to deal with data association issues caused by moving objects. This is highly restrictive in a real setting since almost all real-world applications exist in a highly dynamic environment. If this issue is not carefully addressed the accuracy of the localization and map may be degraded, or the filter may not converge properly.

4. Sparse Data.

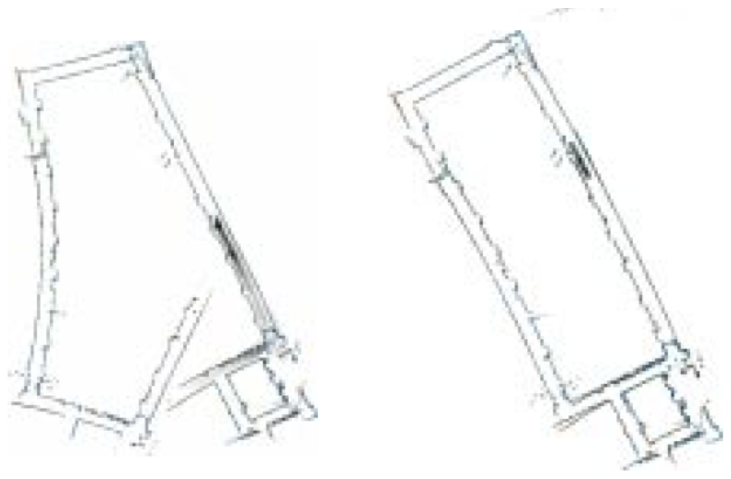


Figure 4.1: Before and after closing a loop. Courtesy of [34].

> Many types of sensors (e.g. sonar or a single laser point rangefinder), provide sparse amounts of data fast while acquiring a more dense amount of data usually requires several seconds. This impacts the map building process since it is usually advantageous to obtain a multitude of points for data association. More points equals more possible features, while sparse data sets allows only a sparse reconstruction and hinders the point matching process due to the lack of features.

In order to accomodate these issues with data association, the use of statistics is typically employed to determine outliers and to find the ’best’ match to the data given sets of noisy measurements.

Most SLAM algorithms cannot cope with a failure to properly associate measurements to map features. For instance, in a Kalman filter based SLAM algorithm, if too many measurements are incorrectly assigned, the filter will diverge. Hahnel [36] explicitly addresses this problem and constructs a theoretical framework for performing a “lazy” data association. This involves maintaining a tree that represents the possible data associations at each update. The log-likelihood for each association is stored in the tree.

4.5 Loop Closing

As a robot moves through the environment, the estimate of its position suffers from accumulated error. Thus, the map quality is degraded and does not necessarily reflect reality. In order to overcome this error, the robot must sense places that it has been before, understand that this position has been already visited and then correct its version of the map accordingly. This process is called *closing the loop* in SLAM since it involves the robot moving in a loop and associating new observations with the existing map. The effect of this can be seen visually in Figure 4.1. In this scenario from Konolige and Gutmann in [34], the robot relies on odometry for its pose information. As can be seen, the robot's map before the loop is closed has drifted quite far from reality. The next image shows the effect of using a map matching algorithm to close the loop and propagate the error throughout the rest of the map positions. The resulting map is much closer to reality and the robot can now explore the rest of the map properly. Loop closing is an important part of the SLAM process and greatly influences the exploration strategy. A successful SLAM algorithm should be able to close loops properly with little computational cost since the map may be very large and the algorithm should have close to real-time performance. If uncertainty is handled properly within the SLAM framework, then the map will deform itself appropriately when loops are closed in order to maintain consistency. Loop closing can also be seen as the effect of correct data association.

Lu and Milios[49] call this *global consistency*, i.e. two landmarks are adjacent if and only if the robot sensed this relationship. They use scan matching and iterative minimization techniques to align multiple laser range scans. This allows for a reduction of uncertainty in the global map estimate.

In the FastSLAM algorithm [53], since it is a particle filter loops are closed automatically during the particle resampling phase. The particle resampling strategy prunes out particles that do not explain the environment properly. As the robot moves, the particles spread out due to uncertainty in the robot pose and sensors. When it visits already seen areas, the error of the robot pose estimated with the particles that close the loop is significantly smaller than the error of the other particles. Thus in the resampling stage, the particles that close the loop are chosen more often to remain in the particle set. This provides a huge advantage over other SLAM approaches since closing loops is a side effect of the algorithm itself. Thus, no additional computation needs to be involved to close loops. Similarly, DP-SLAM[21] is a particle filtering algorithm which addresses the loop closing effect implicitly as part of the algorithm.

EKF-based SLAM algorithms suffer from the inability to implicitly close loops. Thus, an expensive data association computation must be performed to take loops into account. If the loops are large enough, then each map point must be matched with every other map point (a $O(n^2)$ solution). However, if the exploration strategy can gurantee that loops can be indeed be small, efficient solutions using Quad-tree or Octree representations can be developed that reduces the computational complexity.

4.6 Other Open Problems

Coupling sensing in closed loop with SLAM has not been looked at yet to the best of my knowledge. That is, how can you fully incorporate the system dynamics generative model with the SLAM estimation scheme. Is it possible to have a probabilistic framework that can modify the generative model as the robot moves? As an example, consider a stereo sensor system. As the robot maps an environment it will begin to view areas that have been previously sensed. This could inform the sensing process as one has a good estimate of what should be sensed at this time. To date, all sensing paradigms treat the sensors as a black-box, they assume that the parameters can be modeled but not modified. With stereo imagery, for example, the matching process could be guided with an initial estimate, and the errors of this process could be used to improve the map. The question arises, does this closed loop between sensing and SLAM improve performance?

The type of sensor to be used in SLAM is typically a laser range finder. There has been some work using stereo vision [27], and the early work focused on sonar range data. However, no algorithm is able to use the information from multiple different types of sensors simultaneously. The issue of sensor representation within SLAM has not yet been addressed. Possibly, the use of multiple different types of sensors can aid in mapping. The map itself may not necessarily be defined solely by range information to points in space, but rather a collection of sensorial information. It is possible that the mapping process could be made more robust by using visual information, range information, electromagnetic field strength information, or even sound information from microphones. Typically a robot does not have all different types of sensors for monetary reasons, but the algorithms used in SLAM have focused primarily on range information.

Another issue in SLAM is which posterior should be estimated? With appropriate assumptions, Fast-SLAM estimates the posterior over robot paths (trajectories) whereas most other implementations estimate the posterior over robot pose. It is unclear as to which of these is a better choice.

4.7 Summary

Simultaneous localization and mapping is a highly active field of research. Applications of SLAM in autonomous robotics include active exploration of unknown environments which is of major concern for underwater and planetary exploration. The basis of all successful SLAM algorithms is a probabilistic framework which is appropriate since uncertainty is inherent in any robotic system.

There is a multitude of algorithms that have developed to solve this problem and have been discussed in this report. The open problems that require more research are mainly that of representation, full 3D unknown environments, and dynamic environments. For a robot to fully understand the world, it will be required to identify dynamic changes in the world and understand the consequences of these changes. For instance, if the goal is autonomous navigation and exploration of unknown environments, then the robot should be able to create a map in the presence of moving objects such as people or other vehicles, and identify existing areas of its map that has changed state such as doors that have closed, or chairs that have moved. Work has been performed to accomodate such changes, however to date no general framework for SLAM in dynamic environments exists.

Other issues such as the coupling of dynamic sensor models within SLAM has not yet been addressed. It is possible that the sensor used to observe the world changes its parameters over time. Or that the current environment map and robot pose could influence either the characteristics of the sensor or aid in extracting information from the sensor itself. In a sonar scenario, the sensor provides poor or noisy information when close to angled planar walls. As the robot moves closer to such places, this could inform the sensor module that it should evaluate the sonar reading differently or wait longer than usual for a return echo. It is possible that this closed-loop between SLAM and sensor parameters is useful for solving SLAM more efficiently or accurately.

This report has provided an overview of the necessary probabilistic mechanisms used in the state-of-the-art algorithms for solving SLAM. A review of current algorithms has been presented and a discussion on the core issues still to be solved in SLAM has been given.

Appendix A

Kalman Filter Derivation

The expected value of a random variable x at time t is denoted as $E[x_t] = \hat{x}_t$ and its covariance matrix is P_t. The predicted state at time $t+1$ is written as $\hat{x}^-_{t+1}$ and the predicted covariance is P^-_{t+1}.

In general, the Kalman filter models the state dynamics as a linear function of the previous state (Markov assumption), the control input and the system dynamics zero-mean Gaussian noise.

$$x_{t+1} = f(x_t, u_t, w_t) = A_t x_t + B_t u_t + w_t \tag{A.1}$$

Here, the system dynamics is modeled as a linear combination of the previous state, x_t, the control input u_t and the system noise w_t. The matrix A is the state transition matrix and the matrix B is the transition matrix for the control input, i.e. how to relate the control to the state. The state is assumed to be hidden and can only be estimated through observations from some uncertain (noisy) sensor. Thus, the measurements must be modeled as well. This is accomplished in the standard Kalman filter as a linear function of the state plus some noise associated with the sensor.

$$z_t = h(x_t, v_t) = H x_t + v_t \tag{A.2}$$

Here, z_t is the measurement, x_t is the state at time t, v_t is the measurement noise and the matrix H relates the state to the sensor.

In the above, the noise models are assumed to be Gaussian and zero-mean, or that

$$w_t \sim N(0, Q)\ ,\ v_t \sim N(0, R) \tag{A.3}$$

where Q and R are the covariances of the noise models

$$Q = E[w_t w_t^T]\ ,\ R = E[v_t v_t^T] \tag{A.4}$$

Also note that the noise is not correlated with the state or measurements, thus the following are true

$$E[x_i v_j^T] = 0\ ,\ \forall i, j \tag{A.5}$$
$$E[x_i w_j^T] = 0\ ,\ i \leq j \tag{A.6}$$
$$E[z_i v_j^T] = 0\ ,\ i \leq j-1 \tag{A.7}$$
$$E[z_i w_j^T] = 0\ ,\ i \leq j \tag{A.8}$$
$$E[v_i w_j^T] = 0\ ,\ \forall i, j \tag{A.9}$$
$$E[v_i v_j^T] = 0\ ,\ i \neq j\ ,\ E[v_i v_j^T] = R\ ,\ i = j \tag{A.10}$$
$$E[w_i w_j^T] = 0\ ,\ i \neq j\ ,\ E[w_i w_j^T] = Q\ ,\ i = j \tag{A.11}$$

The objective of the Kalman filter is to estimate the posterior of the state, $\hat{x}$, at time $t+1$ given measurements, the previous state, and control inputs, namely $p(x_{t+1}|x_t, z^t, u^t)$ to optimally reduce the error covariance in the state. The derivation that follows is based on that found in [41] and the interested reader should refer to this as well as [83] for more information.

Given the above state dynamics model, the expected value of the state at time $t+1$ is

$$\hat{x}_{t+1} = E[x_{t+1}] \tag{A.12}$$
$$= E[A_t x_t + B_t u_t + w_t] \tag{A.13}$$
$$= E[A_t x_t] + E[B_t u_t] + E[w_t] \tag{A.14}$$
$$E[w_t] = 0 \text{ since it is zero-mean noise}$$
$$= E[A_t x_t] + E[B_t u_t] \tag{A.15}$$
$$E[u_t] = u_t \text{ since it is not a random variable}$$
$$= A_t E[x_t] + B_t u_t \tag{A.16}$$
$$\text{Substituting } E[x_t] = \hat{x}_t \text{ gives} \tag{A.17}$$
$$\hat{x}_{t+1} = A_t \hat{x}_t + B_t u_t \tag{A.18}$$

Now, the expected value of the covariance at time $t+1$ must be estimated. To do so, an error metric for the state must be contrived. Given an estimate of the state vector $\hat{x}_t$, let an error function be denoted as

$$e_t = f(\hat{x}_t) = f(x_t - \hat{x}_t) \tag{A.19}$$

The function $f(\hat{x}_t)$ is implementation specific however it is useful to have the function be positive and increase monotonically. The mean squared error is a typical example,

$$e_t = x_t - \hat{x}_t e_t^2 = (x_t - \hat{x}_t)^2 \tag{A.20}$$

or more importantly, the expected value of the above error function can be computed as

$$\varepsilon(t) = E[e_t^2] = E[e_t e_t^T] = P_t \tag{A.21}$$
$$P_t = E\left[(x_t - \hat{x}_t)(x_t - \hat{x}_t)^T\right] \tag{A.22}$$

where P_t is the state error covariance matrix at time t. It is the goal of the Kalman filter to estimate the state $\hat{x}_{t+1}$ given the measurements $z^t = z_0, z_1, \ldots, z_t$ that minimizes the expected covariance P_t.

The expected error covariance of the state at time $t+1$ can be computed as follows

$$e_{t+1}^- = x_{t+1} - \hat{x}_{t+1} \tag{A.23}$$
$$= (Ax_t + Bu_t + w_t) - (A\hat{x}_t + Bu_t) \tag{A.24}$$
$$= A(x_t - \hat{x}_t) + w_t \tag{A.25}$$
$$= Ae_t + w_t \tag{A.26}$$

Therefore, the expected covariance at time $t+1$ can be estimated as

$$\begin{aligned}
P_{t+1}^{-} &= E[e_{t+1}^{-}e_{t+1}^{T-}] && \text{(A.27)}\\
&= E\left[(Ae_t + w_t)(Ae_t + w_t)^T\right] && \text{(A.28)}\\
&= E\left[Ae_t(Ae_t)^T + 2Ae_t w_t^T + w_t w_t^T\right] && \text{(A.29)}\\
&= E\left[Ae_t(Ae_t)^T\right] + 2E\left[Ae_t w_t^T\right] + E\left[w_t w_t^T\right] && \text{(A.30)}\\
&= E\left[Ae_t(Ae_t)^T\right] + E\left[w_t w_t^T\right] && \text{(A.31)}\\
&= E\left[Ae_t e_t^T A^T\right] + Q && \text{(A.32)}\\
&= AE\left[e_t e_t^T\right] A^T + Q && \text{(A.33)}\\
P_{t+1}^{-} &= AP_t A^T + Q && \text{(A.34)}
\end{aligned}$$

The above so far contains a theoretical framework for predicting the state and covariance forward in time. Now sensor measurements must be taken into account.

Rearranging Equation A.2 and solving for the state at time $t+1$ gives

$$\begin{aligned}
z_{t+1} &= H_{t+1}x_{t+1} + v_{t+1} && \text{(A.35)}\\
z_{t+1} - v_{t+1} &= H_{t+1}x_{t+1} && \text{(A.36)}\\
H_{t+1}^{-1}(z_{t+1} - v_{t+1}) &= H_{t+1}^{-1}H_{t+1}x_{t+1} && \text{(A.37)}\\
x_{t+1} &= H_{t+1}^{-1}(z_{t+1} - v_{t+1}) && \text{(A.38)}
\end{aligned}$$

And the expected value of the above is

$$\begin{aligned}
\hat{x}_{t+1} = E[x_{t+1}] &= E\left[H_{t+1}^{-1}(z_{t+1} - v_{t+1})\right] && \text{(A.39)}\\
&= E\left[H_{t+1}^{-1}z_{t+1}\right] - E\left[H_{t+1}^{-1}v_{t+1})\right] && \text{(A.40)}\\
&= E\left[H_{t+1}^{-1}z_{t+1}\right] && \text{(A.41)}\\
\hat{x}_{t+1} &= H_{t+1}^{-1}z_{t+1} && \text{(A.42)}
\end{aligned}$$

The covariance matrix, Σ_{t+1}, of this result can be computed (note that in the following subscripts are removed for the derivation)

$$\begin{aligned}
\Sigma_{t+1} &= E\left[e_{t+1}e_{t+1}^T\right] && \text{(A.43)}\\
&= E\left[(x-\hat{x})(x-\hat{x})^T\right] && \text{(A.44)}\\
&\text{Substituting } x = H^{-1}(z-v) \text{ and } \hat{x} = H^{-1}z \text{ gives}\\
&= E\left[(H^{-1}(z-v) - H^{-1}z)(H^{-1}(z-v) - H^{-1}z)^T\right] && \text{(A.45)}\\
&= E\left[(H^{-1}z - H^{-1}v - H^{-1}z)(H^{-1}z - H^{-1}v - H^{-1}z)^T\right] && \text{(A.46)}\\
&= E\left[(-H^{-1}v)(-H^{-1}v)^T\right] && \text{(A.47)}\\
&= E\left[(H^{-1}v)(H^{-1}v)^T\right] && \text{(A.48)}\\
&= E\left[H^{-1}vv^T H^{-T}\right] && \text{(A.49)}\\
&= H^{-1}E\left[vv^T\right] H^{-T} && \text{(A.50)}\\
&\text{Noting that } E[vv^T] = R\\
\Sigma_{t+1} &= H_{t+1}^{-1}R_{t+1}H_{t+1}^{-T} && \text{(A.51)}
\end{aligned}$$

Here we have described two versions of covariance for the state at time $t+1$. Namely $\Sigma_{t+1} = H_{t+1}^{-1} R_{t+1} H_{t+1}^{-T}$ which incorporates the measurement model and the predicted covariance $P_{t+1}^{-} = A P_t A^T + Q$ which incorporates the system dynamics. The best estimate of the combined state is a weighted average of the two. In a least-squares scenario, we wish to estimate the state which minimizes the following error function

$$\chi^2 = \sum_i \frac{(x_i - \hat{x}_i)^2}{\sigma_i^2} \tag{A.52}$$

In order to find the minimum value of $\hat{x}$ we take the derivative with respect to $\hat{x}$, set it to zero and solve for $\hat{x}$. It is simpler to see the case of two values, since this is most commonly used in the derivation to follow. Thus, given two values of the state x_1 and x_2, and their variances σ_1 and σ_2 respectively, we wish to estimate the best estimate $\hat{x}$ that minimizes the function

$$\chi^2 = \frac{(x_1 - \hat{x})^2}{\sigma_2^2} + \frac{(x_2 - \hat{x})^2}{\sigma_2^2} \tag{A.53}$$

First, take the derivative and set it to zero

$$\frac{d}{d\hat{x}} \left[\frac{(x_1 - \hat{x})^2}{\sigma_2^2} + \frac{(x_2 - \hat{x})^2}{\sigma_2^2} \right] = 0 \tag{A.54}$$

$$\frac{d}{d\hat{x}} \left[\frac{(x_1 - \hat{x})^2}{\sigma_2^2} \right] + \left[\frac{d}{d\hat{x}} \frac{(x_2 - \hat{x})^2}{\sigma_2^2} \right] = 0 \tag{A.55}$$

$$\frac{2}{\sigma_1^2}(x_1 - \hat{x}) \frac{d}{d\hat{x}}(x_1 - \hat{x}) + \frac{2}{\sigma_2^2}(x_2 - \hat{x}) \frac{d}{d\hat{x}}(x_2 - \hat{x}) = 0 \tag{A.56}$$

$$2\frac{\hat{x} - x_1}{\sigma_1^2} + 2\frac{\hat{x} - x_2}{\sigma_2^2} = 0 \tag{A.57}$$

and now solving for $\hat{x}$ gives

$$2\left(\frac{\hat{x}}{\sigma_1^2} - \frac{x_1}{\sigma_1^2} + \frac{\hat{x}}{\sigma_2^2} - \frac{x_2}{\sigma_2^2} \right) = 0 \tag{A.58}$$

$$2\hat{x}\left(\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2} \right) - 2\left(\frac{x_1}{\sigma_1^2} + \frac{x_2}{\sigma_2^2} \right) = 0 \tag{A.59}$$

$$\hat{x}\left(\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2} \right) = \left(\frac{x_1}{\sigma_1^2} + \frac{x_2}{\sigma_2^2} \right) \tag{A.60}$$

Finally, the expression for $\hat{x}$ can be given as

$$\hat{x} = \frac{\frac{x_1}{\sigma_1^2} + \frac{x_2}{\sigma_2^2}}{\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2}} = \left(\frac{x_1}{\sigma_1^2} + \frac{x_2}{\sigma_2^2} \right) \left(\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2} \right)^{-1} \tag{A.61}$$

This is the value of $\hat{x}$ that minimizes the chi-squared cost function and thus minimizes the error given two estimates of the same state with different variances. In a similar vain, we want to find an estimate of the variance that minimizes the error as well. Thus we wish to compute the following

$$E[(\hat{x}-\mu)^2] = E\left[\left(\frac{\frac{x_1}{\sigma_1^2}+\frac{x_2}{\sigma_2^2}}{\frac{1}{\sigma_1^2}+\frac{1}{\sigma_2^2}} - \mu\frac{\frac{1}{\sigma_1^2}+\frac{1}{\sigma_2^2}}{\frac{1}{\sigma_1^2}+\frac{1}{\sigma_2^2}}\right)^2\right] \tag{A.62}$$

$$= E\left[\left(\frac{\frac{x_1-\mu}{\sigma_1^2}+\frac{x_2-\mu}{\sigma_2^2}}{\frac{1}{\sigma_1^2}+\frac{1}{\sigma_2^2}}\right)^2\right] \tag{A.63}$$

$$= \frac{1}{\left(\frac{1}{\sigma_1^2}+\frac{1}{\sigma_2^2}\right)^2} E\left[\left(\frac{x_1-\mu}{\sigma_1^2}+\frac{x_2-\mu}{\sigma_2^2}\right)^2\right] \tag{A.64}$$

$$\text{Define } N = \frac{1}{\left(\frac{1}{\sigma_1^2}+\frac{1}{\sigma_2^2}\right)^2}$$

$$= N\left(E\left[\left(\frac{x_1-\mu}{\sigma_1^2}\right)^2\right] + 2E\left[\frac{x_1-\mu}{\sigma_1^2}\frac{x_2-\mu}{\sigma_2^2}\right] + E\left[\left(\frac{x_2-\mu}{\sigma_2^2}\right)^2\right]\right) \tag{A.65}$$

Since we are talking about errors, the second $E[\cdot]$ term disappears and also, we notice that $(x_1-\mu)^2 = \sigma_1$ and $(x_2-\mu)^2 = \sigma_2$

$$= N\left(E\left[\frac{\sigma_1^2}{\sigma_1^4}\right] + 0 + E\left[\frac{\sigma_2^2}{\sigma_2^4}\right]\right) \tag{A.66}$$

$$= N\left(E\left[\frac{1}{\sigma_1^2}\right] + E\left[\frac{1}{\sigma_2^2}\right]\right) \tag{A.67}$$

$$= \left(\frac{1}{\left(\frac{1}{\sigma_1^2}+\frac{1}{\sigma_2^2}\right)^2}\right)\left(\frac{1}{\sigma_1^2}+\frac{1}{\sigma_2^2}\right) \tag{A.68}$$

$$= \frac{1}{\frac{1}{\sigma_1^2}+\frac{1}{\sigma_2^2}} \tag{A.69}$$

$$= \left((\sigma_1^2)^{-1} + (\sigma_2^2)^{-1}\right)^{-1} \tag{A.70}$$

In general, Equation A.70 is the "best" estimate (in a least squares sense) of the variance given two variance estimates. This can also be written more compactly in multidimensional form using matrix notation; given two covariance matrices (Σ_1, Σ_2) the best covariance estimate is therefore

$$\left(\Sigma_1^{-1} + \Sigma_2^{-1}\right)^{-1} = \Sigma_1 (\Sigma_1 + \Sigma_2)^{-1} \Sigma_2 \tag{A.71}$$

Hence, we now wish to combine the two estimates of the covariance that we have already computed above; namely $\Sigma_{t+1} = H_{t+1}^{-1} R_{t+1} H_{t+1}^{-T}$ and $P_{t+1}^{-} = AP_tA^T + Q$. We will denote the "best" estimate using these matrices, $\hat{P}_{t+1}$, i.e. the corrected version of the covariance matrix at time $t+1$ since it will incorporate the measurement model uncertainty and the system dynamics uncertainty. For simplicity of the derivation, we drop the $t+1$ subscripts since they are on each term.

$$\hat{P} = P^- \left(P^- + \Sigma\right)^{-1} \Sigma \tag{A.72}$$

$$= P^- \left(P^- + H^{-1}RH^{-T}\right)^{-1} H^{-1}RH^{-T} \tag{A.73}$$

Inserting Identity Matrix: $I = H^{-1}H$

$$= P^- \left(P^- + H^{-1}RH^{-T}\right)^{-1} \mathbf{H^{-1}H} H^{-1}RH^{-T} \tag{A.74}$$

Using the matrix identity: $(AB)^{-1} = (B^{-1}A^{-1})$

$$= P^- \left(H(P^- + H^{-1}RH^{-T})\right)^{-1} RH^{-T} \tag{A.75}$$

$$= P^- \left(HP^- + HH^{-1}RH^{-T}\right)^{-1} RH^{-T} \tag{A.76}$$

$$= P^- \left(HP^- + RH^{-T}\right)^{-1} RH^{-T} \tag{A.77}$$

Inserting Identity Matrix: $I = H^T H^{-T}$

$$= P^- \mathbf{H^T H^{-T}} \left(HP^- + RH^{-T}\right)^{-1} RH^{-T} \tag{A.78}$$

Using the matrix identity: $B^{-1}A^{-1} = (AB)^{-1}$

$$= P^- H^T \left((HP^- + RH^{-T})H^T\right)^{-1} RH^{-T} \tag{A.79}$$

$$= P^- H^T \left(HP^- H^T + RH^{-T}H^T\right)^{-1} RH^{-T} \tag{A.80}$$

$$\hat{P} = P^- H^T \left(HP^- H^T + R\right)^{-1} RH^{-T} \tag{A.81}$$

Now, we can define the Kalman gain to be

$$K_{t+1} = P^-_{t+1} H^T_{t+1} \left(H_{t+1} P^-_{t+1} H^T_{t+1} + R_{t+1}\right)^{-1} \tag{A.82}$$

finally, the covariance update equation can be expressed fully as

$$\hat{P}_{t+1} = K_{t+1} R H^{-T}_{t+1} \tag{A.83}$$

This can also be expressed differently through some matrix manipulation as

$$\hat{P}_{t+1} = P^-_{t+1} - K_{t+1} H_{t+1} P^-_{t+1} \tag{A.84}$$

$$= (I - K_{t+1} H_{t+1}) P^-_{t+1} \tag{A.85}$$

Equation A.61 allows us to estimate the "best" estimate of two states given variance weightings. This for simpler notation can be expressed in matrix notation as

$$\left(\Sigma_1^{-1} + \Sigma_2^{-1}\right)^{-1} \left(\Sigma_1^{-1} x_1 + \Sigma_2^{-1} x_2\right) \tag{A.86}$$

Using a similar approach, a correction to the state can be made since we have the estimate of the state from measurements and the estimate of the predicted state. Thus, we need to combine the estimates of Equation A.42 ($\hat{x}_{t+1} = H^{-1}_{t+1} z_{t+1}$) and Equation A.18 ($\hat{x}_{t+1} = A_t \hat{x}_t + B_t u_t$) The best estimate of the combined states can be simplified into the final form

$$\hat{x}_{t+1} = \hat{x}^-_{t+1} + K_{t+1}(z_{t+1} - H_{t+1}\hat{x}_{t+1}) \tag{A.87}$$

Bibliography

[1] P. Besl and N. McKay. A method for registration of 3-d shapes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 14(2):239–256, February 1992.

[2] M. Bosse, P. Newman, J. Leonard, and S. Teller. An Atlas Framework for Scalable Mapping. In *Robotics and Automation, Proceedings ICRA'03, IEEE International Conference on*, 2003.

[3] J. Buhmann, W. Burgard, A. Cremers, D. Fox, T. Hofmann, F. Schneider, J. Strikos, and S. Thrun. The Mobile Robot Rhino. *AI Magazine*, 16(1), Spring 1995.

[4] W. Burgard, A. Cremers, D. Fox, D. Hahnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. Experiences with an interactive museum tour-guide robot. *Artificial Intelligence*, 114(1-2), 2000.

[5] G. Casella and C. Robert. Rao-Blackwellisation of Sampling Schemes. *Biometrika*, 83(1):81–94, March 1996.

[6] R. Chatila and J. Laumond. Position Referencing and Consistent World Modeling for Mobile Robots. In *Robotics and Automation. Proceedings ICRA'85. IEEE International Conference on*, pages 138 – 145. IEEE Computer Society Press, March 1985.

[7] H. Choset and K. Nagatani. Topological simultaneous localization and mapping (SLAM): toward exact localization without explicit localization. *Robotics and Automation, IEEE Transactions on*, 17(2):125–137, April 2001.

[8] I. Cox. Blanche-an experiment in guidance and navigation of an autonomous robot vehicle. *Robotics and Automation, IEEE Transactions on*, 7(2):193 – 204, April 1991.

[9] I. Cox. A review of statistical data association techniques for motion correspondence. *International Journal of Computer Vision*, 10(1):53–66, 1993.

[10] M. Csorba. *Simultaneous Localisation and Map Building*. PhD thesis, University of Oxford, 1997.

[11] A. Davison. Real-Time Simultaneous Localisation and Mapping with a Single Camera. In *Computer Vision. Proceedings ICCV'03. The 9th International Conference on*, 2003.

[12] A. Davison, Y. Cid, and N. Kita. Real-Time 3D SLAM with Wide-Angle Vision. In *Intelligent Autonomous Vehicles. Proceedings IAV'04. The 5th IFAC/EURON Symposium on*, July 5-7 2004.

[13] G. Dissanayake and P. Newman. A solution to the simultaneous localization and map building (SLAM) problem. *Robotics and Automation, IEEE Transactions on*, 17(3):229–241, June 2001.

[14] G. Dissanayake, P. Newman, H. Durrant-Whyte, S. Clark, and M. Csorba. Lecture Notes in Control and Information Sciences. *Experimental Robotics VI*, 2000.

[15] A. Doucet, N. Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, January 2001.

[16] A. Doucet, N. Freitas, K. Murphy, and S. Russell. Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks. In *Uncertainty in Artificial Intelligence, Proceedings UAI2000, The 16th Conference on*, 2000.

[17] H. Durrant-Whyte, E. Nebot, and G. Dissanayake. Autonomous Localisation and Map building in Large-Scale Unstructured Environments. In *Proc. Learning 2001*, April 2001.

[18] A. Elfes. *Occupancy grids: a probabilistic framework for robot perception and navigation*. PhD thesis, Carnegie Mellon University, 1989.

[19] A. Elfes. Using Occupancy Grids for Mobile Robot Perception and Navigation. *Computer*, 22(6):46–57, 1989.

[20] A. Eliazar and R. Parr. DP-SLAM: Fast, Robust Simultaneous Localization and Mapping Without Predetermined Landmarks. In *Artificial Intelligence. Proceedings IJCAI. 18th International Joint Conference on*, August 9-15 2003.

[21] A. Eliazar and R. Parr. DP-SLAM 2.0. In *Proceedings. ICRA '04. 2004 IEEE International Conference on*, pages 1314 – 1320. IEEE Computer Society Press, April 26-May 1 2004.

[22] W. Feller. *An Introduction to Probability and Its Applications*. John Wiley, third edition, 1968.

[23] M. Fischler and R. Bolles. Random sample consensus: A paradigm for model fitting with application to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–385, 1981.

[24] D. Fox. *Markov Localization: A Probabilistic Framework for Mobile Robot Localization and Navigation*. PhD thesis, University of Bonn, December 1998.

[25] D. Fox, S. Thrun, F. Dellaert, and W. Burgard. Particle filters for mobile robot localization. *Sequential Monte Carlo Methods in Practice*, 2000.

[26] U. Frese and G. Hirzinger. Simultaneous Localization and Maping - A Discussion. In *Reasoning with Uncertainty in Robotics. Proceedings of the IJCAI Workshop on*, August 4-5 2001.

[27] M. Garcia and A. Solanas. 3D simultaneous localization and modeling from stereo vision. In *Robotics and Automation. Proceedings ICRA'04. IEEE International Conference on*, pages 847–853. IEEE Computer Society Press, Apr. 26-May 1 2004.

[28] C. Grinstead and J. Snell. *Introduction to Probability*. Dartmouth Chance Project, 2005.

[29] G. Grisetti, C. Stachniss, and W. Burgard. Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling. In *Robotics and Automation, Proceedings ICRA '05, IEEE International Conference on*. IEEE Computer Society Press, Apr. 18-22 2005.

[30] J. Guivant. *Efficient Simultaneous Localization and Mapping in Large Environments*. PhD thesis, ACFR - University of Sydney, May 2002.

[31] J. Guivant and E. Nebot. Optimization of the Simultaneous Localization and map Building Algorithm for Real Time Implementation. *Robotics and Automation, IEEE Transactions on*, 17(3):242–257, June 2001.

[32] J. Guivant and E. Nebot. Improving computational and memory requirements of simultaneous localization and map building algorithms. In *Robotics and Automation. Proceedings ICRA '02. IEEE International Conference on*, pages 2731 – 2736. IEEE Computer Society Press, May 2002.

[33] J. Guivant, E. Nebot, J. Nieto, and F. Masson. Navigation and Mapping in Large Unstructured Environments. *International Journal of Robotics Research*, 23(4-5):449–472, April-May 2004.

[34] J. Gutmann and K. Konolige. Incremental Mapping of Large Cyclic Environments. In *Computational Intelligence in Robotics and Automation. Proceedings CIRA'99. Int'l Symposium on*, November 1999.

[35] D. Guzzoni, A. Cheyer, L. Julia, and K. Konolige. Many Robots Make Short Work. *AI Magazine*, 18(1):55–64, Spring 1997.

[36] D. Hahnel, W. Burgard, B. Wegbreit, and S. Thrun. Towards Lazy Data Association in SLAM. In *Proceedings of the 11th International Symposium of Robotics Research (ISRR'03)*, Sienna, Italy, 2003. Springer.

[37] D. Hahnel, D. Schulz, and W. Burgard. Map building with mobile robots in populated environments. In *Intelligent Robots and System. Proceedings IROS'02. IEEE/RSJ International Conference on*, pages 496 – 501. IEEE Computer Society Press, Sept 30-Oct 5 2002.

[38] D. Hahnel, R. Triebel, W. Burgard, and S. Thrun. Map building with mobile robots in dynamic environments. In *Robotics and Automation. Proceedings ICRA '03. IEEE International Conference on*, pages 1557 – 1563. IEEE Computer Society Press, Sept. 14-19 2003.

[39] M. Iosifescu. *Finite Markov Processes and Their Applications*. John Wiley, 1980.

[40] S. Julier and J. Uhlmann. A new extension of the Kalman filter to nonlinear systems. In *International Symposium on Aerospace/Defense Sensing, Simulation and Controls*, 1997.

[41] R. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME– Journal of Basic Engineering*, 82(D):35–45, 1960.

[42] J. Kemeny and J. Snell. *Finite Markov Chains*. Springer Verlag, New York, 1960.

[43] B. Kuipers and Y. Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Robotics and Autonomous Systems*, 8:47–63, November 1991.

[44] J. Leonard and H. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *Robotics and Automation, IEEE Transactions on*, 7(3):376 – 382, June 1991.

[45] J. Leonard and H. Durrant-Whyte. Simultaneous Map Building and Localization for an Autonomous Mobile Robot. In *Intelligent Robots and Systems. Proceedings IROS'91. IEEE/RSJ International Workshop on*, pages 1442–1447, May 1991.

[46] J. Liu, R. Chen, and T. Logvinenko. *A Theoretical Framework for Sequential Importance Sampling and Resampling*. Sequential Monte Carlo Methods in Practice. Springer Verlag, New York, 2001.

[47] D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[48] F. Lu and E. Milios. Robot pose estimation in unknown environments by matching 2D range scans. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94. IEEE Computer Society Conference on*, pages 935 – 938. IEEE Computer Society Press, June 1994.

[49] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4:333–349, 1997.

[50] P. Maybeck. *Stochastic Models, Estimation, and Control*, volume 1. Academic Press, Inc., 1979.

[51] M. Montemerlo. *FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem with Unknown Data Association*. PhD thesis, AAAI Press, Edmonton, Canada, June 2003.

[52] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem. In *Artificial Intelligence, Proc. of the AAAI National Conference on*, Edmonton, Canada, 2002. AAAI Press.

[53] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Provably Converges. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)*. IJCAI, 2003.

[54] M. Montemerlo, S. Thrun, and W. Whittaker. Conditional particle filters for simultaneous mobile robot localization and people-tracking. In *Robotics and Automation. Proceedings ICRA '02. IEEE International Conference on*, pages 695 – 701. IEEE Computer Society Press, May 11-12 2002.

[55] H. Moravec and A. Elfes. High-Resolution Maps from Wide-Angle Sonar. In *Robotics and Automation, Proceedings ICRA '85, IEEE International Conference on*, Los Alamitos, Calif., 1985. CS Press.

[56] K. Murphy. Bayesian Map Learning in Dynamic Environments. In *Neural Information Processing Systems. Proceedings NIPS'99. 13th Conference on*, Nov. 29-Dec. 4 1999.

[57] K. Murphy and S. Russell. *Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks*. Sequential Monte Carlo Methods in Practice. Springer Verlag, New York, 2001.

[58] E. Nettleton, P. Gibbens, and H. Durrant-Whyte. Closed form solutions to the multiple platform simultaneous localisation and map building (SLAM) problem. In B. Dasarathy, editor, *Sensor Fusion: Architectures, Algorithms,and Applications IV*, pages 428–437, 2000.

[59] J. Nieto, J. Guivant, and E. Nebot. The HYbrid metric maps (HYMMs): a novel map representation for DenseSLAM. In *Robotics and Automation. Proceedings ICRA '04. IEEE International Conference on*, pages 391 – 396. IEEE Computer Society Press, Apr 26 - May 1 2004.

[60] J. Nieto, J. Guivant, E. Nebot, and S. Thrun. Real time data association for FastSLAM. In *Robotics and Automation. Proceedings ICRA '03. IEEE International Conference on*, pages 412–418, Sept 14-19 2003.

[61] M. Paskin. Thin Junction Tree Filters for Simultaneous Localization and Mapping. Computer Science Division Technical Report CSD-02-1198, University of California, Berkeley, September 2002.

[62] M. Paskin. Thin Junction Tree Filters for Simultaneous Localization and Mapping. In J. Liu, editor, *Artificial Intelligence. Proceedings IJCAI'03. 18th International Joint Conference on*, pages 1157–1164, San Francisco, CA, 2003. Morgan Kaufmann Publishers.

[63] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes in C*. Cambridge University Press, 2002.

[64] I. Rekleitis. *Cooperative Localization and Multi-robot exploration*. PhD thesis, McGill University, Montreal, QC, Canada, January 2003.

[65] B. Ristic, S. Arulampalam, and N. Gordon. *Beyond the Kalman Filter - Particle Filters for Tracking Applications*. Artech House, 2004.

[66] B. Schiele and J. Crowley. A Comparison of Position Estimation Techniques using Occupancy Grids. In *Robotics and Automation. Proceedings ICRA'94 IEEE International Conference on*, pages 1628 – 1634. IEEE Computer Society Press, May 8-13 1994.

[67] S. Se, D. Lowe, and J. Little. Mobile Robot Localization and Mapping with Uncertainty using Scale-Invariant Visual Landmarks. *International Journal of Robotics Research*, 21(8), August 2002.

[68] S. Se, H. Ng, P. Jasiobedzki, and T. Moyung. Vision based Modeling and Localization for Planetary Exploration Rovers. In *International Astronautical Congress. Proceedings IAC 2004.*, October 2004.

[69] S. Simhon and G. Dudek. A Global Topological Map formed by Local Metric Maps. In *Intelligent Robotic Systems, Proceedings IROS'98, IEEE/RSJ International Conference on*, pages 1708–1714, October 1998.

[70] R. Smith and P. Cheeseman. On the representation and estimation of spatial uncertainly. *International Journal of Robotics Research*, 5(4):56–68, 1987.

[71] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. *Autonomous Robot Vehicles*, pages 167–193, 1990.

[72] R. Smith, M. Self, and P. Cheeseman. A Stochastic Map for Uncertain Spatial Relationships. In S. Iyengar and A. Elfes, editors, *Autonomous Mobile Robots: Perception, Mapping, and Navigation*, pages 323–330. IEEE Computer Society Press, 1991.

[73] H. Stark and J. Woods. *Probability and Random Processes with Applications to Signal Processing*. Prentice-Hall, 2002.

[74] S. Thrun. Learning Metric-Topological Maps for Indoor Mobile Robot Navigation. *Artificial Intelligence*, 99:1:21–71, 1998.

[75] S. Thrun. Probabilistic Algorithms in Robotics. *AI Magazine*, 21(4):93–109, 2000.

[76] S. Thrun. *Robotic Mapping: A Survey*. Exploring Artificial Intelligence in the New Millenium. Morgan Kaufmann, 2002.

[77] S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. Probabilistic Algorithms and the Interactive Museum Tour-Guide Robot Minerva. *International Journal of Robotics Research*, 19(11):972–999, 2000.

[78] S. Thrun, D. Fox, and W. Burgard. A Probabilistic Approach to Concurrent Mapping and Localization for Mobile Robots. *Machine Learning*, 31:29–53, 1998. also appeared in Autonomous Robots 5, 253-271 (joint issue).

[79] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust Monte Carlo Localization for Mobile Robots. *Artificial Intelligence*, 128(1-2):99–141, 2000.

[80] S. Thrun, Y. Liu, D. Koller, A. Ng, and Z. Ghahramani. Simultaneous Localization and Mapping with Sparse Extended Information Filters. *International Journal of Robotics Research*, 23:693–716, 2004.

[81] E. Wan and R. Merwe. *The Unscented Kalman Filter*. Kalman Filtering and Neural Networks. Wiley Publishing, 2001.

[82] C. Wang. *Simultaneous Localization, Mapping and Moving Object Tracking*. PhD thesis, Carnegie Mellon University, April 2004.

[83] G. Welch and G. Bishop. An Introduction to the Kalman Filter. Technical Report TR95-041, University of North Carolina at Chapel Hill (UNC), 1995.

[84] D. Wolf and G. Sukhatme. Online simultaneous localization and mapping in dynamic environments. In *Robotics and Automation. Proceedings ICRA ’04. 2004 IEEE International Conference on*, pages 1301 – 1307. IEEE Computer Society Press, Apr. 26-May 1 2004.