FastSLAM: An Efficient Solution to the Simultaneous Localization And Mapping Problem with Unknown Data Association

Sebastian Thrun[1], **Michael Montemerlo**[1], **Daphne Koller**[1], **Ben Wegbreit**[1]
Juan Nieto[2], **and Eduardo Nebot**[2]

[1]Computer Science Department
Stanford University

[2]Australian Centre for Field Robotics
The University of Sydney, Australia

Abstract

This article provides a comprehensive description of FastSLAM, a new family of algorithms for the simultaneous localization and mapping problem, which specifically address hard data association problems. The algorithm uses a particle filter for sampling robot paths, and extended Kalman filters for representing maps acquired by the vehicle. This article presents two variants of this algorithm, the original algorithm along with a more recent variant that provides improved performance in certain operating regimes. In addition to a mathematical derivation of the new algorithm, we present a proof of convergence and experimental results on its performance on real-world data.

1 Introduction

The simultaneous localization and mapping (SLAM) problem has received tremendous attention in the robotics literature. The SLAM problem involves a moving vehicle attempting to recover a spatial map of its environment, while simultaneously estimating its own pose (location and orientation) relative to the map. SLAM problems arises in the navigation of mobile robots through unknown environments for which no accurate map is available. Since robot motion is subject to error, the mapping problem necessarily induces a robot localization problem—hence the name SLAM. Applications of SLAM include indoors [8, 68], outdoors [3], underwater [73], underground [69, 62], and planetary exploration [36, 71]. The number of robots that use maps for navigation is long [7, 10, 13, 32, 33].

Mapping problems come at varying degrees of difficulty. In the most basic case, the vehicle has access to a global positioning system (GPS) which provides it with accurate pose information. The problem of acquiring a map with known robot poses [49, 66] is significantly easier than the general SLAM problem. When GPS is unavailable, as is the case indoors, underground, or underwater, the vehicle will inevitably accrue pose errors during mapping. Such pose errors have the displeasing side-effect that they induce systematic errors in the map. SLAM addresses this very problem of acquiring a map without an external source of vehicle pose information.

The dominant approach to the SLAM problem was introduced in a seminal paper by Smith, Self, and Cheeseman [64]. It was first developed into an implemented system by Moutarlier and Chatila [50, 51]. This approach uses an extended Kalman filter (EKF) for estimating the posterior distribution over the map and the robot pose. The EKF approach represents the vehicle's internal map (and the robot pose estimate) by a high-dimensional Gaussian, over all features in the map and the vehicle pose. The off-diagonal elements in the covariance matrix of this multivariate Gaussian represent the correlations between errors in the vehicle pose and the features in the map. As a result, the EKF can accommodate the correlated nature of errors in the map. The EKF approach has been the basis of many recent developments in the field [17, 34].

One limitation of the EKF approach is computational in nature. Maintaining a multivariate Gaussian requires time *quadratic* in the number of features in the map. This limitation has been recognized, and a number of more efficient approaches has been proposed [3, 11, 25, 35, 56, 57, 70]. The common idea underlying most of these approaches is to decompose the problem of building one large map into a collection of smaller maps, which can be updated more efficiently. Depending on the nature of the local maps and the mechanics of tracing dependencies among them, the resulting savings range from a much reduced constant factor to implementations that require constant update time [35, 56, 70].

A second and more important limitation of the EKF approach is related to the *data association problem*, also known as the *correspondence problem* [4, 14]. The data association problem arises when different features in the environment look alike. In such cases, different data association hypotheses induce multiple, distinct looking maps. Gaussians cannot represent such multi-modal distributions. The standard approach in the SLAM literature is to restrict the inference to the most plausible of these map hypotheses, incorporating only the most likely data association given the robot's current map. The determination of the most likely data association may be performed on a per-measurement basis [17], or it may incorporate multiple measurements at a time [3, 53]. The latter approach is more robust; however, both approaches tend to fail catastrophically when the alleged data association is incorrect. Alternative approaches exist that interleave data association decisions with map building in a way that enables them to revise past data association decisions, such as the RANSAC algorithm [22], the expectation maximization approach [63, 68], or approaches based on MCMC techniques [1]. However, such techniques cannot be executed in real-time and are therefore of lesser relevance to the problems studied here.

This article describes a family of algorithms called FastSLAM [27, 46]. FastSLAM is a SLAM algorithm that integrates particle filters [18, 37] and extended Kalman filters. It exploits a structural property of the SLAM problem first pointed out by Murphy [52]: feature estimates are conditional independent given the robot path. More specifically, correlations in the uncertainty among different map features arise *only* through robot pose uncertainty. If the robot was told its correct path, the errors in its feature estimates would be independent of each other. This observation allows us to define a factored representation of the posterior over poses and maps. FastSLAM implements such a factored representation, using particle filters for estimating the robot path. Conditioned on these particles the individual map errors are independent, hence the mapping problem can be factored into separate problems, one for each feature in the map. FastSLAM estimates these feature locations by EKFs. The basic algorithm can be implemented in time logarithmic in the number of landmarks, using efficient tree representations of the map [45]. Hence, FastSLAM offers computational advantages over plain EKF implementations and many of its descendants.

The key advantage of FastSLAM , however, is the fact that data association decisions can be made on a per-particle basis, similar to multi-hypothesis tracking algorithms [60]. As a result, the filter maintains posteriors over multiple data associations, not just the most likely one. As shown empirically, this feature makes FastSLAM significantly more robust to data association problems than algorithms based on maximum likelihood data association. A final, advantage of FastSLAM over EKF-style approaches arises from the fact that particle filters can cope with non-linear robot motion models, whereas EKF-style techniques approximate such models via linear functions.

This article describes two instantiations of the FastSLAM algorithm, referred here to FastSLAM 1.0 and 2.0. FastSLAM 1.0 is the original FastSLAM algorithm [45], which is conceptually simple and easy to implement. In certain situations, however, the particle filter component of FastSLAM 1.0 generates samples inefficiently. The algorithm FastSLAM 2.0 [45] overcomes this problem through an improved proposal distribution, but at the expense of an implementation that is significantly more involved (as is the mathematical derivation). For both algorithms, we provide techniques for estimating data association in SLAM [44, 55]. The derivation of all algorithms is carried out using probabilistic notation; however, the resulting expressions will be provided using linear algebraic equations familiar from the filtering literature. We offer a proof of convergence in expectation for FastSLAM 2.0 in linear-Gaussian SLAM. Further, we provide extensive experimental results using real-world data. We show empirically that the FastSLAM 2.0 outperforms EKFs in situations plagued with hard data association problems, thanks to its ability to pursue multiple data association hypotheses simultaneously. We also provide experimental results for learning maps with as many as 10^6 features, which is orders of magnitude larger than the largest maps ever built with EKFs.

2 The SLAM Problem

The SLAM problem is defined as the problem of recovering a map and a robot pose (location and orientation) from data acquired by a mobile robot. The robot gathers information about nearby landmarks, and it also measures its own motion. Both types of measurements are subject to noise. They are compiled into a probabilistic estimate of the map along with the robot's momentary pose (location and orientation).

Figure 1 illustrates the SLAM problem graphically. Panel (a) shows the uncertainty accrued along a robot's path, along with the uncertainty in the location of all features seen thus far. As this graphic illustrates, the robot's pose uncertainty increases over time, as does its estimate on the absolute location of individual features. A key characteristic of the SLAM problem is highlighted in Figure 1b: Here the robot senses a previously observed landmark whose position is relatively well known. This observation provides the robot with information about its momentary position. It also increases its knowledge of other feature locations in the map, which leads to a reduction of map uncertainty as indicated in Figure 1b. Notice that while in principle, the robot could also improve its estimate of past poses, it is common in SLAM no to consider past poses so as to keep the amount of computation independent on the length of the robot's history.

To describe SLAM more formally, let us denote the map by Θ. The map consists of a collection of features, each of which will be denoted θ_n. The total number of stationary features will be denoted N. The robot pose is defined at s_t, where t is a discrete time index. Poses of robots operating on the plane typically comprise the robot's two-dimensional Cartesian coordinates, along with its angular orientation. The sequence $s^t = s_1, s_2, \ldots, s_t$ denotes the path of the robot up to time t. Throughout this article, we will use the superscript s^t to denote sequence of variables from time 1 up to time t.

To acquire a map, the robot can sense. Sensor measurements convey information about the range, distance, appearance etc. of nearby features. This is illustrated in Figure 2, in which a robot measures the range and bearing to a nearby landmark. Without loss of generality, we assume that the robot observes exactly one landmark at a time. The measurement at time t, denoted z_t, may be the range and bearing of a nearby feature. The assumption of observing a single feature at a time is adopted for convenience; multiple feature sightings are easily processed sequentially. Highly restrictive, however, is an assumption that we will initially adopt but eventually drop in later sections of this paper, namely that the robot can determine the identify of each feature. For each measurement z_t, n_t specifies the identity of the observed feature. The range of the correspondence variable n_t is the finite set $\{1, \ldots, N\}$.

At the core of our SLAM algorithm is a generative model of sensor measurements, that is, a probabilistic law that specifies the process according to which measurements are generated. This model will be referred to as *measurement model* and is of the following form:

$$p(z_t \mid s_t, \theta_{n_t}, n_t) \quad = \quad g(\theta_{n_t}, s_t) + \varepsilon_t \tag{1}$$

The measurement model is conditioned on the robot pose s_t, the landmark identity n_t, and the specific feature θ_{n_t} that is being observed. It is governed by a (deterministic) function g distorted by noise. The noise at time t is modeled by the random variable ε_t, which will be assumed to be normally distributed with mean zero and covariance R_t. The Gaussian noise assumption is usually just an approximation,

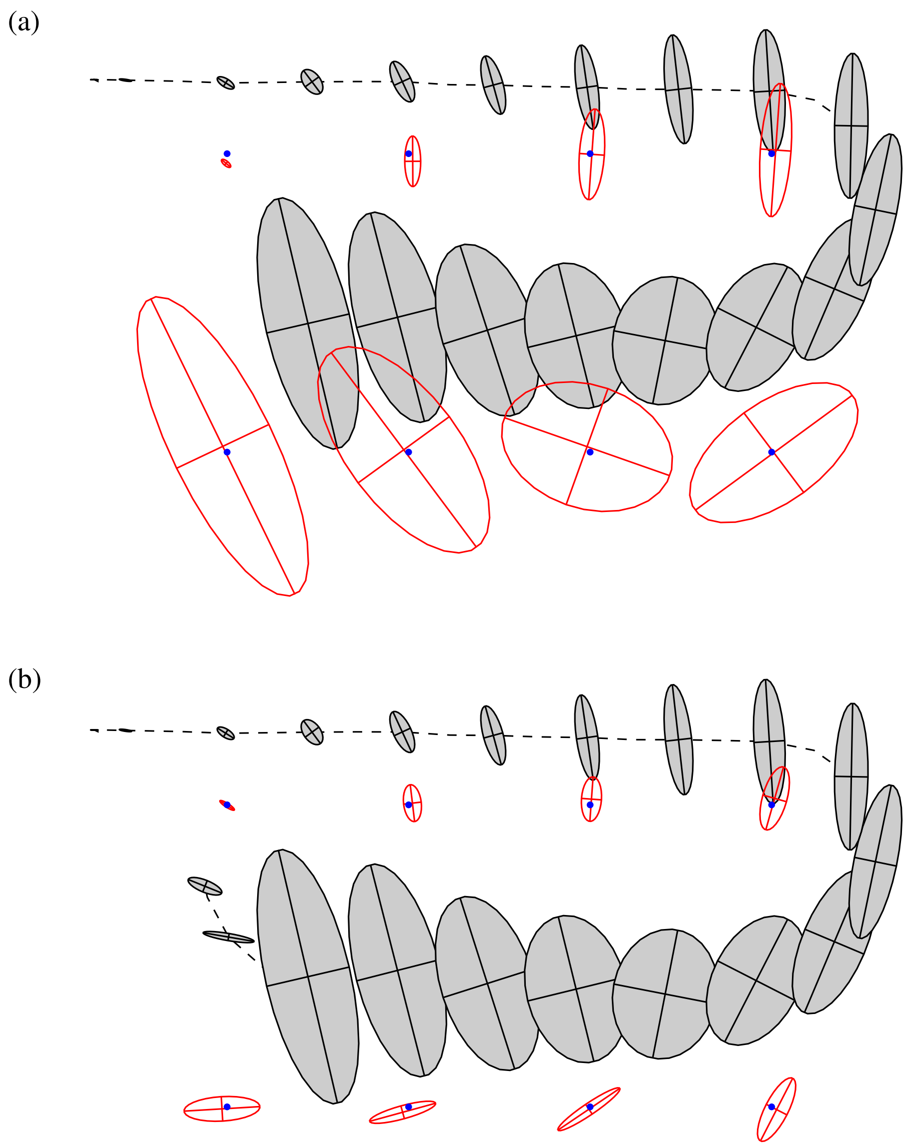

Figure 1: The SLAM problem: (a) A robot navigates through unknown terrain; as it progresses, its own pose uncertainty increases, as indicated by the shaded ellipses along the robot's path, and so does the uncertainty in the map (red ellipses). (b) Loop closure: Revisiting a previously seen landmark leads to a reduction in the uncertainty of the momentary pose and all landmarks. In online SLAM algorithms, this reduction is usually only applied to the momentary pose.

but one that tends to work well across a range of sensors [64, 17]. The measurement function g is generally nonlinear in its arguments. A common example is that of range and bearing measurement, as discussed above. The range (distance) and bearing (angle) to a landmarks are easily calculated through simple trigonometric functions that are non-linear in the coordinate variables of the robot and the sensed feature.

A second source of information for solving SLAM problems are the controls of the vehicle. Controls are denoted u_t, and refer to the collective motor commands carried out in the time interval $[t-1, t)$. The probabilistic law governing the evolution of poses is commonly referred to as *kinematic motion model*,

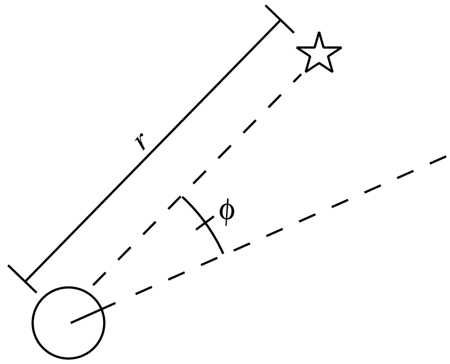

Figure 2: Vehicle observing the range and bearing to a nearby landmark.

and will be assumed to be of the following form:

$$p(s_t \mid u_t, s_{t-1}) \quad = \quad h(u_t, s_{t-1}) + \delta_t \tag{2}$$

As this expression suggests, the pose at time t is a function h of the robot's pose one time step earlier, distorted by Gaussian noise. The latter is captured by the random variable δ_t, whose mean is zero and whose covariance will be denoted by P_t. As was in the case of the measurement model, the function h is usually nonlinear in its argument.

The goal of SLAM is the recovery of the map from sensor measurements z^t and controls u^t. Most SLAM algorithms are instances of Bayes filters [29], and as such recover at any instant in time a probability distribution over the map Θ and the momentary robot pose s_t:

$$p(s_t, \Theta \mid z^t, u^t, n^t) \tag{3}$$

If this probability is calculated recursively from earlier probabilities of the same kind, the estimation algorithm is a filter. Most SLAM algorithms are instantiations of the Bayes filter, which computes this posterior from the one calculated one time step earlier (see [65] for a derivation):

$$\begin{aligned} & p(s_t, \Theta \mid z^t, u^t, n^t) \\ & \quad = \quad \eta\, p(z_t \mid s_t, \theta_{n_t}, n_t) \int p(s_t \mid s_{t-1}, u_t)\, p(s_{t-1}, \Theta \mid z^{t-1}, u^{t-1}, n^{t-1})\, ds_{t-1} \end{aligned} \tag{4}$$

Here η is a normalization constant (which is equivalent to $p(z_t \mid z^{t-1}, u^t, n^t)$ in this equation). The normalizer η does not depend on any of the variables over which the posterior is being computed. Throughout this article, we will adopt the common notation of using the letter η for generic normalization constants, regardless of their actual values.

The Bayes filter (4) is at the core of many contemporary SLAM algorithms. In cases where both g and h are linear, (4) is equivalent to the well-known Kalman filter [31, 40]. Extended Kalman filters (EKFs) allow for nonlinear functions g and h, but approximate those using a linear function, obtained through a first degree Taylor expansion. Taylor expansions are used by the seminal EKF algorithm for SLAM [64]. Other, less explored options for linearization include the unscented filter [30, 72] and moments matching [42].

At first glance, one might consider that the posterior (3) captures all relevant information, hence is should the "gold standard" for robotic SLAM. However, there are other, more elaborate distributions that can be estimated in SLAM. The algorithm FastSLAM, in particular, estimates a posterior over robot paths, not just momentary poses, along with the map:

$$p(s^t, \Theta \mid z^t, u^t, n^t) \tag{5}$$

At first glance, estimating the entire path posterior might appear to be a questionable choice. As the path length increases, so does the space over which the posterior (5) is defined. Such a property seems to be at odds with the real-time execution of a filter. However, as we will see below, specific types of filters calculate posteriors over paths just as efficiently as over momentary poses. This alone, however, would barely serve as a motivation to prefer path posteriors over pose posteriors. The true motivation behind (5) arises from the fact that it can be decomposed into a product of much smaller terms—a topic that will be discussed in a separate section below.

The filter for calculating the posterior (5) is as follows:

$$p(s^t, \Theta \mid n^t, z^t, u^t) \quad = \quad \eta \, p(z_t \mid s_t, \theta_{n_t}, n_t) \, p(s_t \mid s_{t-1}, u_t) \, p(s^{t-1}, \Theta \mid n^{t-1}, z^{t-1}, u^{t-1}) \tag{6}$$

This update equation differs from the standard Bayes filter (4) in the absence of an integral sign: in particular, the pose at time $t-1$, s_{t-1}, is not integrated out. Its derivation is mostly analogous to that of the regular Bayes filter (4), as provided in [65]. Bayes rule enables us to transform the left-hand side of (6) into the following product:

$$p(s^t, \Theta \mid n^t, z^t, u^t) \quad = \quad \eta \, p(z_t \mid s^t, \Theta, n^t, z^{t-1}, u^t) \, p(s^t, \Theta \mid n^t, z^{t-1}, u^t) \tag{7}$$

We now exploit the fact that the measurement z_t depends only on three variables: the robot pose s_t at the time the measurement was taken and the identity n_t and location θ_{n_t} of the observed feature. Put into equations, we have

$$p(z_t \mid s^t, \Theta, n^t, z^{t-1}, u^t) \quad = \quad p(z_t \mid s_t, \theta_{n_t}, n_t) \tag{8}$$

Furthermore, the probability $p(s^t, \Theta \mid n^t, z^{t-1}, u^t)$ in (7) can be factored as follows:

$$p(s^t, \Theta \mid n^t, z^{t-1}, u^t) \quad = \quad p(s_t \mid s^{t-1}, \Theta, n^t, z^{t-1}, u^t) \, p(s^{t-1}, \Theta \mid n^t, z^{t-1}, u^t) \tag{9}$$

Both terms can greatly be simplified, by dropping variables that convey no information for the specific probability. In particular, knowledge of s_{t-1} and u_t are sufficient to predict s_t; all other variables in the first term on the right hand side of (9) carry no additional information and can therefore be omitted. Similarly, n_t and u_t carry no information about the posterior over s^{t-1} and Θ. Hence, we can re-write (9) as follows:

$$p(s^t, \Theta \mid n^t, z^{t-1}, u^t) \quad = \quad p(s_t \mid s^{t-1}, u_t) \, p(s^{t-1}, \Theta \mid n^{t-1}, z^{t-1}, u^{t-1}) \tag{10}$$

As the reader may easily verify, substituting this equation and (8) back into (7) yields the desired filter (6). This filter, and the posterior it represents, form the core of all FastSLAM algorithms.

3 Factoring the SLAM Posterior

A key mathematical insight pertains to the fact that the posterior (5) possess an important characteristic. This characteristic was first reported in [52] and later exploited in [2, 47] and various FastSLAM 2.0 papers [45, 44, 55]. It also was used in an earlier mapping algorithms [67] but was not made explicit at that time.

The insight is that the SLAM posterior can be written in the factored form give by the following product:

$$p(s^t, \Theta \mid n^t, z^t, u^t) \quad = \quad p(s^t \mid n^t, z^t, u^t) \prod_{n=1}^{N} p(\theta_n \mid s^t, n^t, z^t) \tag{11}$$

This factorization states that the calculation of the posterior over paths and maps can be decomposed into $N+1$ recursive estimators, an estimator over robot paths, $p(s^t \mid n^t, z^t, u^t)$, and N separate estimators

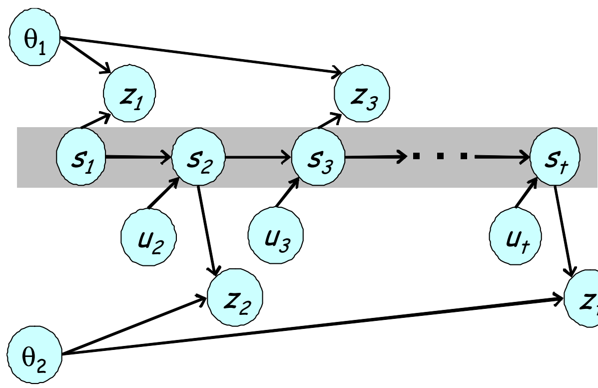

Figure 3: The SLAM problem: The robot moves from pose s_1 through a sequence of controls, $u_1, u_2, \ldots$. As it moves, it measures nearby landmarks. At time $t = 1$, it observes landmark θ_1 out of two landmarks, $\{\theta_1, \theta_2\}$. The measurement is denoted z_1 (range and bearing). At time $t = 1$, it observes the other landmark, θ_2, and at time $t = 3$, it observes θ_1 again. The SLAM problem is concerned with estimating the locations of the landmarks and the robot's path from the controls u^t and the measurements z^t. The gray shading illustrates a conditional independence relation.

over feature locations $p(\theta_n \mid s^t, n^t, z^t)$ conditioned on the path estimate, one for each $n = 1, \ldots, N$. The product of these probabilities represent the desired posterior in a factored way. This factored representation is exact, not just an approximation. It is a generic property of the SLAM problem.

To illustrate the correctness of this factorization, Figure 3 depicts the data acquisition process graphically, in form of a dynamic Bayesian network [24]. As this graph suggests, each measurement $z_1, \ldots, z_t$ is a functions of the position of the corresponding feature, along with the robot pose at the time the measurement was taken. Knowledge of the robot path "d-separates" [58] the individual feature estimation problems and renders them independent of each other. Knowledge of the exact location of one feature will therefore tell us nothing about the locations of other features.

The same observation is easily derived mathematically. The stated independence is given by the following product form:

$$p(\Theta \mid s^t, n^t, z^t) \quad = \quad \prod_{n=1}^{N} p(\theta_n \mid s^t, n^t, z^t) \tag{12}$$

Notice that all probabilities are conditioned on the robot path s^t. Our derivation of (12) requires the distinction of two possible cases, depending on whether or not the feature θ_n was observed in the most recent measurement. In particular, if $n_t \neq n$, the most recent measurement z_t has no effect on the posterior, and neither has the robot pose s_t or the correspondence n_t. Thus, we obtain:

$$p(\theta_n \mid s^t, n^t, z^t) \quad = \quad p(\theta_n \mid s^{t-1}, n^{t-1}, z^{t-1}) \tag{13}$$

If $n_t = n$, that is, if $\theta_n = \theta_{n_t}$ was observed by the most recent measurement z_t, the situation calls for applying Bayes rule, followed by some standard simplifications:

$$\begin{aligned} p(\theta_{n_t} \mid s^t, n^t, z^t) \quad &= \quad \frac{p(z_t \mid \theta_{n_t}, s^t, n^t, z^{t-1})\; p(\theta_{n_t} \mid s^t, n^t, z^{t-1})}{p(z_t \mid s^t, n^t, z^{t-1})} \\ &= \quad \frac{p(z_t \mid s_t, \theta_{n_t}, n_t)\; p(\theta_{n_t} \mid s^{t-1}, n^{t-1}, z^{t-1})}{p(z_t \mid s^t, n^t, z^{t-1})} \end{aligned} \tag{14}$$

This gives us the following expression for the probability $p(\theta_{n_t} \mid s^{t-1}, n^{t-1}, z^{t-1})$:

$$p(\theta_{n_t} \mid s^{t-1}, n^{t-1}, z^{t-1}) \quad = \quad \frac{p(\theta_{n_t} \mid s^t, n^t, z^t)\; p(z_t \mid s^t, n^t, z^{t-1})}{p(z_t \mid s_t, \theta_{n_t}, n_t)} \tag{15}$$

The proof of the correctness of (12) is now carried out by mathematical induction. Let us assume that the posterior at time $t-1$ is already factored:

$$p(\Theta \mid s^{t-1}, n^{t-1}, z^{t-1}) \quad = \quad \prod_{n=1}^{N} p(\theta_n \mid s^{t-1}, n^{t-1}, z^{t-1}) \tag{16}$$

This statement is trivially true at $t = 1$, since in the beginning the robot has no knowledge about any feature, and hence all estimates are independent. At time t, the posterior is of the following form:

$$\begin{aligned} p(\Theta \mid s^t, n^t, z^t) &= \frac{p(z_t \mid \Theta, s^t, n^t, z^{t-1})\; p(\Theta \mid s^t, n^t, z^{t-1})}{p(z_t \mid s^t, n^t, z^{t-1})} \\ &= \frac{p(z_t \mid s_t, \theta_{n_t}, n_t)\; p(\Theta \mid s^{t-1}, n^{t-1}, z^{t-1})}{p(z_t \mid s^t, n^t, z^{t-1})} \end{aligned} \tag{17}$$

Plugging in our inductive hypothesis (16) gives us:

$$\begin{aligned} & p(\Theta \mid s^t, n^t, z^t) \\ &= \frac{p(z_t \mid s_t, \theta_{n_t}, n_t)}{p(z_t \mid s^t, n^t, z^{t-1})} \prod_{n=1}^{N} p(\theta_n \mid s^{t-1}, n^{t-1}, z^{t-1}) \\ &= \frac{p(z_t \mid s_t, \theta_{n_t}, n_t)}{p(z_t \mid s^t, n^t, z^{t-1})} \underbrace{p(\theta_{n_t} \mid s^{t-1}, n^{t-1}, z^{t-1})}_{Eq.\ (15)} \prod_{n \neq n_t} \underbrace{p(\theta_n \mid s^{t-1}, n^{t-1}, z^{t-1})}_{Eq.\ (13)} \\ &= p(\theta_{n_t} \mid s^t, n^t, z^t) \prod_{n \neq n_t} p(\theta_n \mid s^t, n^t, z^t) \\ &= \prod_{n=1}^{N} p(\theta_n \mid s^t, n^t, z^t) \end{aligned} \tag{18}$$

Notice that we have substituted our Equations (13) and (15) as indicated. This shows the correctness of Equation (12). The correctness of the main form (11) follows now directly from this result and the following generic transformation:

$$\begin{aligned} p(s^t, \Theta \mid n^t, z^t, u^t) &= p(s^t \mid n^t, z^t, u^t)\; p(\Theta \mid s^t, n^t, z^t, u^t) \\ &= p(s^t \mid n^t, z^t, u^t)\; p(\Theta \mid s^t, n^t, z^t) \\ &= p(s^t \mid n^t, z^t, u^t) \prod_{n=1}^{N} p(\theta_n \mid s^t, n^t, z^t) \end{aligned} \tag{19}$$

We note that conditioning on the entire path s^t is indeed essential for this result. The most recent pose s_t would be insufficient as conditioning variable, as dependencies may arise through previous poses This observation provides the motivation for our choice of posterior over paths and maps (5), in place of the much more common form stated in Equation (3).

4 FastSLAM with Known Data Association

Historically, FastSLAM 1.0 was the earliest version of the FastSLAM family of algorithms, and it is also the easiest to implement [45]. We will therefore begin our description of FastSLAM with version 1.0, although most of the observations in this sections apply equally to FastSLAM 2.0. Both FastSLAM algorithms exploit the factored posterior derived in the previous section. The factorial nature of the posterior provides us with significant computational advantages over SLAM algorithms that estimate an unstructured posterior distribution. FastSLAM exploits the factored representation by maintaining $N+1$ filters, one for each factor in (11). By doing so, all $N+1$ filters are low dimensional.

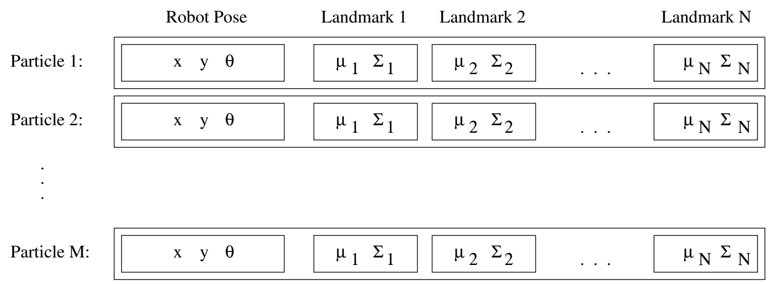

Figure 4: Particles in FastSLAM.

More specifically, both FastSLAM versions calculates the posterior over robot paths $p(s^t \mid n^t, z^t, u^t)$ by a particle filter [18, 37], similar to previous work in mobile robot localization [23], mapping [65], and visual tracking [28]. The particle filter has the pleasing property that the amount of computation needed for each incremental update stays constant, regardless of the path length t. Additionally, it can cope gracefully with non-linear robot motion models. The remaining N (conditional) posteriors over feature locations $p(\theta_n \mid s^t, n^t, z^t, u^t)$ are calculated by extended Kalman filters (EKFs). Each EKF estimates a single landmark pose, hence it is low-dimensional. The individual EKFs are conditioned on robot paths. Hence, each particle possesses its own set of EKFs. In total there are NM EKFs, one for each feature in the map and one for each landmark.[1]

Figure 4 illustrates the structure of the M particles in FastSLAM. Put into equations, each particle is of the form

$$S_t^{[m]} \quad = \quad \left\langle s^{t,[m]}, \mu_{1,t}^{[m]}, \Sigma_{1,t}^{[m]}, \ldots, \mu_{N,t}^{[m]}, \Sigma_{N,t}^{[m]} \right\rangle \tag{20}$$

The bracketed notation $[m]$ indicates the index of the particle; $s^{t,[m]}$ is its path estimate, and $\mu_{n,t}^{[m]}$ and $\Sigma_{n,t}^{[m]}$ are the mean and variance of the Gaussian representing the n-th feature location. Together, all these quantities form the m-th particle $S_t^{[m]}$, of which there are a total of M in the FastSLAM posterior.

Filtering, that is, calculating the posterior at time t from the one at time $t-1$ involves generating a new particle set S_t from S_{t-1}, the particle set one time step earlier. This new particle set incorporates a new control u_t and a measurement z_t (with associated correspondence n_t). This update is performed in the following steps:

1. **Extending the path posterior by sampling new poses.** FastSLAM 1.0 uses the control u_t to sample new robot pose s_t for each particle in S_{t-1}. More specifically, consider the m-the particle in S_{t-1}, denoted by $S_t^{[m]}$. FastSLAM 1.0 samples the pose s_t in accordance with the m-th particle, by drawing a sample according to the motion posterior

$$s_t^{[m]} \quad \sim \quad p(s_t \mid s_{t-1}^{[m]}, u_t) \tag{21}$$

Here $s_{t-1}^{[m]}$ is the posterior estimate for the robot location at time $t-1$, residing in the m-th particle. The resulting sample $s_t^{[m]}$ is then added to a temporary set of particles, along with the path of previous poses, $s^{t-1,[m]}$. This operation requires constant time per particle, independent of the map size N. The sampling step is graphically depicted in Figure 5, which illustrates a set of pose particles drawn from a single initial pose.

[1]Readers familiar with the statistical literature may want to note that both FastSLAM versions are instances of so-called Rao-Blackwellized particle filters [19, 52], by virtue of the fact that it combines particle representations with closed-form representations of certain marginals.

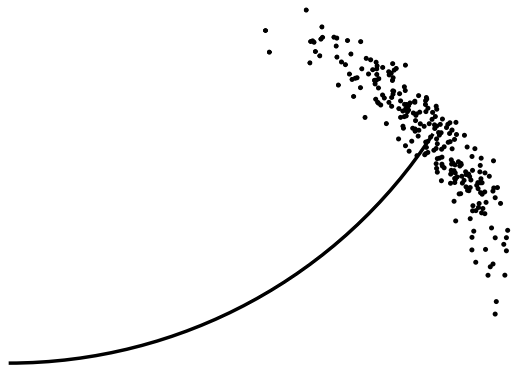

Figure 5: Samples drawn from the probabilistic motion model.

2. **Updating the observed landmark estimate.** Next, FastSLAM 1.0 updates the posterior over the landmark estimates, represented by the mean $\mu_{n,t-1}^{[m]}$ and the covariance $\Sigma_{n,t-1}^{[m]}$. The updated values are then added to the temporary particle set, along with the new pose.

   The update depends on whether or not a landmark n was observed at time t. For $n \neq n_t$, we already established in Equation (13) that the posterior over the landmark remains unchanged. This implies the simple update:

$$\left\langle \mu_{n,t}^{[m]}, \Sigma_{n,t}^{[m]} \right\rangle \quad = \quad \left\langle \mu_{n,t-1}^{[m]}, \Sigma_{n,t-1}^{[m]} \right\rangle \tag{22}$$

   For the observed feature $n = n_t$, the update is specified through Equation (14), restated here with the normalizer denoted by η:

$$p(\theta_{n_t} \mid s^t, n^t, z^t) \quad = \quad \eta\, p(z_t \mid s_t, \theta_{n_t}, n_t)\; p(\theta_{n_t} \mid s^{t-1}, n^{t-1}, z^{t-1}) \tag{23}$$

   The probability $p(\theta_{n_t} \mid s^{t-1}, n^{t-1}, z^{t-1})$ at time $t-1$ is represented by a Gaussian with mean $\mu_{n,t-1}^{[m]}$ and covariance $\Sigma_{n,t-1}^{[m]}$. For the new estimate at time t to also be Gaussian, FastSLAM linearizes the perceptual model $p(z_t \mid s_t, \theta_{n_t}, n_t)$ in the same way as EKFs [40]. In particular, FastSLAM approximates the measurement function g by the following first-degree Taylor expansion:

$$\begin{aligned} g(\theta_{n_t}, s_t^{[m]}) &\approx \underbrace{g(\mu_{n_t,t-1}^{[m]}, s_t^{[m]})}_{=:\, \hat{z}_t^{[m]}} + \underbrace{g'(s_t^{[m]}, \mu_{n_t,t-1}^{[m]})}_{=:\, G_t^{[m]}}(\theta_{n_t} - \mu_{n_t,t-1}^{[m]}) \\ &= \hat{z}_t^{[m]} + G_t^{[m]}\,(\theta_{n_t} - \mu_{n_t,t-1}^{[m]}) \end{aligned} \tag{24}$$

   Here the derivative g' is taken with respect to the feature coordinates θ_{n_t}. This linear approximation is tangent to g at $s_t^{[m]}$ and $\mu_{n_t,t-1}^{[m]}$. Under this approximation, the posterior for the location of feature n_t is indeed Gaussian. The new mean and covariance are obtained using the standard EKF measurement update [40]:

$$\begin{aligned} K_t^{[m]} &= \Sigma_{n_t,t-1}^{[m]} G_t^{[m]} (G_t^{[m]T} \Sigma_{n_t,t-1}^{[m]} G_t^{[m]} + R_t)^{-1} \\ \mu_{n_t,t}^{[m]} &= \mu_{n_t,t-1}^{[m]} + K_t^{[m]} (z_t - \hat{z}_t^{[m]})^T \\ \Sigma_{n_t,t}^{[m]} &= (I - K_t^{[m]} G_t^{[m]T}) \Sigma_{n_t,t-1}^{[m]} \end{aligned} \tag{25}$$

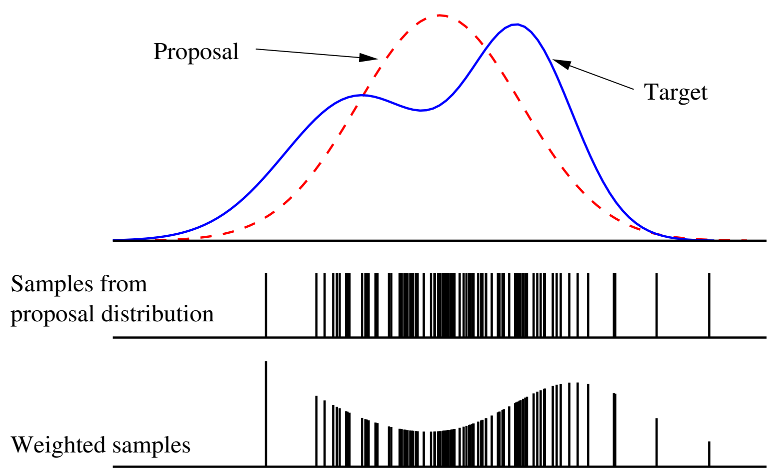

Figure 6: Samples cannot be drawn conveniently from the target target distribution (shown as a solid line). Instead, the importance sampler draws samples from the proposal distribution (dashed line), which has a simpler form. Below, samples drawn from the proposal distribution are drawn with lengths proportional to their importance weights..

Steps 1 and 2 are repeated M times, resulting in a temporary set of M particles.

3. **Resampling.** In a final step, FastSLAM resamples this set of particles, that is, FastSLAM draws from this temporary set M particles (with replacement), which then form the new particle set, S_t. The necessity to resample arises from the fact that the particles in the temporary set are not distributed according to the desired posterior: Step 1 generates poses s_t only in accordance with the most recent control u_t, paying no attention to the measurement z_t. Resampling is a common technique in particle filtering to correct for such mismatches.

   This situation is illustrated—for a simplified 1-D example—in Figure 6. Here the dashed line symbolizes the *proposal distribution*, which is the distribution at which particles are generated, and the solid line is the target distribution [41]. In FastSLAM, the proposal distribution does not depend on z_t, but the target distribution does. By weighing particles as shown in the bottom of this figure, and resampling according to those weights, the resulting particle set indeed approximates the target distribution. The weight of each sample used in the resampling step is called the *importance factor* [61].

   To determine importance factor of each particle, it will prove useful to calculate the actual proposal distribution of the path particles in the temporary set. Under the assumption that the set of path particles in S_{t-1} is distributed according to $p(s^{t-1} \mid z^{t-1}, u^{t-1}, n^{t-1})$ (which is an asymptotically correct approximation), path particles in the temporary set are distributed according to:

$$p(s^{t,[m]} \mid z^{t-1}, u^t, n^{t-1}) \quad = \quad p(s_t^{[m]} \mid s_{t-1}^{[m]}, u_t)\; p(s^{t-1,[m]} \mid z^{t-1}, u^{t-1}, n^{t-1}) \tag{26}$$

   The factor $p(s_t^{[m]} \mid s_{t-1}^{[m]}, u_t)$ is the sampling distribution used in Equation (21).

   The *target distribution* takes into account the measurement at time z_t, along with the correspondence n_t:

$$p(s^{t,[m]} \mid z^t, u^t, n^t) \tag{27}$$

   The resampling process accounts for the difference of the target and the proposal distribution. The importance factor for resampling is given by the quotient of the target and the proposal distribution [41]:

$$
\begin{aligned}
w_t^{[m]} &= \frac{\text{target distribution}}{\text{proposal distribution}} \\
&= \frac{p(s^{t,[m]} \mid z^t, u^t, n^t)}{p(s^{t,[m]} \mid z^{t-1}, u^t, n^{t-1})} \\
&= \eta\, p(z_t \mid s^{t,[m]}, z^{t-1}, n^t)
\end{aligned} \tag{28}
$$

The last transformation is a direct consequence of the following transformation of the enumerator in (28):

$$
\begin{aligned}
p(s^{t,[m]} \mid z^t, u^t, n^t) &= \eta\, p(z_t \mid s^{t,[m]}, z^{t-1}, u^t, n^t)\; p(s^{t,[m]} \mid z^{t-1}, u^t, n^t) \\
&= \eta\, p(z_t \mid s^{t,[m]}, z^{t-1}, n^t)\; p(s^{t,[m]} \mid z^{t-1}, u^t, n^{t-1})
\end{aligned} \tag{29}
$$

To calculate the probability $p(z_t \mid s^{t,[m]}, z^{t-1}, n^t)$ in (28), it will be necessary to transform it further. In particular, it is equivalent to the following integration, where we once again omit variables irrelevant to the prediction of sensor measurements:

$$
\begin{aligned}
w_t^{[m]} &= \eta \int p(z_t \mid \theta_{n_t}, s^{t,[m]}, z^{t-1}, n^t)\; p(\theta_{n_t} \mid s^{t,[m]}, z^{t-1}, n^t)\; d\theta_{n_t} \\
&= \eta \int p(z_t \mid \theta_{n_t}, n_t, s_t^{[m]})\; \underbrace{p(\theta_{n_t} \mid s^{t-1,[m]}, z^{t-1}, n^{t-1})}_{\sim \mathcal{N}(\mu^{[m]}_{n_t,t-1}, \Sigma^{[m]}_{n_t,t-1})}\; d\theta_{n_t}
\end{aligned} \tag{30}
$$

Here $\mathcal{N}(x; \mu, \Sigma)$ denotes a Gaussian distribution over the variable x with mean μ and covariance Σ. The integration in (30) involves the estimate of the observed landmark location at time t, and the measurement model. To calculate (30) in closed form, FastSLAM employs the very same linear approximation used in the measurement update in Step 2. In particular, the importance factor is given by

$$
w_t^{[m]} \approx \eta\, |2\pi Q_t^{[m]}|^{-\frac{1}{2}} \exp\left\{-\tfrac{1}{2}(z_t - \hat{z}_t^{[m]})^T Q_t^{[m]-1}(z_t - \hat{z}_t^{[m]})\right\} \tag{31}
$$

with the covariance

$$
Q_t^{[m]} = G_t^{[m]T} \Sigma^{[m]}_{n,t-1} G_t^{[m]} + R_t \tag{32}
$$

This expression is the probability of the actual measurement z_t under the Gaussian that results from the convolution of the distributions in (30), exploiting our linear approximation of g. The resulting importance weights are used to draw (with replacement) M new samples from the temporary sample set. Through this resampling process, particles survive in proportion of their measurement probability. Unfortunately, resampling may take time linear in the number of features N, since entire maps may have to be duplicated when a particle is drawn more than once.

These three steps together constitute the update rule of the FastSLAM 1.0 algorithm for SLAM problems with known data association. We note that the execution time of the update does not depend on the total path length t. In fact, only the most recent pose $s_{t-1}^{[m]}$ is used in the process of generating a new particle at time t. Consequently, past poses can safely be discarded. This has the pleasing consequence that neither the time requirements, not the memory requirements of FastSLAM depend on t.

(a) (b)

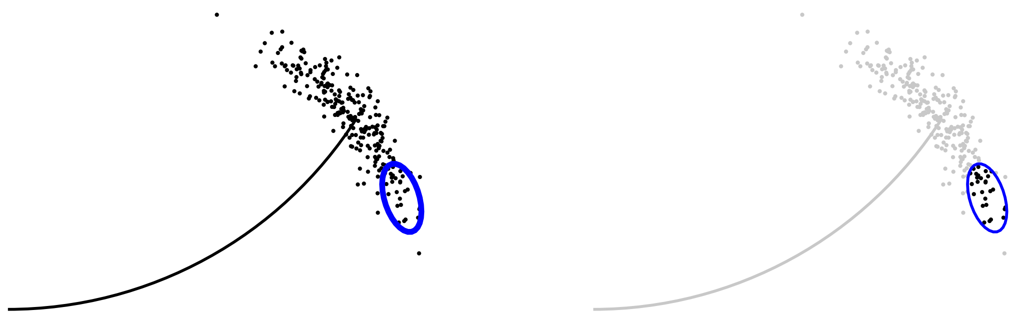

Figure 7: Mismatch between proposal and posterior distributions: (a) illustrates the forward samples generated by FastSLAM 1.0, and the posterior induced by the measurement (ellipse). Diagram (b) shows the sample set after the resampling step.

5 FastSLAM 2.0: Improved Proposal Distribution

FastSLAM 2.0 [46] is largely equivalent to FastSLAM 1.0, with one important exception: Its proposal distribution takes the measurement z^t into consideration. By doing to it can avoid some important problems that can arise in FastSLAM 1.0. In particular, FastSLAM 1.0 samples poses bases on the control u_t only, and then uses the measurement z_t to resample those poses. This is problematic when the accuracy of control is low relative to the accuracy of the robot's sensors. Such a situation is illustrated in Figure 7: Here the proposal generates a large spectrum of samples shown in Figure 7a, but only a small subset of these samples have high likelihood, as indicated by the ellipsoid. After resampling, only particles within the ellipsoid "survive" with reasonably high likelihood. Clearly, it would be advantageous to take the measurement into consideration when generating particles—which FastSLAM 1.0 fails to do. FastSLAM 2.0 achieves this by sampling poses based on the measurement z_t, in addition to the control u_t. Thus, as a result, FastSLAM 2.0 is less wasteful with its particle than FastSLAM 1.0. We will quantify the effect of this change in detain in the experimental results section of this paper. Unfortunately FastSLAM 2.0 is more difficult to implement than FastSLAM 1.0, and its mathematical derivation is more involved. In the remainder of this section we will discuss the individual update steps in FastSLAM 2.0, which parallel the corresponding steps in FastSLAM 1.0 as described in the previous section.

5.1 Extending The Path Posterior By Sampling A New Pose

In FastSLAM 2.0, the pose $s_t^{[m]}$ is drawn from the posterior

$$s_t^{[m]} \quad \sim \quad p(s_t \mid s^{t-1,[m]}, u^t, z^t, n^t) \tag{33}$$

which differs from the proposal distribution provided in (21) in that (33) takes the measurement z_t into consideration, along with the correspondence n_t. The reader may recall that $s^{t-1,[m]}$ is the path up to time $t-1$ of the m-th particle.

The mechanism for sampling from (33) requires further analysis. First, we rewrite (33) in terms of the 'known' distributions, such as the measurement and motion models, and the Gaussian feature estimates in the m-th particle.

$$\begin{aligned}
& p(s_t \mid s^{t-1,[m]}, u^t, z^t, n^t) \\
& \quad \overset{\text{Bayes}}{=} \quad \frac{p(z_t \mid s_t, s^{t-1,[m]}, u^t, z^{t-1}, n^t)\; p(s_t \mid s^{t-1,[m]}, u^t, z^{t-1}, n^t)}{p(z_t \mid s^{t-1,[m]}, u^t, z^{t-1}, n^t)} \\
& \quad = \quad \eta^{[m]}\; p(z_t \mid s_t, s^{t-1,[m]}, u^t, z^{t-1}, n^t)\; p(s_t \mid s^{t-1,[m]}, u^t, z^{t-1}, n^t)
\end{aligned}$$

$$
\begin{aligned}
&\overset{\text{Markov}}{=} \quad \eta^{[m]}\, p(z_t \mid s_t, s^{t-1,[m]}, u^t, z^{t-1}, n^t)\; p(s_t \mid s_{t-1}^{[m]}, u_t) \\
&= \quad \eta^{[m]} \int p(z_t \mid \theta_{n_t}, s_t, s^{t-1,[m]}, u^t, z^{t-1}, n^t)\; p(\theta_{n_t} \mid s_t, s^{t-1,[m]}, u^t, z^{t-1}, n^t)\; d\theta_{n_t} \\
&\qquad p(s_t \mid s_{t-1}^{[m]}, u_t) \\
&\overset{\text{Markov}}{=} \quad \eta^{[m]} \int \underbrace{p(z_t \mid \theta_{n_t}, s_t, n_t)}_{\sim \mathcal{N}(z_t; g(\theta_{n_t}, s_t), R_t)}\; \underbrace{p(\theta_{n_t} \mid s^{t-1,[m]}, z^{t-1}, n^{t-1})}_{\sim \mathcal{N}(\theta_{n_t}; \mu_{n_t,t-1}^{[m]}, \Sigma_{n_t,t-1}^{[m]})}\; d\theta_{n_t} \quad \underbrace{p(s_t \mid s_{t-1}^{[m]}, u_t)}_{\sim \mathcal{N}(s_t; h(s_{t-1}^{[m]}, u_t), P_t)}
\end{aligned}
\tag{34}
$$

This expression makes apparent that our sampling distribution is truly the convolution of two Gaussians multiplied by a third. Unfortunately, in the general case the sampling distribution possesses no closed form from which we could easily sample. The culprit is the function g: If it were linear, this probability would be Gaussian, a fact that shall become more obvious below. In the general case, not even the integral in (34) possess a closed form solution. For this reason, sampling from the probability (34) is difficult.

This observation motivates the replacement of g by a linear approximation. As in FastSLAM 1.0, this approximation is obtained through a first order Taylor expansion, given by the following linear function:

$$
g(\theta_{n_t}, s_t) \quad \approx \quad \hat{z}_t^{[m]} + G_\theta(\theta_{n_t} - \mu_{n_t,t-1}^{[m]}) + G_s(s_t - \hat{s}_t^{[m]}) \tag{35}
$$

Here we use the following abbreviations:

$$
\hat{z}_t^{[m]} \quad = \quad g(\mu_{n_t,t-1}^{[m]}, \hat{s}_t^{[m]}) \tag{36}
$$

$$
\hat{s}_t^{[m]} \quad = \quad h(s_{t-1}^{[m]}, u_t) \tag{37}
$$

The matrices G_θ and G_s are the Jacobians of g, that is, they are the derivatives of g with respect to θ_{n_t} and s_t, respectively, evaluated at the expected values of their arguments:

$$
G_\theta \quad = \quad \nabla_{\theta_{n_t}} g(\theta_{n_t}, s_t)\Big|_{s_t = \hat{s}_t^{[m]}; \theta_{n_t} = \mu_{n_t,t-1}^{[m]}} \tag{38}
$$

$$
G_s \quad = \quad \nabla_{s_t} g(\theta_{n_t}, s_t)\big|_{s_t = \hat{s}_t^{[m]}; \theta_{n_t} = \mu_{n_t,t-1}^{[m]}} \tag{39}
$$

Under this approximation, the desired sampling distribution (34) is a Gaussian with the following parameters:

$$
\Sigma_{s_t}^{[m]} \quad = \quad \left[G_s^T Q_t^{[m]-1} G_s + P_t^{-1} \right]^{-1} \tag{40}
$$

$$
\mu_{s_t}^{[m]} \quad = \quad \Sigma_{s_t}^{[m]} G_s^T Q_t^{[m]-1} (z_t - \hat{z}_t^{[m]}) + \hat{s}_t^{[m]} \tag{41}
$$

where the matrix $Q_t^{[m]}$ is defined as follows:

$$
Q_t^{[m]} \;=\; R_t + G_\theta \Sigma_{n_t,t-1}^{[m]} G_\theta^T \tag{42}
$$

To see, we note that under out linear approximation the convolution theorem provides us with a closed form for the integral term in (34):

$$
\mathcal{N}(z_t; \hat{z}_t^{[m]} + G_s s_t - G_s \hat{s}_t^{[m]}, Q_t^{[m]}) \tag{43}
$$

The sampling distribution (34) is now given by the product of this normal distribution and the rightmost term in (34), the normal $\mathcal{N}(s_t; \hat{s}_t^{[m]}, P_t)$. Written in Gaussian form, we have

$$
p(s_t \mid s^{t-1,[m]}, u^t, z^t, n^t) \quad = \quad \eta\; \exp\left\{ -y_t^{[m]} \right\} \tag{44}
$$

with

$$\begin{aligned} y_t^{[m]} &= \tfrac{1}{2}\Big[(z_t - \hat{z}_t^{[m]} - G_s s_t + G_s \hat{s}_t^{[m]})^T Q_t^{[m]-1} (z_t - \hat{z}_t^{[m]} - G_s s_t + G_s \hat{s}_t^{[m]}) \\ &\quad + (s_t - \hat{s}_t^{[m]})^T P_t^{-1} (s_t - \hat{s}_t^{[m]})\Big] \end{aligned} \tag{45}$$

This expression is obviously quadratic in our target variable s_t, hence $p(s_t \mid s^{t-1,[m]}, u^t, z^t, n^t)$ is Gaussian. The mean and covariance of this Gaussian are equivalent to the minimum of $y_t^{[m]}$ and its curvature. Those are identified by calculating the first and second derivatives of $y_t^{[m]}$ with respect to s_t:

$$\begin{aligned} \frac{\partial y_t^{[m]}}{\partial s_t} &= -G_s^T Q_t^{[m]-1}(z_t - \hat{z}_t^{[m]} - G_s s_t + G_s \hat{s}_t^{[m]}) + P_t^{-1}(s_t - \hat{s}_t^{[m]}) \\ &= (G_s^T Q_t^{[m]-1} G_s + P_t^{-1}) s_t - G_s^T Q_t^{[m]-1}(z_t - \hat{z}_t^{[m]} + G_s \hat{s}_t^{[m]}) - P_t^{-1} \hat{s}_t^{[m]} \end{aligned} \tag{46}$$

$$\frac{\partial^2 y_t^{[m]}}{\partial s_t^2} = G_s^T Q_t^{[m]-1} G_s + P_t^{-1} \tag{47}$$

The covariance $\Sigma_{s_t}^{[m]}$ of the sampling distribution is now obtained by the inverse of the second derivative

$$\Sigma_{s_t}^{[m]} = \left[G_s^T Q_t^{[m]-1} G_s + P_t^{-1}\right]^{-1} \tag{48}$$

The mean $\mu_{s_t}^{[m]}$ of the sample distribution is obtained by setting the first derivative (46) to zero, which gives us:

$$\begin{aligned} \mu_{s_t}^{[m]} &= \Sigma_{s_t}^{[m]}\left[G_s^T Q_t^{[m]-1}(z_t - \hat{z}_t^{[m]} + G_s \hat{s}_t^{[m]}) + P_t^{-1}\hat{s}_t^{[m]}\right] \\ &= \Sigma_{s_t}^{[m]} G_s^T Q_t^{[m]-1}(z_t - \hat{z}_t^{[m]}) + \Sigma_{s_t}^{[m]}\left[G_s^T Q_t^{[m]-1} G_s + P_t^{-1}\right]\hat{s}_t^{[m]} \\ &= \Sigma_{s_t}^{[m]} G_s^T Q_t^{[m]-1}(z_t - \hat{z}_t^{[m]}) + \hat{s}_t^{[m]} \end{aligned} \tag{49}$$

This Gaussian is the approximation of the desired sampling distribution (33) in FastSLAM 2.0. Obviously, this proposal distribution is quite a bit more involved than the much simpler one for FastSLAM 1.0 in Equation (21). Its advantage will be characterized below, when we will empirically compare both FastSLAM algorithms.

5.2 Updating The Observed Landmark Estimate

Just like FastSLAM 1.0, FastSLAM 2.0 updates the posterior over the landmark estimates based on the measurement z_t and the sampled pose $s_t^{[m]}$. The estimates at time $t-1$ are once again represented by the mean $\mu_{n,t-1}^{[m]}$ and the covariance $\Sigma_{n,t-1}^{[m]}$, and the updated estimates are the mean $\mu_{n,t}^{[m]}$ and the covariance $\Sigma_{n,t}^{[m]}$. The nature of the update depends on whether or not a landmark n was observed at time t. For $n \neq n_t$, we already established in Equation (13) that the posterior over the landmark remains unchanged. This implies that instead of updating the estimated, we merely have to copy it.

For the observed feature $n = n_t$, the situation is more intricate. Equation (14) already specified the posterior over observed features, here written with the particle index m:

$$p(\theta_{n_t} \mid s^{t,[m]}, n^t, z^t) = \eta \underbrace{p(z_t \mid \theta_{n_t}, s_t^{[m]}, n_t)}_{\sim \mathcal{N}(z_t; g(\theta_{n_t}, s_t^{[m]}), R_t)} \underbrace{p(\theta_{n_t} \mid s^{t-1,[m]}, z^{t-1}, n^{t-1})}_{\sim \mathcal{N}(\theta_{n_t}; \mu_{n_t,t-1}^{[m]}, \Sigma_{n_t,t-1}^{[m]})} \tag{50}$$

As in (34), the nonlinearity of g causes this posterior to be non-Gaussian, which is at odds with FastSLAM 2.0's Gaussian representation for feature estimates. Luckily, the exact same linearization as above provides the solution:

$$g(\theta_{n_t}, s_t) \approx \hat{z}_t^{[m]} + G_\theta(\theta_{n_t} - \mu_{n_t,t-1}^{[m]}) \tag{51}$$

(Notice that s_t is not a free variable here, hence we can omit the third term in (35).) This approximation renders the probability (50) Gaussian in the target variable θ_{n_t}:

$$\begin{aligned} & p(\theta_{n_t} \mid s^{t,[m]}, n^t, z^t) \\ = \;& \eta \, \exp\Big\{ -\tfrac{1}{2}(z_t - \hat{z}_t^{[m]} - G_\theta(\theta_{n_t} - \mu_{n_t,t-1}^{[m]}))^T R_t^{-1} (z_t - \hat{z}_t^{[m]} - G_\theta(\theta_{n_t} - \mu_{n_t,t-1}^{[m]})) \\ & \qquad -\tfrac{1}{2}(\theta_{n_t} - \mu_{n_t,t-1}^{[m]})^T \Sigma_{n_t,t-1}^{[m]-1} (\theta_{n_t} - \mu_{n_t,t-1}^{[m]}) \Big\} \end{aligned} \tag{52}$$

The new mean and covariance are obtained using the standard EKF measurement update equations [29, 40], whose derivation can be found in in Appendix A.

$$K_t^{[m]} = \Sigma_{n_t,t-1}^{[m]} G_\theta^T Q_t^{[m]-1} \tag{53}$$

$$\mu_{n_t,t}^{[m]} = \mu_{n_t,t-1}^{[m]} + K_t^{[m]}(z_t - \hat{z}_t^{[m]}) \tag{54}$$

$$\Sigma_{n_t,t}^{[m]} = (I - K_t^{[m]} G_\theta) \Sigma_{n_t,t-1}^{[m]} \tag{55}$$

5.3 Calculating Importance Factors

The particles generated thus far do not yet match the desired posterior. In FastSLAM 2.0, the culprit is the normalizer $\eta^{[m]}$ in (34), which may be different for different particles m. These differences are not yet accounted for in the re sampling process. As in FastSLAM 1.0, the importance factor is given by the following quotient.

$$w_t^{[m]} = \frac{\text{target distribution}}{\text{proposal distribution}} \tag{56}$$

Once again, the target distribution that we would like our particles to assume is given by the path posterior, $p(s^{t,[m]} \mid z^t, u^t, n^t)$. Under the (asymptotically correct) assumptions that paths in $s^{t-1,[m]}$ were generated according to the target distribution one time step earlier, $p(s^{t-1,[m]} \mid z^{t-1}, u^{t-1}, n^{t-1})$, we note that the proposal distribution is now given by the product

$$p(s^{t-1,[m]} \mid z^{t-1}, u^{t-1}, n^{t-1}) \; p(s_t^{[m]} \mid s^{t-1,[m]}, u^t, z^t, n^t) \tag{57}$$

The second term in this product is the pose sampling distribution (34). The importance weight is obtained as follows:

$$\begin{aligned} w_t^{[m]} &= \frac{p(s^{t,[m]} \mid u^t, z^t, n^t)}{p(s_t^{[m]} \mid s^{t-1,[m]}, u^t, z^t, n^t) \; p(s^{t-1,[m]} \mid u^{t-1}, z^{t-1}, n^{t-1})} \\ &= \frac{p(s_t^{[m]} \mid s^{t-1,[m]}, u^t, z^t, n^t) \; p(s^{t-1,[m]} \mid u^t, z^t, n^t)}{p(s_t^{[m]} \mid s^{t-1,[m]}, u^t, z^t, n^t) \; p(s^{t-1,[m]} \mid u^{t-1}, z^{t-1}, n^{t-1})} \\ &= \frac{p(s^{t-1,[m]} \mid u^t, z^t, n^t)}{p(s^{t-1,[m]} \mid u^{t-1}, z^{t-1}, n^{t-1})} \\ &\overset{\text{Bayes}}{=} \eta \, \frac{p(z_t \mid s^{t-1,[m]}, u^t, z^{t-1}, n^t) \; p(s^{t-1,[m]} \mid u^t, z^{t-1}, n^t)}{p(s^{t-1,[m]} \mid u^{t-1}, z^{t-1}, n^{t-1})} \\ &\overset{\text{Markov}}{=} \eta \, \frac{p(z_t \mid s^{t-1,[m]}, u^t, z^{t-1}, n^t) \; p(s^{t-1,[m]} \mid u^{t-1}, z^{t-1}, n^{t-1})}{p(s^{t-1,[m]} \mid u^{t-1}, z^{t-1}, n^{t-1})} \\ &= \eta \, p(z_t \mid s^{t-1,[m]}, u^t, z^{t-1}, n^t) \end{aligned} \tag{58}$$

The reader may notice that this expression is in essence the inverse of our normalization constant $\eta^{[m]}$ in (34). Further transformations give us the following form:

$$w_t^{[m]}$$

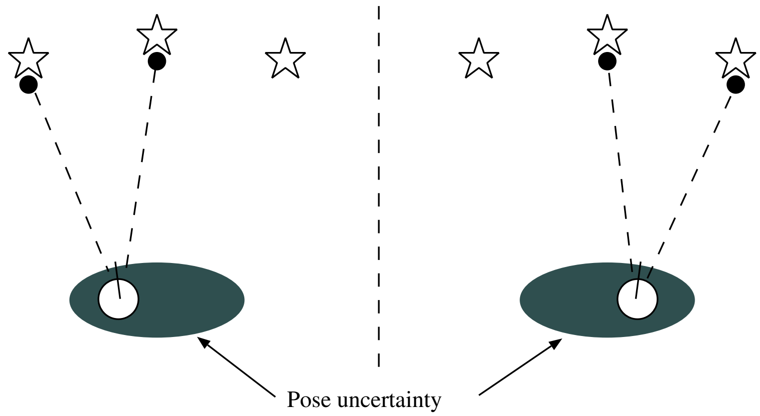

Figure 8: The data association problem in SLAM.

$$
\begin{aligned}
&= \eta \int p(z_t \mid s_t, s^{t-1,[m]}, u^t, z^{t-1}, n^t)\; p(s_t \mid s^{t-1,[m]}, u^t, z^{t-1}, n^t)\; ds_t \\
&\overset{\text{Markov}}{=} \eta \int p(z_t \mid s_t, s^{t-1,[m]}, u^t, z^{t-1}, n^t)\; p(s_t \mid s_{t-1}^{[m]}, u_t)\; ds_t \\
&= \eta \int\int p(z_t \mid \theta_{n_t}, s_t, s^{t-1,[m]}, u^t, z^{t-1}, n^t)\; p(\theta_{n_t} \mid s_t, s^{t-1,[m]}, u^t, z^{t-1}, n^t)\; d\theta_{n_t} \\
&\quad\; p(s_t \mid s_{t-1}^{[m]}, u_t)\; ds_t \\
&\overset{\text{Markov}}{=} \eta \int\int \underbrace{p(z_t \mid \theta_{n_t}, s_t, n_t)}_{\sim \mathcal{N}(z_t; g(\theta_{n_t}, s_t), R_t)}\; \underbrace{p(\theta_{n_t} \mid s^{t-1,[m]}, u^{t-1}, z^{t-1}, n^{t-1})}_{\sim \mathcal{N}(\theta_{n_t}; \mu_{n_t,t-1}^{[m]}, \Sigma_{n_t,t-1}^{[m]})}\; d\theta_{n_t}\; \underbrace{p(s_t \mid s_{t-1}^{[m]}, u_t)}_{\sim \mathcal{N}(s_t; \hat{s}_{t-1}^{[m]}, P_t)}\; ds_t
\end{aligned}
\tag{59}
$$

We find that this expression can again be approximated by a Gaussian over measurements z_t by linearizing g. As it is easily shown, the mean of the resulting Gaussian is $\hat{z}_t$, and its covariance is

$$
L_t^{[t]} \;=\; G_s P_t G_s^T + G_\theta \Sigma_{n_t,t-1}^{[m]} G_\theta^T + R_t \tag{60}
$$

Put differently, the (non-normalized) importance factor of the m-the particle is given by the following expression:

$$
w_t^{[m]} \;=\; |2\pi L_t^{[t]}|^{-\frac{1}{2}} \exp\left\{-\tfrac{1}{2}(z_t - \hat{z}_t)^T L_t^{[t]-1}(z_t - \hat{z}_t)\right\} \tag{61}
$$

As in FastSLAM 1.0, particles generated in Steps 1 and 2, along with their importance factor calculated in Step 3, are collected in a temporary particle set. The final step of the FastSLAM 2.0 update is a resampling step. Just like in FastSLAM 1.0, FastSLAM 2.0 draws (with replacement) M particles from the temporary particle set. Each particle is drawn with a probability proportional to its importance factor $w_t^{[m]}$. The resulting particle set represent (asymptotically) the desired factored posterior at time t.

6 Unknown Data Association

6.1 Data Association in SLAM

The biggest limitation of both FastSLAM algorithms, as described so far, has been the assumption of known data association. Real-world features are usually ambiguous. This section extends the FastSLAM algorithms to cases where the correspondence variables n^t are unknown.

Formally, the data association problem at time t is the problem of determining the variable n_t based on the available data. This problem is illustrated in Figure 8: Here a robot observes to landmarks. Depending on its actual pose relative to these landmarks, these measurements correspond to different landmarks in the map (depicted as stars in Figure 8). The 'classical' solution to the data association problem [4, 14, 17] is to chose n_t so that it maximizes the likelihood of the sensor measurement z_t:

$$\hat{n}_t = \underset{n_t}{\operatorname{argmax}}\, p(z_t \mid n_t, \hat{n}^{t-1}, s^t, z^{t-1}, u^t) \tag{62}$$

Such an estimator is called *maximum likelihood estimator* (ML). The term $p(z_t \mid n_t, \hat{n}^{t-1}, s^t, z^{t-1}, u^t)$ is usually referred to as *likelihood*. ML data association is often referred to as nearest neighbor method, interpreting the negative log likelihood as distance function. For Gaussians, the negative log likelihood is a Mahalanobis distance, and ML selects data associations by minimizing this Mahalanobis distance.

An alternative to the ML method is data association sampling (DAS):

$$\hat{n}_t \sim \eta\, p(z_t \mid n_t, \hat{n}^{t-1}, s^t, z^{t-1}, u^t) \tag{63}$$

DAS samples the data association variable according to the likelihood function, instead of deterministically selecting its most likely value. Both techniques, ML and DAS, make it possible to estimate the number of features in the map. SLAM techniques using ML create new features in the map if the likelihood falls below a threshold p_0 for all known features in the map. DAS associates an observed measurement with a new, previously unobserved feature stochastically. They do so with probability proportional to ηp_0, where η is a normalizer defined in (63).

In EKF-style approaches to the SLAM problem, ML is usually given preference over DAS, since the number of data association errors in ML is smaller. Because only a single data association decision is made for each measurement in most EKF-based implementations, these approaches tend to be brittle with regards to data association errors. A single data association error can induce significant errors in the map which in turn cause new data association errors, often with fatal consequences. Therefore, the corresponding SLAM algorithms tend to work well only when ambiguous features in the environment are spaced sufficiently far apart from each other to make confusions unlikely. For this reason, many implementations of SLAM extract sparse features from otherwise rich sensor measurements.

6.2 Data Association in FastSLAM

The key advantage of the FastSLAM over EKF-style approaches is its ability to pursue multiple data association hypotheses at the same time. This is due to the fact that the posterior is represented by multiple particles. In particular, FastSLAM estimates the correspondences on a per-particle basis, not on a per-filter basis as is the case for the EKF. This enables FastSLAM to use ML or even DAS for generating particle-specific data association decisions. As long as a small subset of the particles is based on the correct data association, data association errors are not as fatal as in EKF approaches. This is because particles subject to such errors tend to possess inconsistent maps, which increases the probability that they are simply sampled away in future resampling steps.

The mathematical definition of the per-particle data association is straightforward. Each particle maintains a local set of data association variables, denoted $\hat{n}_t^{[m]}$. In ML data association, each $\hat{n}_t^{[m]}$ is determined by maximizing the likelihood of the measurement z_t:

$$\hat{n}_t^{[m]} = \underset{n_t}{\operatorname{argmax}}\, p(z_t \mid n_t, \hat{n}^{t-1,[m]}, s^{t,[m]}, z^{t-1}, u^t) \tag{64}$$

DAS data association samples from the likelihood:

$$\hat{n}_t^{[m]} \sim \eta\, p(z_t \mid n_t, \hat{n}^{t-1,[m]}, s^{t,[m]}, z^{t-1}, u^t) \tag{65}$$

For both techniques cases, the likelihood is calculated as follows:

$$\begin{aligned}
& p(z_t \mid n_t, \hat{n}^{t-1,[m]}, s^{t,[m]}, z^{t-1}, u^t) \\
= & \int p(z_t \mid \theta_{n_t}, n_t, \hat{n}^{t-1,[m]}, s^{t,[m]}, z^{t-1}, u^t)\ p(\theta_{n_t} \mid n_t, \hat{n}^{t-1,[m]}, s^{t,[m]}, z^{t-1}, u^t)\ d\theta_{n_t} \\
= & \int \underbrace{p(z_t \mid \theta_{n_t}, n_t, s_t^{[m]})}_{\sim \mathcal{N}(z_t; g(\theta_{n_t}, s_t^{[m]}), R_t)} \ \underbrace{p(\theta_{n_t} \mid \hat{n}^{t-1,[m]}, s^{t-1,[m]}, z^{t-1})}_{\sim \mathcal{N}(\mu_{n_t,t-1}^{[m]}, \Sigma_{n_t,t-1}^{[m]})}\ d\theta_{n_t}
\end{aligned} \tag{66}$$

Linearization of g enables us to obtain this in closed form:

$$\begin{aligned}
& p(z_t \mid n_t, \hat{n}^{t-1,[m]}, s^{t,[m]}, z^{t-1}, u^t) \\
= & |2\pi Q_t^{[m]}|^{-\frac{1}{2}}\ \exp\left\{-\tfrac{1}{2}(z_t - g(\mu_{n_t,t-1}^{[m]}, s_t^{[m]}))^T Q_t^{[m]-1}(z_t - g(\mu_{n_t,t-1}^{[m]}, s_t^{[m]}))\right\}
\end{aligned} \tag{67}$$

The variable $Q_t^{[m]}$ was defined in Equation (42), as a function of the data association variable n_t. New features are added to the map in exactly the same way as outlined above. In the ML approach, a new feature is added when the probability $p(z_t \mid n_t, \hat{n}^{t-1,[m]}, s^{t,[m]}, z^{t-1}, u^t)$ falls beyond a threshold p_0. The DAS includes the hypothesis that an observation corresponds to a previously unobserved feature in its set of hypotheses, and samples it with probability ηp_0. To accommodate the particle-specific map size, each particle carries its own feature count. This count will be denoted $N_t^{[m]}$.

6.3 Feature Initialization

As in EKF-implementations of SLAM, initializing the newly added Kalman filter can be tricky, especially when individual measurements are insufficient to constrain the feature's location in all dimensions [15]. In many SLAM problems the measurement function g is invertible. This the case, for example, for robots measuring range and bearing to landmarks in the plane, in which a single measurement suffices to produce a (non-degenerate) estimate on the feature location. The initialization of the EKF is then straightforward:

$$s_t^{[m]} \sim p(s_t \mid s_{t-1}^{[m]}, u_t) \tag{68}$$
$$\mu_{n,t}^{[m]} = g^{-1}(z_t, s_t^{[m]}) \tag{69}$$
$$\Sigma_{n,t}^{[m]} = (G_{\hat{n}}^{[m]} R_t^{-1} G_{\hat{n}}^{[m]T})^{-1} \tag{70}$$
$$w_t^{[m]} = p_0 \tag{71}$$

Notice that for newly observed features, the pose $s_t^{[m]}$ is sampled according to the motion model $p(s_t \mid s_{t-1}^{[m]}, u_t)$. This distribution is equivalent to the FastSLAM sampling distribution (33) in situations where no previous location estimate for the observed feature is available.

Initialization techniques for situations in which g is not invertible are discussed in [15]. In general, such situations require the accumulation of multiple measurements, to obtain a good estimate for the linearization of g.

6.4 Feature Elimination and Negative Information

To accommodate features introduced erroneously into the map, FastSLAM features a mechanism for eliminating features that are not supported by sufficient evidence. In particular, our approach keeps track of the probabilities on the actual existence of individual features in the map, a technique commonly used in EKF-style algorithms [17]. Let $i_n^{[m]} \in \{0, 1\}$ be a binary variable that indicates the existence of the feature $\theta_n^{[m]}$. Our approach exploits the fact that each sensor measurement z_t carries evidence with regards to the physical existence of nearby features $\theta_n^{[m]}$: Observing the feature provides positive evidence for its existence, whereas not observing it when $\mu_n^{[m]}$ falls within the robot's perceptual range provides negative evidence. The resulting posterior probability

$$p(i_n^{[m]} \mid \hat{n}^{t,[m]}, s^{t,[m]}, z^{t-1}) \tag{72}$$

is estimated by a binary Bayes filter, familiar from the literature on occupancy grid maps [49]. FastSLAM represents the posterior in its log-odds form:

$$\tau_n^{[m]} \;=\; \ln \frac{p(i_n^{[m]} \mid \hat{n}^{t,[m]}, s^{t,[m]}, z^{t-1})}{1 - p(i_n^{[m]} \mid \hat{n}^{t,[m]}, s^{t,[m]}, z^{t-1})} \;=\; \sum_t \ln \frac{p(i_n^{[m]} \mid s_t^{[m]}, z_t, \hat{n}_t^{[m]})}{1 - p(i_n^{[m]} \mid s_t^{[m]}, z_t, \hat{n}_t^{[m]})} \tag{73}$$

The advantage of this rule lies in the fact that updates are additive (see [66] for a derivation). In the most simple implementation, observing of a feature leads to the addition of a positive value

$$\rho^+ \;=\; \ln \frac{p(i_{\hat{n}_t}^{[m]} \mid s_t^{[m]}, z_t, \hat{n}_t^{[m]})}{1 - p(i_{\hat{n}_t}^{[m]} \mid s_t^{[m]}, z_t, \hat{n}_t^{[m]})} \tag{74}$$

to the log-odds value, and not observing it leads to the addition of a negative value

$$\rho^- \;=\; \ln \frac{p(i_{n \neq \hat{n}_t}^{[m]} \mid s_t^{[m]}, z_t, \hat{n}_t^{[m]})}{1 - p(i_{n \neq \hat{n}_t}^{[m]} \mid s_t^{[m]}, z_t, \hat{n}_t^{[m]})} \tag{75}$$

To implement this approach in real-time, the variable t starts at the time a feature is first introduced in the map. Features are terminated when their log-odds of existence falls beyond a certain bound. This mechanism enables FastSLAM 's particles to free themselves of spurious features.

6.5 The FastSLAM Algorithms

Tables 1 and 2 summarize both FastSLAM algorithms. In both algorithms, particles are of the form

$$S_t^{[m]} \;=\; \left\langle s_t^{[m]}, N_t^{[m]}, \left\langle \mu_{1,t}^{[m]}, \Sigma_{1,t}^{[m]}, \tau_1^{[m]} \right\rangle, \ldots, \left\langle \mu_{N_t^{[m]},t}^{[m]}, \Sigma_{N_t^{[m]},t}^{[m]}, \tau_{N_t^{[m]}}^{[m]} \right\rangle \right\rangle \tag{76}$$

In addition to the pose $s_t^{[m]}$ and the feature estimates $\mu_{n,t}^{[m]}$ and $\Sigma_{n,t}^{[m]}$, each particle maintains the number of features $N_t^{[m]}$ in its local map, and each feature carries a probabilistic estimate of its existence $\tau_n^{[m]}$. Iterating the filter requires time linear in the maximum number of features $\max_m N_t^{[m]}$ in each map, and it is also linear in the number of particles M. Further below, we will discuss advanced data structure that yield more efficient implementations.

We note that both versions of FastSLAM, as described here, consider a single measurement at a time. As discussed above, this choice has been made for notational convenience. Most competitive SLAM implementations (including ours) consider multiple features in the data association step [3, 26, 53, 65]. Doing so tends to decreases the data association error rate, and the resulting maps become more accurate. This follows from a mutual exclusion property, which states that no landmark can be observed at two different locations at the same time [16]. Like many other implementations before, our implementation considers all features observed in a single sensor scan when calculating the measurement likelihood. The necessary modification of FastSLAM is straightforward but will not be further elaborated here. Below, we will provide an empirical comparison of both variants of this algorithm, highlighting the advantages of FastSLAM 2.0 over 1.0.

Algorithm FastSLAM 1.0(z_t, u_t, S_{t-1})**:**

```
for m = 1 to M do                                                    // loop over all particles
  retrieve ⟨ s_{t-1}^{[m]}, N_{t-1}^{[m]}, ⟨μ_{1,t-1}^{[m]}, Σ_{1,t-1}^{[m]}, i_1^{[m]}⟩, ..., ⟨μ_{N_{t-1}^{[m]},t-1}^{[m]}, Σ_{N_{t-1}^{[m]},t-1}^{[m]}, i_{N_{t-1}^{[m]}}^{[m]}⟩ ⟩ from S_{t-1}
  s_t^{[m]} ~ p(s_t | s_{t-1}^{[m]}, u_t)                            // sample new pose
  for n = 1 to N_{t-1}^{[m]} do                                      // calculate measurement likelihoods
    ẑ_n = g(μ_{n,t-1}^{[m]}, s_t^{[m]})                              // measurement prediction
    G_n = g'(s_t^{[m]}, μ_{n,t-1}^{[m]})                             // calculate Jacobian
    Q_n = G_n^T Σ_{n,t-1}^{[m]} G_n + R_t                            // measurement covariance
    w_n = |2πQ_n|^{-1/2} exp{ -1/2 (z_t - ẑ_n)^T Q_n^{-1} (z_t - ẑ_n) }   // likelihood of correspondence
  endfor
  w_{N_{t-1}^{[m]}+1} = p_0                                          // importance factor of new landmark
  n̂ = argmax_{n=1,...,N_{t-1}^{[m]}+1} w_n                           // max likelihood correspondence
  N_t^{[m]} = max{N_{t-1}^{[m]}, n̂}                                  // new number of features in map
  for n = 0 to N_t^{[m]} do                                          // update Kalman filters
    if n = N_{t-1}^{[m]} + 1 then                                    // is new feature?
      μ_{n,t}^{[m]} = g^{-1}(z_t, s_t^{[m]})                         // initialize mean
      Σ_{n,t}^{[m]} = G_n̂^{-1} R_t (G_n̂^{-1})^T                     // initialize covariance
      i_{n,t}^{[m]} = 1                                              // initialize counter
    else if n = n̂ then                                               // is observed feature?
      K = Σ_{n,t-1}^{[m]} G_n̂ Q_n̂^{-1}                                // calculate Kalman gain
      μ_{n,t}^{[m]} = μ_{n,t-1}^{[m]} + K(z_t - ẑ_n̂)^T                // update mean
      Σ_{n,t}^{[m]} = (I - K G_n̂^T) Σ_{n,t-1}^{[m]}                   // update covariance
      i_{n,t}^{[m]} = i_{n,t-1}^{[m]} + 1                            // increment counter
    else                                                             // all other features
      μ_{n,t}^{[m]} = μ_{n,t-1}^{[m]}                                // copy old mean
      Σ_{n,t}^{[m]} = Σ_{n,t-1}^{[m]}                                // copy old covariance
      if μ_{n,t-1}^{[m]} outside perceptual range of s_t^{[m]} then  // should feature have been observed?
        i_{n,t}^{[m]} = i_{n,t-1}^{[m]}                              // no, do not change
      else
        i_{n,t}^{[m]} = i_{n,t-1}^{[m]} - 1                          // yes, decrement counter
        if i_{n,t-1}^{[m]} < 0 then discard feature n endif          // discard dubious features
      endif
    endif
  endfor
  add ⟨ s_t^{[m]}, N_t^{[m]}, ⟨μ_{1,t}^{[m]}, Σ_{1,t}^{[m]}, i_1^{[m]}⟩, ..., ⟨μ_{N_t^{[m]},t}^{[m]}, Σ_{N_t^{[m]},t}^{[m]}, i_{N_t^{[m]}}^{[m]}⟩ ⟩ to S_aux
endfor
S_t = ∅                                                              // construct new particle set
for m' = 1 to M do                                                   // resample M particles
  draw random index m with probability ∝ w_t^{[m]}                   // resample
  add ⟨ s_t^{[m]}, N_t^{[m]}, ⟨μ_{1,t}^{[m]}, Σ_{1,t}^{[m]}, i_1^{[m]}⟩, ..., ⟨μ_{N_t^{[m]},t}^{[m]}, Σ_{N_t^{[m]},t}^{[m]}, i_{N_t^{[m]}}^{[m]}⟩ ⟩ to S_t
end for
return S_t
end algorithm
```

Table 1: Summary of the algorithm FastSLAM 1.0 with unknown data association, as published in [45]. This version does not implement any of the efficient tree representations discussed in the paper, and it relies on an inferior proposal distribution. Its chief advantage is that it easier to implement than FastSLAM 2.0.

Algorithm FastSLAM 2.0(z_t, u_t, S_{t-1}):

for $m = 1$ *to* M *do* // loop over all particles

retrieve $\left\langle s_{t-1}^{[m]}, N_{t-1}^{[m]}, \left\langle \mu_{1,t-1}^{[m]}, \Sigma_{1,t-1}^{[m]}, \tau_1^{[m]} \right\rangle, \ldots, \left\langle \mu_{N_{t-1}^{[m]},t-1}^{[m]}, \Sigma_{N_{t-1}^{[m]},t-1}^{[m]}, \tau_{N_{t-1}^{[m]}}^{[m]} \right\rangle \right\rangle$ *from* S_{t-1}

for $n = 1$ *to* $N_{t-1}^{[m]}$ *do* // calculate sampling distribution

$\hat{s}_t^{[m]} = h(s_{t-1}^{[m]}, u_t)$; $\hat{z}_{t,n_t}^{[m]} = g(\mu_{n_t,t-1}^{[m]}, \hat{s}_t^{[m]})$

$G_{\theta,n_t} = \nabla_{\theta_{n_t}} g(\theta_{n_t}, s_t)\big|_{s_t=\hat{s}_t^{[m]};\theta_{n_t}=\mu_{n_t,t-1}^{[m]}}$; $G_{s,n_t} = \nabla_{s_t} g(\theta_{n_t}, s_t)\big|_{s_t=\hat{s}_t^{[m]};\theta_{n_t}=\mu_{n_t,t-1}^{[m]}}$

$Q_{t,n_t}^{[m]} = R_t + G_{\theta,n_t} \Sigma_{n_t,t-1}^{[m]} G_{\theta,n_t}^T$

$\Sigma_{s_t,n_t}^{[m]} = \left[G_{s,n_t}^T Q_{t,n_t}^{[m]-1} G_{s,n_t} + P_t^{-1} \right]^{-1}$; $\mu_{s_t,n_t}^{[m]} = \Sigma_{s_t,n_t}^{[m]} G_{s,n_t}^T Q_{t,n_t}^{[m]-1} (z_t - \hat{z}_{t,n_t}^{[m]}) + \hat{s}_t^{[m]}$

$s_{n_t,t}^{[m]} \sim \mathcal{N}(\mu_{s_t,n_t}^{[m]}, \Sigma_{s_t,n_t}^{[m]})$ // sample pose

$p_{n_t} = |2\pi Q_{t,n_t}^{[m]}|^{-\frac{1}{2}} \exp\left\{ -\frac{1}{2}(z_t - g(\mu_{n_t,t-1}^{[m]}, s_{n_t,t}^{[m]}))^T Q_{t,n_t}^{[m]-1} (z_t - g(\mu_{n_t,t-1}^{[m]}, s_{n_t,t}^{[m]})) \right\}$

endfor

$p_{N_{t-1}^{[m]}+1} = p_0$ // likelihood of new feature

$\hat{n}_t^{[m]} = \text{argmax}_{n_t}\, p_{n_t}$ **or** *draw random* $\hat{n}_t^{[m]}$ *with probability* $\propto p_{n_t}$ // data association

for $n = 1$ *to* $N_{t-1}^{[m]} + 1$ *do* // process measurement

if $n_t = \hat{n}_t \leq N_{t-1}^{[m]}$ *then* // known feature?

$N_t^{[m]} = N_{t-1}^{[m]}$; $\tau_{n_t,t}^{[m]} = \tau_{n_t,t}^{[m]} + \rho^+$; $K_t^{[m]} = \Sigma_{\hat{n}_t,t-1}^{[m]} G_{\theta,\hat{n}_t}^T Q_{t,\hat{n}_t}^{[m]-1}$

$\mu_{\hat{n}_t,t}^{[m]} = \mu_{\hat{n}_t,t-1}^{[m]} + K_t^{[m]}(z_t - \hat{z}_{t,\hat{n}_t}^{[m]})$; $\Sigma_{\hat{n}_t,t}^{[m]} = (I - K_t^{[m]} G_{\theta,\hat{n}_t}) \Sigma_{\hat{n}_t,t-1}^{[m]}$

$L_t^{[t]} = G_{s,\hat{n}_t} P_t G_{s,\hat{n}_t}^T + G_{\theta,\hat{n}_t} \Sigma_{n_t,t-1}^{[m]} G_{\theta,\hat{n}_t}^T + R_t$

$w_t^{[m]} = |2\pi L_t^{[t]}|^{-\frac{1}{2}} \exp\left\{ -\frac{1}{2}(z_t - \hat{z}_{t,\hat{n}_t})^T L_t^{[t]-1} (z_t - \hat{z}_{t,\hat{n}_t}) \right\}$

else if $n_t = \hat{n}_t = N_{t-1}^{[m]} + 1$ *then* // new feature?

$n = N_t^{[m]} = N_{t-1}^{[m]} + 1$; $\tau_{n_t,t}^{[m]} = \rho^+$; $w_t^{[m]} = p_0$

$s_{n,t}^{[m]} \sim p(s_t \mid s_{t-1}^{[m]}, u_t)$; $G_{\theta,n} = \nabla_{\theta_n} g(\theta_n, s_t)\big|_{s_t=s_{n,t}^{[m]};\theta_n=\mu_{n,t}^{[m]}}$

$\mu_{n,t}^{[m]} = g^{-1}(z_t, s_{n,t}^{[m]})$; $\Sigma_{n,t}^{[m]} = (G_{\theta,n} R_t^{-1} G_{\theta,n}^T)^{-1}$

else if $n_t \neq \hat{n}_t$ *and* $n_t \leq N_{t-1}^{[m]}$ // handle unobserved features

$\mu_{n_t,t}^{[m]} = \mu_{n_t,t-1}^{[m]}$; $\Sigma_{n_t,t}^{[m]} = \Sigma_{n_t,t-1}^{[m]}$

if $\mu_{n_t,t}^{[m]} \notin \text{range}(s_{\hat{n}_1,t}^{[m]})$ *then* // outside sensor range?

$\tau_{n_t,t}^{[m]} = \tau_{n_t,t-1}^{[m]}$

else // inside sensor range?

$\tau_{n_t,t}^{[m]} = \tau_{n_t,t-1}^{[m]} - \rho^-$; *if* $\tau_{n_t,t}^{[m]} < 0$ *then remove* n_t // discontinue feature?

endif

endif

endfor

add $\left\langle s_t^{[m]}, N_t^{[m]}, \left\langle \mu_{1,t}^{[m]}, \Sigma_{1,t}^{[m]}, \tau_1^{[m]} \right\rangle, \ldots, \left\langle \mu_{N_t^{[m]},t}^{[m]}, \Sigma_{N_t^{[m]},t}^{[m]}, \tau_{N_t^{[m]}}^{[m]} \right\rangle \right\rangle$ *to* S_{aux}

endfor // end loop over all particles

$S_t = \emptyset$ // construct new particle set

for $m' = 1$ *to* M *do* // generate M particles

draw random index m *with probability* $\propto w_t^{[m]}$ // resample

add $\left\langle s_t^{[m]}, N_t^{[m]}, \left\langle \mu_{1,t}^{[m]}, \Sigma_{1,t}^{[m]}, \tau_1^{[m]} \right\rangle, \ldots, \left\langle \mu_{N_t^{[m]},t}^{[m]}, \Sigma_{N_t^{[m]},t}^{[m]}, \tau_{N_t^{[m]}}^{[m]} \right\rangle \right\rangle$ *to* S_t

end for

return S_t

end algorithm

Table 2: The FastSLAM 2.0 Algorithm, stated here unknown data association.

7 Convergence of FastSLAM 2.0 For Linear-Gaussian SLAM

In this section, we will establish a convergence result for FastSLAM 2.0. This result crucially exploits the proposal distribution in FastSLAM 2.0 and therefore is not directly applicable to FastSLAM 1.0. It applies to a subset of all SLAM problems, namely for linear SLAM problems with Gaussian noise. LG-SLAM problems are defined to possess motion and measurement models of the following linear form:

$$g(\theta_{n_t}, s_t) = \theta_{n_t} - s_t \tag{77}$$

$$h(u_t, s_{t-1}) = u_t + s_{t-1} \tag{78}$$

The LG-SLAM framework can be thought of as a robot operating in a Cartesian space equipped with a noise-free compass, and sensors that measure distances to features along the coordinate axes.

While LG-SLAM is clearly too restrictive to be of practical significance, it plays an important role in the literature. To our knowledge, the only known convergence proof for a SLAM algorithm is a recently published result for Kalman filters (KF) applied to specific linear-Gaussian problems. As shown in [17, 54], the KF approach (which is equivalent to EKFs for linear-Gaussian SLAM problems) converges to a state in which all map features are fully correlated. If the location of one feature is known, the KF asymptotically recovers the location of all other feature.

The central convergence result in this paper is the following:

Theorem. *Linear-Gaussian FastSLAM 2.0 converges in expectation to the correct map with $M = 1$ particle if all features are observed infinitely often, and if the location of one feature is known in advance.*

If no feature location is known in advance, the map will be correct in relative terms, up to a fixed offset that uniformly applies to all features. The proof of this result can be found in Appendix B. Its significance lies in the fact that is shows that for specific SLAM problems, FastSLAM 2.0 may converge with a finite number of particles. In particular, the number of particles required for convergence in LG-SLAM is independent of the size of the map N. This result holds even if all features are arranged in a large cycle, a situation often thought of as worst case for SLAM problems [26]. However, our analysis says nothing about the convergence speed of the algorithm, which in practice depends on the particle set size M. Below, we will investigate the speed of convergence through empirical means.

8 Efficient Implementation

At first glance, it may appear that each update in FastSLAM requires time $O(MN)$, where M is the number of particles M and N the number of features in the map. The linear complexity in M is unavoidable, given that we have to process M particles with each update. The linear complexity in N is the result of the resampling process. Whenever a particle is drawn more than once in the resampling process, a "naive" implementation might duplicate the entire map attached to the particle. Such a duplication process is linear in the size of the map N. Furthermore, a naive implementation of data association may result in evaluating the measurement likelihood for each of the N features in the map, resulting again in linear complexity in N. We note that a poor implementation of the sampling process might easily add another factor of $logN$ to the update complexity.

FastSLAM iterations can be executed in $O(M \log N)$ time; in particular, FastSLAM updates can be implemented in time logarithmic in the size of the map N. First, consider the situation with known data association. Linear copying costs can be avoided by introducing a data structure for representing particles that allow for more selective updates. The basic idea is to organize the map as a balanced binary tree. Figure 9a shows such a tree for a single particle, in the case of $K = 8$ features. Notice that

(a)

(b)

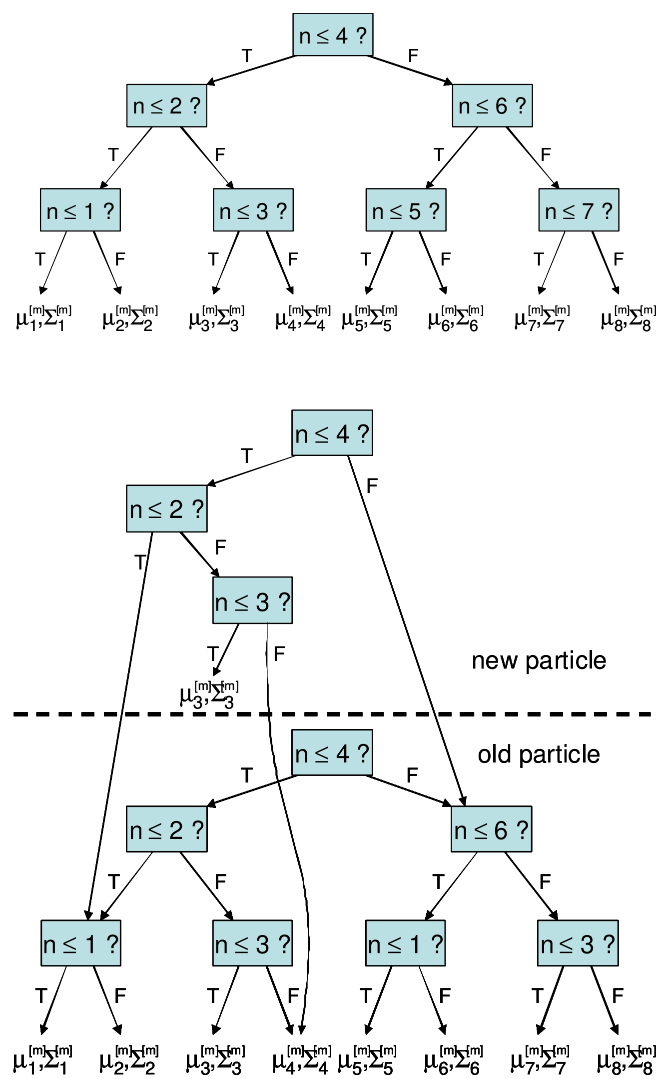

Figure 9: (a)A tree representing $N = 8$ feature estimates within a single particle. (b) Generating a new particle from an old one, while modifying only a single Gaussian. The new particle receives only a partial tree, consisting of a path to the modified Gaussian. All other pointers are copied from the generating tree. This can be done in time logarithmic in N.

the Gaussian parameters $\mu_k^{[m]}$ and $\Sigma_k^{[m]}$ are located at the leaves of the tree. Assuming that the tree is balanced, accessing a leaf required time logarithmic in N.

Suppose FastSLAM incorporates a new control u_t and a new measurement z_t. Each new particle in S_t will differ from the corresponding one in S_{t-1} in two ways: First, it will possess a different pose estimate obtained via (33), and second, the observed feature's Gaussian will have been updated, as specified in Equations (54) and (55). All other Gaussian feature estimates, however, will be equivalent to the generating particle. When copying the particle, thus, only a single path has to be modified in the tree representing all Gaussians. An example is shown in Figure 9b: Here we assume $n_t = 3$, that is, only the Gaussian parameters $\mu_3^{[m]}$ and $\Sigma_3^{[m]}$ are updated. Instead of generating an entire new tree, only a single path is created, leading to the Gaussian $n_t = 3$. This path is an incomplete tree. The tree is completed by copying the missing pointers from the tree of the generating particle. Thus, branches that leave the path will point to the same (unmodified) subtree as that of the generating tree. Clearly, generating this tree takes only time logarithmic in N. Moreover, accessing a Gaussian also takes time logarithmic in

Figure 10: The utility car used for collecting outdoor data is equipped with a SICK laser range and bearing sensor, linear variable differential transformer sensor for the steering and back wheel velocity encoder. This image shows the vehicle in the Victoria Park environment.

N, since the number of steps required to navigate to a leaf of the tree is equivalent to the length of the path (which is by definition logarithmic). Thus, both generating and accessing a partial tree can be done in time $O(\log N)$. Since in each updating step M new particles are created, an entire update requires time in $O(M \log N)$. The insight of using trees for efficient mapping can be found in [45]; a similar tree representation can be found in [21].

Organizing particles in trees raises the question as to when to deallocate memory. Memory deallocation can equally be implemented in amortized logarithmic time. The idea is to assign a variable to each node—internal or leaf—that counts the number of pointers pointing to it. The counter of a newly created node will be initialized by 1. It will be incremented as pointers to a node are created in other particles. Decrements occur when pointers are removed (e.g., pointers of pose particles that fail to survive the resampling process). When a counter reaches zero, its children's counters are decremented and the memory of the corresponding node is deallocated. The processes is then applied recursively to all children of the node whose counter may have reached zero. This recursive process will require $O(M \log N)$ time on average. Furthermore, it can be shown to be an optimal deallocation algorithm in that all unneeded memory will be freed instantaneously.

To obtain logarithmic time complexity for FastSLAM with unknown data association further assumptions are needed. In particular, the number features in the robot's sensor range must be independent of N; otherwise simple operations such as keeping track of the existence posteriors $\tau_n^{[m]}$ may require more than logarithmic time. Furthermore, a DAS sampler must be restricted to features in the robot's vicinity to avoid calculating the likelihood (63) for all N features. Finally, the number of rejected features should be small (e.g., within a constant factor of all accepted ones). All these assumptions are plausible when applying FastSLAM to real-world SLAM problems. Under these assumptions, variants of kd-trees [6, 48] can guarantee logarithmic time search for high likelihood features, and features in the robot's measurement range. Incremental techniques for constructing balanced kd-trees are described in [39, 59]. For example, the bkd-tree proposed in [59] maintains a sequence of trees of growing complexity. By carefully shifting items across those trees, a logarithmic time recall can be guaranteed under amortized logarithmic time for inserting new features in the map. In this way, all necessary operations in FastSLAM can be carried out in logarithmic time on average.

9 Experimental Results: FastSLAM 1.0

A number of experimental comparisons were carried out, comparing both FastSLAM algorithms with each other and to the popular EKF solution to the SLAM problem. The goal of these experiments were to investigate the scaling properties of each algorithm, especially in relation to a classical solution to the SLAM problem. The experiments were carried out using a benchmark dataset in the SLAM field known as the "Victoria Park Dataset" [25]; supplemental experiments were obtained using a second real-world data set obtained in a parking lot, and through robot simulation. Further experiments can be found in [43],

In this section, we will describe our experiments for FastSLAM 1.0, illustrating that even this simple-to-implement algorithm yields excellent results. The next section characterizes the advantages of the more complex FastSLAM 2.0 algorithm over FastSLAM 1.0.

9.1 Victoria Park

The benchmark SLAM data set used in most of our experiments was provided by researchers from the University of Sydney [25]. An instrumented vehicle, shown in Figure 10, equipped with a laser range finder was repeatedly driven through Victoria Park, in Sydney, Australia. Victoria Park is an ideal setting for testing feature-based SLAM algorithms because the park's trees are distinctive features in the vehicle's laser scans. Encoders measured the vehicle's velocity and steering angle. Range and bearing measurements to nearby trees were extracted from the laser data using a local minima detector. The vehicle was driven around for approximately 30 minutes, covering a distance of over 4 km. The vehicle is also equipped with GPS in order to capture ground truth data. Due to occlusion by foliage and buildings, ground truth data is only available for part of the overall traverse. While ground truth is available for the vehicle's path, no ground truth data is available for the locations of the landmarks. None of the GPS data was used for mapping; the sole function of this data is to provide ground truth for evaluating the accuracy of the filter.

Since the vehicle is driving over uneven terrain, the measured controls are fairly noisy. Figure 11 (a) shows the path of the vehicle obtained by integrating the estimated controls. After 30 minutes of driving, the estimated position of the vehicle is well over 100 meters away from its true position measured by GPS. The laser data, on the other hand, is a very accurate measure of range and bearing. However, not all objects in the vehicle's field of view are trees, or even static objects. As a result, the feature detector produced relatively accurate observations of trees, but also generated frequent outliers.

Data association for this experiment was performed using per-particle ML data association. Since the accuracy of the observations is high relative to the average density of landmarks, data association in the Victoria Park data set is a relatively straightforward problem. In a later experiment, more difficult data association problems will be simulated by adding extra control noise.

The output of FastSLAM 1.0 is shown in Figure 11 b&c. The GPS path is shown as a dashed line, and the output of FastSLAM 1.0 is shown as a solid line. The RMS error of the resulting path is just over 4 meters over the 4 km traverse. This experiment was run with $M = 100$ particles. This error is indistinguishable from the error of other state-of-the-art SLAM algorithms [25, 38].

9.2 Performance Without Odometry

FastSLAM 1.0 was also run on the Victoria Park data set without using the odometry data. The pose of the vehicle in each particle was supplemented with translational velocity v_t and rotational velocity w_t.

$$S_t^{[m]} = \langle s_{x,t}, s_{y,t}, s_{\theta,t}, s_{v,t}, s_{w,t}, N_t, \mu_{1,t}, \Sigma_{1,t}, \ldots, \mu_{N_t,t}, \Sigma_{N_t,t} \rangle \tag{79}$$

A Brownian motion model was used to predict the pose of the vehicle at time $t+1$ given the pose at time t. This model assumes that the velocity at time $t+1$ is equal to the velocity at time t plus some random

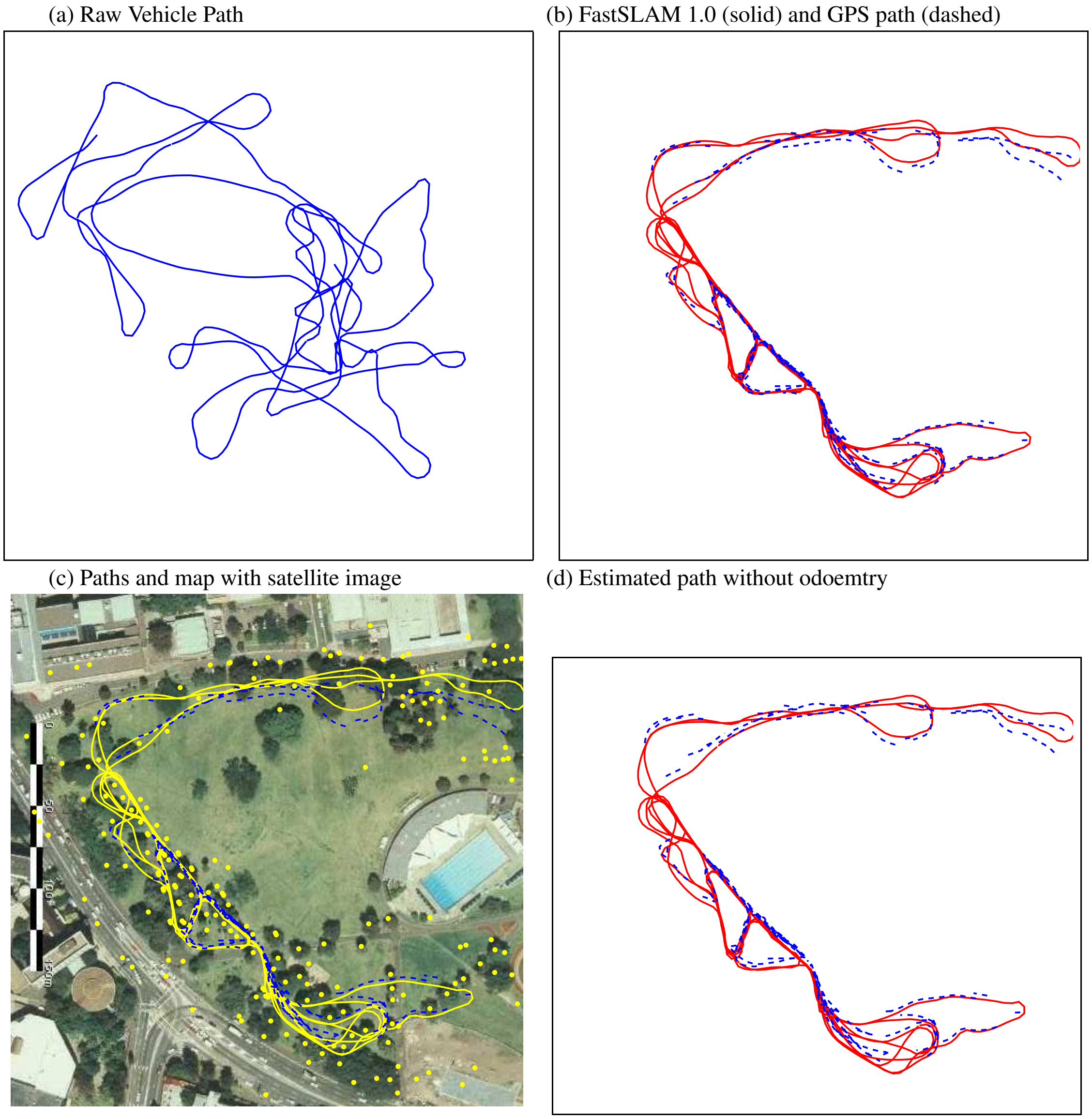

Figure 11: (a) Vehicle path predicted by the odometry; (b) True path (dashed line) and FastSLAM 1.0 path (solid line); (c) Victoria Park results overlayed on aerial imagery with the GPS path in blue (dashed), average FastSLAM 1.0 path in yellow (solid), and estimated landmarks as yellow circles. (d) Victoria Park Map created without odometry information.

perturbation.

$$\begin{aligned} v_t &= v_{t-1} + \mathcal{N}(v; 0, \alpha_1^2) \\ w_t &= w_{t-1} + \mathcal{N}(w; 0, \alpha_2^2) \end{aligned} \tag{80}$$

After drawing a perturbed velocity, the vehicle's position is updated accordingly.

$$\begin{aligned} x_t &= x_{t-1} + v_t \cos(\theta_{t-1})\Delta t \\ y_t &= y_{t-1} + v_t \sin(\theta_{t-1})\Delta t \\ \theta_t &= \theta_{t-1} + w_t \Delta t \end{aligned} \tag{81}$$

The specific values of α_1 and α_2 depend on the maximum translational and rotational accelerations that the vehicle is capable of executing. The map created without using the odometry is shown in Figure 11d. The average error of the map is equivalent to the results obtained with odometry. To our knowledge, no previous technique has been capable of generating consistent maps from this data set without using odometry information.

(a) Map without feature elimination (b) Map with feature elimination

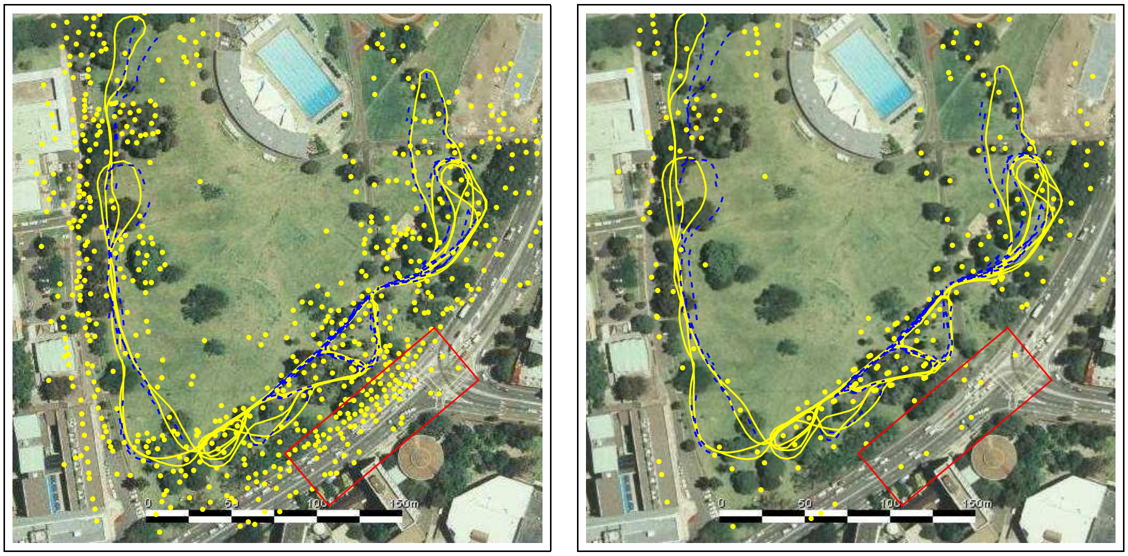

Figure 12: FastSLAM 1.0 (a) without and (b) with feature elimination based on negative information.

9.3 Negative Information

In the Victoria Park data set, observations corresponding to non-point objects or non-static objects result in a large number of spurious landmarks being added to every FastSLAM 1.0 particle. When negative information is used to estimate the existence of each landmark, as described in Section 6.4, many of these spurious landmarks can be removed. In the case of Victoria Park, use of negative information results in 44map. While the *correct* number of landmarks is not available, visual inspection of the maps suggests that many of the spurious features have been eliminated. Figure 12 shows the Victoria Park map built with and without considering negative evidence. The number of landmarks in areas that should be free of landmarks (the roadway, highlighted with a box in the figure) has been significantly reduced.

9.4 Comparison of FastSLAM 1.0 and the EKF

9.5 Accuracy

We compared the accuracy of FastSLAM 1.0 with that of the EKF on a simulated data set with 100 landmarks. The RMS vehicle pose error was computed for FastSLAM 1.0 for various numbers of particles from 1 to 5000. Each experiment was run 10 times. The results are shown in Figure 13. The error of the EKF is shown as a dashed horizontal line.

In this experiment, the accuracy of FastSLAM 1.0 approaches the accuracy of the EKF as the number of particles is increased. Most notably, the error of FastSLAM 1.0 becomes statistically indistinguishable from that of the EKF past approximately 10 particles. This is interesting because FastSLAM 1.0 with 10 particles and 100 landmarks requires an order of magnitude fewer parameters than the EKF in order to achieve this level of accuracy. Clearly, the specific value of this threshold of performance will depend on both the parameters of the motion and measurement model and the vehicle's control policy. However, this experiment suggests that in normal circumstances, a relatively small number of particles may suffice to achieve high estimation accuracy.

9.6 Scaling Performance

The scaling performance of FastSLAM 1.0 was also evaluated on simulated data. Simulated maps of constant landmark density were created with varying numbers of landmarks. Constant landmark density

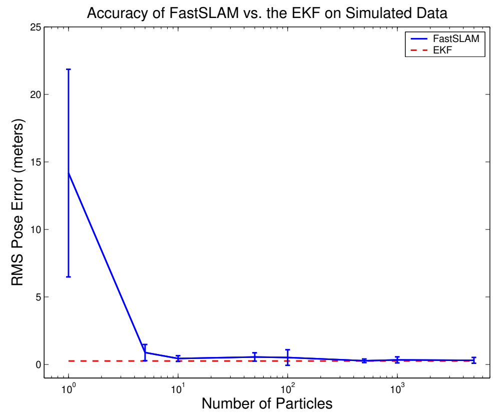

Figure 13: A comparison of the accuracy of FastSLAM 1.0 and the EKF on simulated data

ensures that the simulated vehicle observed a constant number of landmarks on average across all trials. The performance of the linear time and logarithmic time versions of the FastSLAM 1.0 algorithm were compared. The linear time algorithm was tested with up to 10,000 landmarks, and the logarithmic time algorithm was tested with up to 1,000,000 landmarks. The time required to compute 500 sensor updates with all landmarks incorporated into the map was evaluated over 10 different runs. All experiments were done with 100 particles.

The results of the experiment are shown in Figure 14. The performance of the $\log(N)$ algorithm is plotted on a logarithmic scale. The results validate the scaling performance of the tree-based algorithm, and demonstrate the substantial performance increase enabled by sharing landmark trees across particles.

Sharing subtrees is not only computationally efficient; it also decreases the overall memory required by the algorithm. The memory required by both versions of the FastSLAM 1.0 algorithm scales linearly with the number of landmarks. Overall, the FastSLAM 1.0 algorithm must maintain $M \cdot N$ landmark filters. With 100 particles and 1,000,000 landmarks, this can add up to a substantial amount of memory (hundreds of megabytes) just to represent the map. In very large maps, landmarks that have not been visited for a long period of time will be shared in subtrees between all of the particles of the $\log(N)$ algorithm. If only a fraction of the total landmarks are observed at every time step, this memory sharing may result in a significant savings in memory consumption. A plot of the memory consumed by the linear and logarithmic FastSLAM 1.0 algorithms for varying numbers of landmarks is shown in Figure 15. In this experiment, the tree-based representation resulted in over an order-of-magnitude decrease in memory consumption over the basic FastSLAM 1.0 algorithm.

9.7 Ambiguous Data Association

The performance of FastSLAM 1.0 given unknown data association was evaluated against that of the Extended Kalman Filter using the Victoria Park data set. Under normal conditions, the levels of odometric and measurement noise present in the Victoria Park data set do not cause a significant data association problem. The error of the vehicle's laser is quite low compared to the average distance between trees in the park. In order to test performance given data association ambiguity, additional odometric noise was added to the vehicle controls. Additional control noise results in high motion ambiguity in the data associations of new observations.

Prototypical outputs of the EKF and FastSLAM 1.0 given low and high levels of odometric noise are shown in Figure 16. While both algorithms generate accurate maps when control noise is low, the EKF fails catastrophically with high error. The map generated by FastSLAM 1.0 under high odometric error is not degraded in quality. The RMS error of the vehicle position was computed for FastSLAM 1.0 and

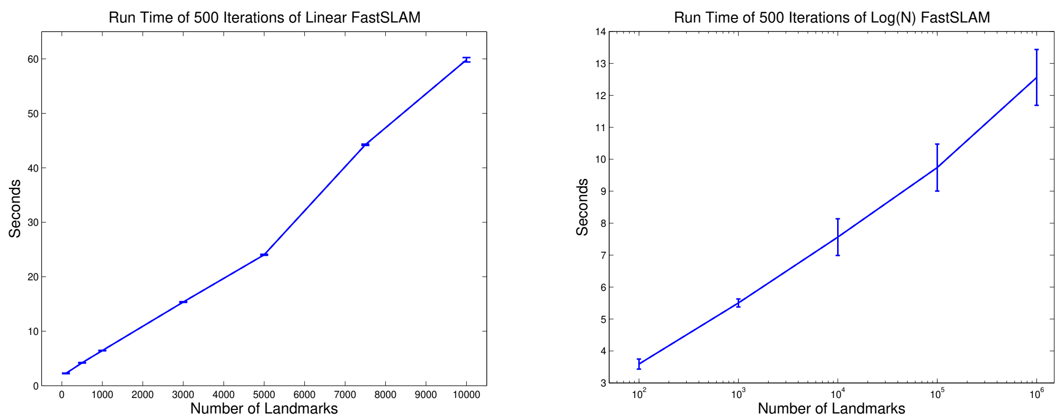

Figure 14: Timing results for FastSLAM 1.0 in simulated environments.

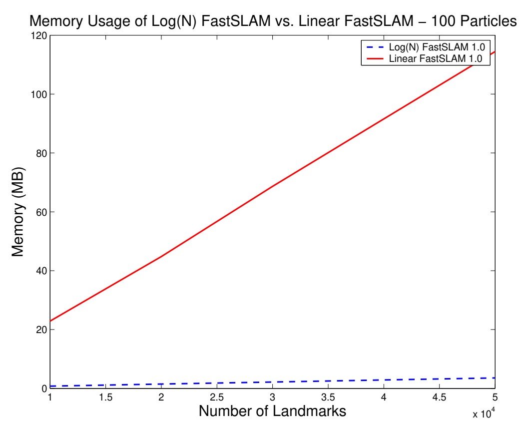

Figure 15: Memory requirements for linear and $\log(N)$ version of FastSLAM 1.0 in simulated environments.

the EKF over 20 different runs with four different levels of odometric noise. The results are shown in Figure 17. As control noise increases, there is no measurable increase in the RMS error of FastSLAM 1.0, while the error of the vehicle path emitted by the EKF goes up substantially. More telling is the variance in the error of the EKF maps across multiple runs, indicated by the confidence bars. This suggests that for high levels of control noise, the EKF is diverging.

9.8 Results for Parking Lot Database:

The next series of experiments was conducted using data collected on the top level of the car park building at the University of Sydney. The testing site was chosen to maximize the number of satellites in view to obtain high quality GPS information. A kinematic GPS system of 2 cm CEP accuracy was used to measure the vehicle location for evaluation. In this experiment, artificial landmarks were used that consisted of 60 mm steel poles covered with reflective tape. With this, the feature extraction becomes trivial and the landmark observation model accurate. Since the true position of the landmarks were also obtained with GPS, a true navigation map is also available for comparison purposes.

In our first series of experiments, the correspondences between the observation and the landmarks were assumed to be known. Instead of providing this information manually, we used a highly tuned EKF to provide the correct data association for each measurement. The EKF algorithm was run with the same data set, dropping all measurements that were not associated with any landmark. As a result, FastSLAM 1.0 only received measurements corresponding to actual landmarks in the environment. Figure 18a

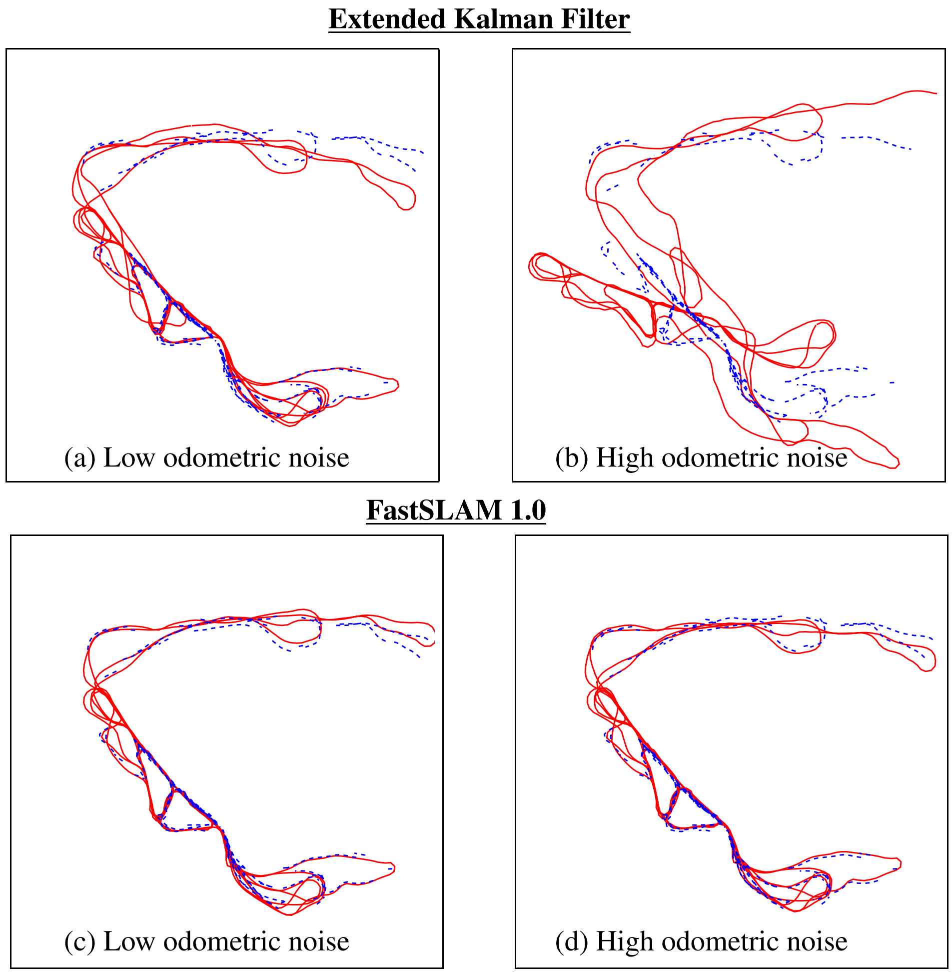

Figure 16: Performance of EKF and FastSLAM 1.0 on the Victoria Park data set with varying levels of odometric noise.

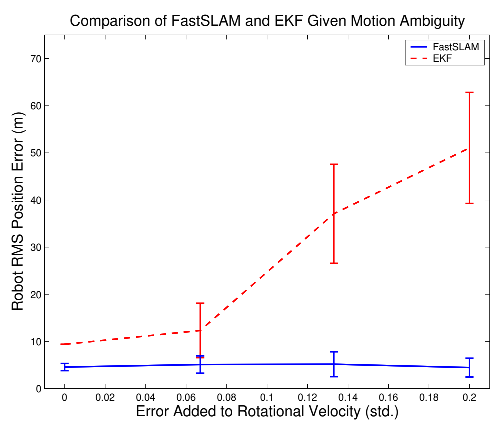

Figure 17: Position error of vehicle under various levels of odometric noise.

shows the path and beacons position estimation for the car park experimental run using the algorithm FastSLAM 1.0. This figure shows the particles average for the vehicle trajectory and the average of all the Gaussian means for the landmarks locations. Figure 18b shows similar results obtained with the EKF based algorithm. Figure 19 presents the vehicle position error for the EKF and FastSLAM 1.0 filter respectively. It can be appreciated that the error is very small and similar in magnitude and shape when compared with the GPS ground truth. This is important to verify the consistency of the algorithm

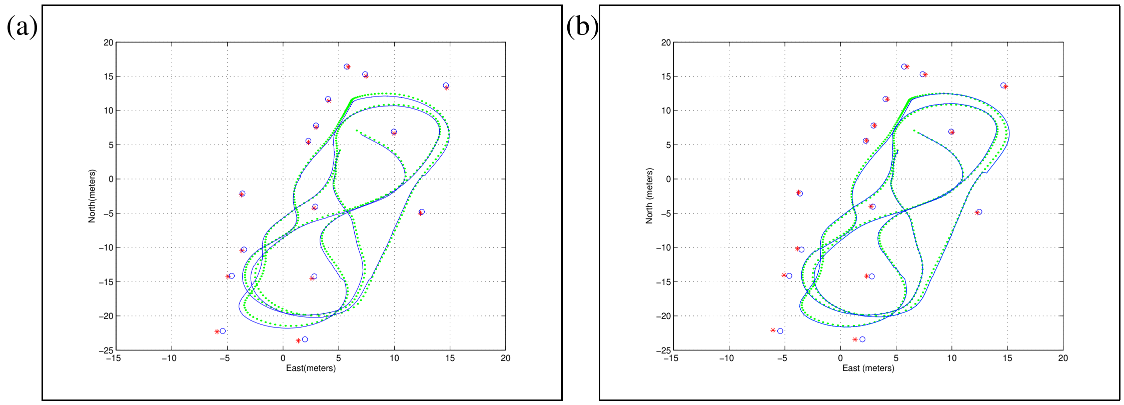

Figure 18: Estimated path and landmarks with (a) FastSLAM 1.0 and (b). The '-' is the path estimated, the '*' are the beacons position estimated, the '.' is the GPS path reference and the 'o' are the beacons position given by the GPS.

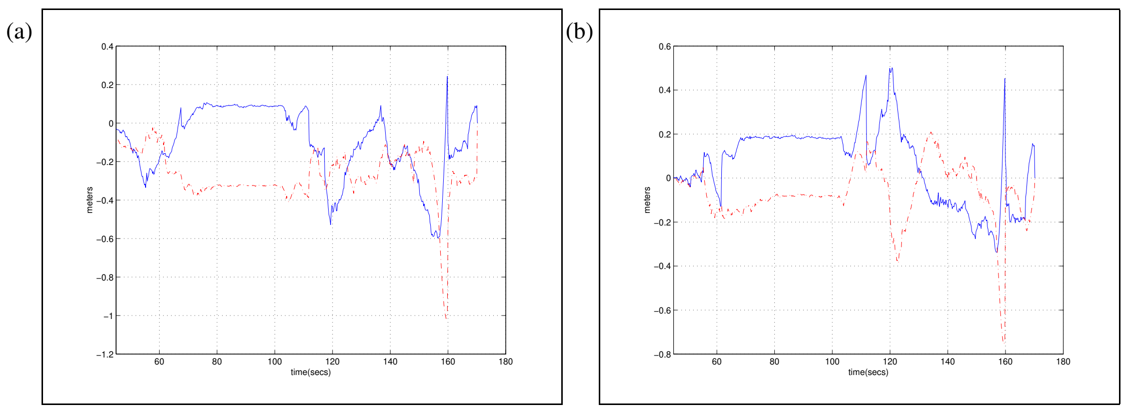

Figure 19: SLAM error (a) FastSLAM 1.0 position error respect to the GPS position. '-' indicates the error in the East and '-.'in North (b) EKF position error respect to the GPS position. '-' indicates the error in East and '-.' in North.

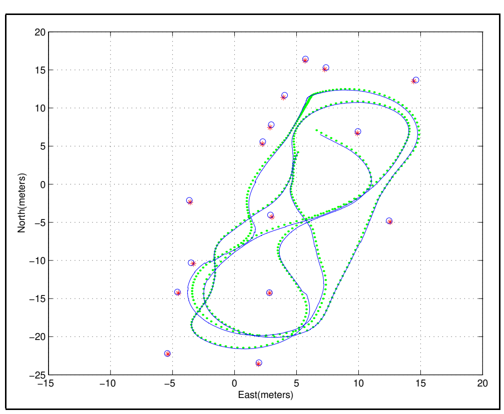

Figure 20: Estimated Path and Landmarks with unknown data association. The '-' is the path estimated, the '*' are the beacons position estimated, the '.' is the GPS path reference and the 'o' are the beacons position given by the GPS.

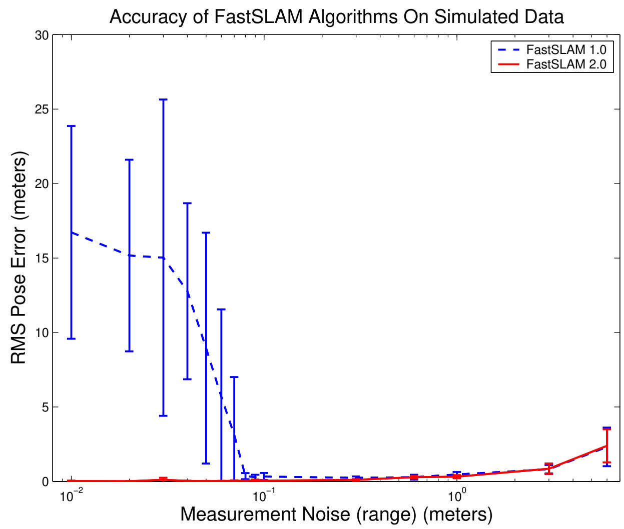

Figure 21: FastSLAM 1.0 and 2.0 with varying levels of measurement noise: As to be expected, FastSLAM 2.0 is uniformly superior to FastSLAM 1.0. The difference is particularly obvious for small particle sets, where the improved proposal distribution focuses the particles much better.

FastSLAM 1.0. Figure 20 shows the corresponding results for a run with unknown data association. The accuracy of the resulting map is comparable to our results with known data association, illustrating that FastSLAM 1.0 succeeds in solving the data association problem in this instance.

10 Experimental Results: FastSLAM 2.0

10.1 Comparison of FastSLAM 2.0 and 1.0

In general, FastSLAM 2.0 is superior to FastSLAM 1.0, but the performance of FastSLAM 2.0 will often be similar to the performance of FastSLAM 1.0. However, in situations where the measurement error is significantly small compared to the motion error, FastSLAM 2.0 will outperform the original algorithm. In the following experiment, the performance of the two algorithms is compared on simulated data while varying the level of measurement noise. All experiments were run with 100 particles and known data association. The range and bearing error parameters were scaled equally.

The results of this experiment are shown in Figure 21. As the measurement error gets very large, the errors of both FastSLAM 1.0 and 2.0 begin to increase slowly, as expected. In this range, the performance of the two algorithms is roughly equal. For very low values of measurement error, FastSLAM 1.0 clearly begins to diverge, while the error of FastSLAM 2.0 continues to shrink. By adding more particles, the threshold below which FastSLAM 1.0 diverges can be decreased. However, FastSLAM 2.0 can produce accurate maps in these situations without increasing the number of particles.

The performance of the two algorithms can also be compared by keeping the measurement model constant and varying the number of particles. FastSLAM 2.0 will require fewer particles than FastSLAM 1.0 in order to achieve a given level of accuracy, especially when measurement error is low. In the limit, FastSLAM 2.0 can produce reasonable maps with just a single particle, while FastSLAM 1.0 will diverge. Figure 22 shows the results of an experiment comparing the performance of FastSLAM 1.0 and 2.0 given different numbers of particles. The two algorithms were run repeatedly on simulated data and the Victoria Park data set. On the simulated data, the accuracy of the two algorithms is similar with more than five particles. Below five particles, FastSLAM 1.0 begins to diverge and the performance of FastSLAM 2.0 stays approximately constant.

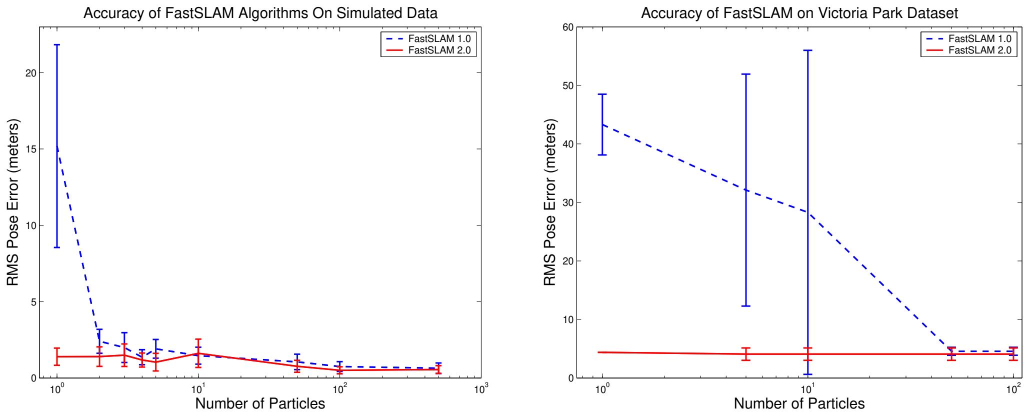

Figure 22: Performance of FastSLAM algorithms with different numbers of particles.

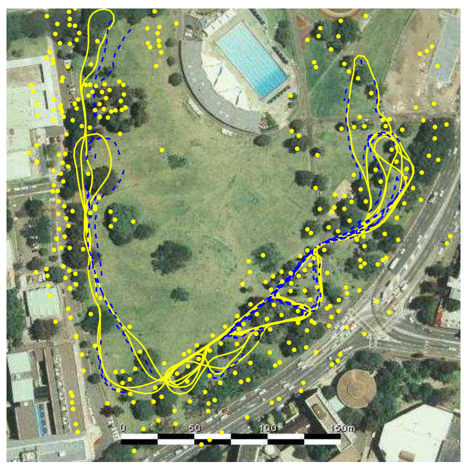

Figure 23: Map of Victoria Park by FastSLAM 2.0 with $M = 1$ particle.

On the Victoria Park dataset the difference between the two algorithms is even more pronounced. Below 50 particles, FastSLAM 1.0 starts to diverge. Again, this is because the vehicle's controls are noisy relative to the sensor observations.

10.2 One Particle FastSLAM 2.0

The data associations in the Victoria Park data set are relatively unambiguous, so the one particle version of FastSLAM 2.0 can be used. With only a single particle, data association in FastSLAM 2.0 is equivalent to the maximum likelihood data association algorithm of the EKF. Figure 23 shows the output of FastSLAM with a single particle. The algorithm is able to produce results on par with those of the EKF and FastSLAM 1.0 without storing any correlations between landmarks.

10.3 Scaling Performance

The experiment in Section 10.1 demonstrates that FastSLAM 2.0 requires fewer particles than FastSLAM 1.0 in order to achieve a given level of estimation accuracy. Fewer particles, in turn, results in faster sensor updates. However, the construction of the improved proposal distribution requires extra time over the FastSLAM 1.0 proposal. As the number of landmarks in the map increases, the sensor updates take a smaller fraction of the overall run time relative to the importance resampling. In larger maps, the large

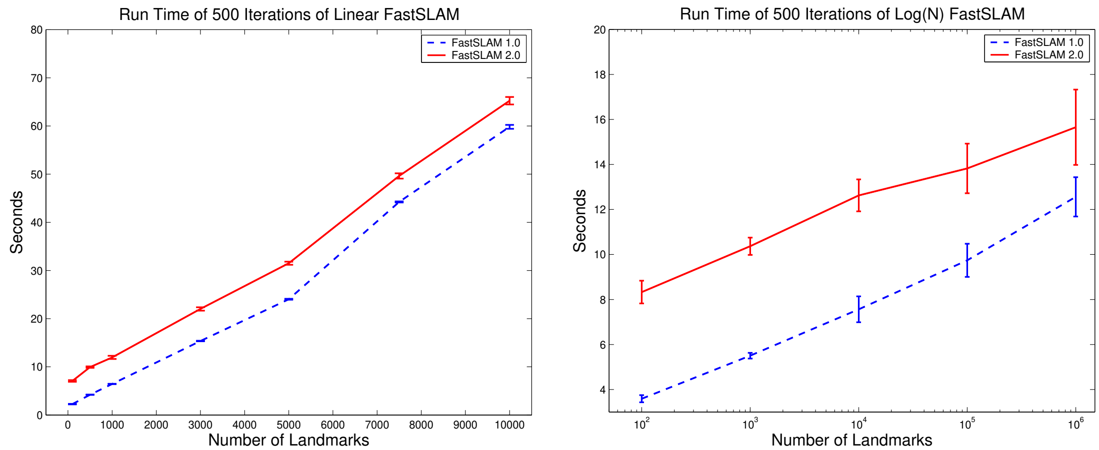

Figure 24: Comparison of FastSLAM 1.0 and FastSLAM 2.0 timing.

savings gained as a result of needing fewer particles overwhelms the additional complexity of drawing from the proposal distribution. The actual savings will depend on the parameters of the motion and measurement models. Figure 24 shows the run time for the linear and $\log(N)$ versions of FastSLAM 1.0 and 2.0 all with 100 particles. In very small maps (i.e. 100 landmarks), FastSLAM 2.0 requires approximately 3 times longer to perform a sensor update. However, in larger maps the sensor updates only require 10-20% more time. The constant difference between FastSLAM 1.0 and 2.0 with an equal number of particles depends primarily on the average number of observations incorporated per time step.

10.4 Loop Closing

In FastSLAM, the ability to close loops effectively depends on the number of particles M. The minimum number of particles is difficult to quantify, because it depends on a number of factors, including the parameters of the motion and measurement models and the density of landmarks in the environment. FastSLAM 2.0's improved proposal distribution insure that fewer particles are eliminated in resampling compared to FastSLAM 1.0. Better diversity in the sample set results in better loop closing performance, because new observations can affect the pose of the vehicle further back in the past.

Examples of loop closing with FastSLAM 1.0 and FastSLAM 2.0 are shown in Figure 25a&b, respectively. The histories of all M particles are drawn for both algorithms. In Figure 25a, the FastSLAM 1.0 particles share a common history part of the way around the loop. New observations can not affect the positions of landmarks observed before this threshold. In this case of FastSLAM 2.0, the algorithm is able to maintain diversity that extends back to the beginning of the loop. This is crucial for reliable loop closing and fast convergence.

Figure 26a shows the result of an experiment comparing the loop closing performance of FastSLAM 1.0 and 2.0. As the size of the loop increases, the error of both algorithms increases. However, FastSLAM 2.0 consistently outperforms FastSLAM 1.0. Alternately, this result can rephrased in terms of particles. FastSLAM 2.0 requires fewer particles to close a given loop than FastSLAM 1.0.

10.5 Convergence Speed

By pruning away improbable trajectories of the vehicle, resampling eventually causes all of the FastSLAM particles to share a common history at some point in the past. New observations cannot affect the positions of landmarks observed prior to this point. This common history point can be pushed back in time by increasing the number of particles M. This process of throwing away correlation data over time enables FastSLAM's efficient sensor updates. This efficiency comes at the cost of slower convergence

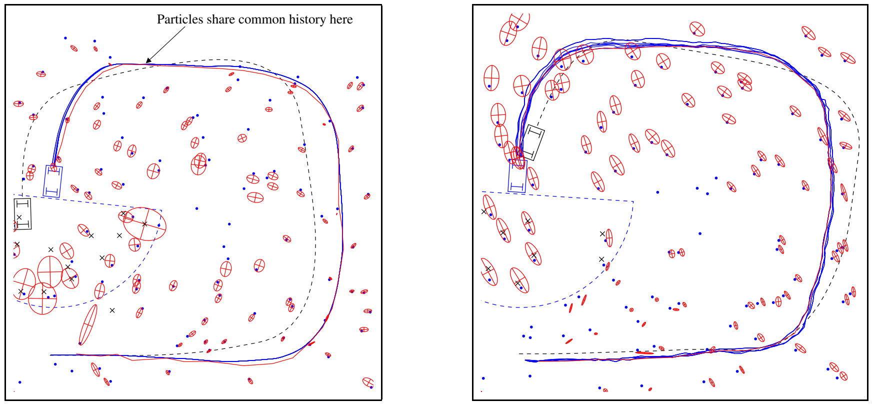

Figure 25: FastSLAM 2.0 can close larger loops than FastSLAM 1.0 given a constant number of particles.

speed. Throwing away correlation information means that more observations will be required to achieve a given level of accuracy.

The trade-off of number of particles versus convergence speed occurs in both FastSLAM 1.0 and FastSLAM 2.0. However, FastSLAM 2.0 can operate without maintaining any cross-correlations between landmarks, so the relationship effect of throwing away correlation data on convergence speed is easier to study. In particular, this effect is most prominent when closing a large loop. Revisiting a known landmark should refine the positions of all landmarks around the loop. If correlation information is thrown away, convergence to the true map will be slower, and more trips around the loop will be necessary to achieve the same level of accuracy.

Figure 26b shows the results of an experiment comparing the convergence speed of FastSLAM 2.0 and the EKF. FastSLAM 2.0 (with 1, 10, and 100 particles) and the EKF were each run 10 times around a large simulated loop of landmarks, similar to the ones shown in Figure 26a&b. Different random seeds were used for each run, causing different controls and observations to be generated for each loop. The RMS position error in the map at every time step was averaged over the 10 runs for each algorithm.

As the vehicle goes around the loop, error should gradually build up in the map. When the vehicle closes the loop at iteration 150, revisiting old landmarks should affect the positions of landmarks all around the loop, causing the overall error in the map to decrease. This clearly happens in the EKF. FastSLAM 2.0 with a single particle has no way to affect the positions of past landmarks so there is no drop in the landmark error. As more particles are added to FastSLAM 2.0, the filter is able to apply observations to landmark positions further back in time, gradually approaching the convergence speed of the EKF. Clearly, the number of particles necessary to achieve convergence time close to the EKF will increase with the size of the loop. It is unknown at this time whether the number of particles necessary to achieve a given accuracy is polynomial or exponential in the size of the loop. The lack of long-range correlations in the FastSLAM representation is arguably the most important weakness of FastSLAM algorithm over previous EKF-style techniques.

11 Discussions

This article described FastSLAM, a new family of algorithms for the simultaneous localization and mapping (SLAM) problem. Like many previously published SLAM algorithms, FastSLAM calculates posterior probability distributions over featured maps and robot locations. It does so recursively, that is, the current estimate is calculated from the estimate one time step earlier, using the data accrued in between. FastSLAM is based on a key property of the SLAM problem: the conditional independence of feature estimates given the vehicle path. This conditional independence gives rise to a factored representation of

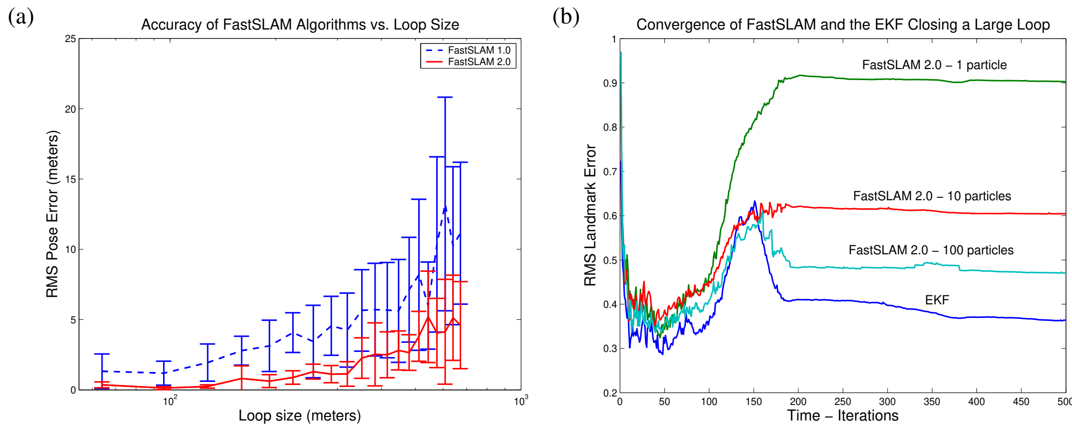

Figure 26: (a) Accuracy as a function of loop size: FastSLAM 2.0 can close larger loops than FastSLAM 1.0 given a fixed number of particles. (b) Comparison of the convergence speed of FastSLAM 2.0 and the EKF.

the posterior that can be updated more efficiently than unstructured, monolithic posteriors. FastSLAM represents this factored posterior using a combination of particle filters for estimating the robot path and Kalman filters for estimating the map. The use of particle filters enables FastSLAM to sample over data association hypotheses in SLAM problems with unknown data associations. This article described two instantiations of this idea: The original FastSLAM algorithm, coined here FastSLAM 1.0, is a straightforward implementation of this idea. The more recent version FastSLAM 2.0, also described in this article, offers an improved proposal distribution that yields superior practical results; however, it is more difficult to implement than FastSLAM 1.0, and the improvement pays out only in somewhat extreme circumstances.

The article presented several results, in addition to stating and deriving the basic algorithm. Convergence of the FastSLAM 2.0 algorithm was proven for a restrictive family of linear-Gaussian SLAM problems. The theoretical results were complemented with extensive empirical evaluations using real world data. One of the experiments compared FastSLAM to the extended Kalman filter (EKF), using a sequence of problems with increasingly hard data association problems. While FastSLAM 1.0 performed equally well in all these problems, EKFs failed to generate consistent maps in an increasing number of problems. Among other things, we attribute this finding to FastSLAM's ability to sample over data associations. Further experiments characterized the superior performance of FastSLAM 2.0 over 1.0 in regimes with low sensor noise and high motion noise. Finally, the article provided a tree-based implementation that makes it possible to update the filter in time logarithmic in the map, which makes FastSLAM more efficient than most other SLAM algorithms that are capable of maintaining globally consistent maps.

The research presented here raises many open questions that warrant future research. While convergence has been established for a relatively simple class of problems, no formal results are presently available for more realistic SLAM problems. Furthermore, little is known regarding the convergence speed of FastSLAM, both in absolute terms and in comparison to the EKF approach.

Clearly, the idea of sampling over data associations is more general than the specific setting here, and it is highly related to previous techniques for tracking multiple objects [5, 60]. Maintaining multiple hypotheses is currently poorly explored in the SLAM field, despite an important early contribution to SLAM with known robot poses [12], and a recent approach to apply mixtures of Gaussians to the SLAM problem [20]. We conjecture that the sampling technique over data associations is not specific to FastSLAM, but can be applied to a wide range of SLAM algorithms.

FastSLAM, as presented in this paper, applies to feature-based maps only. Much of the recent research in the field has focused on developing feature-less, volumetric maps. Recently, Hähnel and colleagues developed a highly efficient implementation of FastSLAM [27] that uses raw laser range measurements to represent maps, instead of isolated landmarks. By doing so, the approach makes the data association problem much easier. Similar techniques have been reported in [21, 67], with similarly encouraging results. Results in [27] show excellent results when closing loops in such dense maps, previously postulated as one of the hardest problems in SLAM [9, 26]. A precursor to this work, which can be thought of as an instantiation of the same idea [65], has extended FastSLAM to a multi-robot SLAM technique. To date, this work is one of a handful of techniques capable of generating maps with teams of robots. Initial results in [55] illustrate high promise for FastSLAM in multi-robot SLAM problems. Murphy's paper applied a technique similar to FastSLAM to idealized versions of occupancy grid maps [52]. Finally, FastSLAM might also yield improved results in tracking moving features, a domain in which similar decompositions have recently been developed using somewhat different posterior representations [2, 47, 52].

Possibly the biggest limitation of FastSLAM is the fact that maintains dependencies in the estimates of feature locations only implicitly, through the diversity of its particle set. This disadvantage is also the source of FastSLAM's efficiency—a key advantage of FastSLAM over previous techniques. However, as the experiments in Section 10.5 suggests, in certain environments this can negatively affect the convergence speed when compared to the mathematically more cumbersome EKF. Since FastSLAM was invented, several variants of EKFs have been proposed that maintain some of these long-range dependencies without sacrificing computational efficiency by more than a constant factor [9, 70]. It should be straightforward to combine FastSLAM's particle-approach to data association with these new techniques for efficient Gaussian estimation.

Despite these open research topics, the algorithm FastSLAM has been shown unprecedented scalability to SLAM problems with hard data association problems and large number of features. Further, FastSLAM 1.0 tends to be easier to implement than most, if not all, published SLAM algorithms. We believe the insights and techniques presented in this paper transcend to a large number of existing SLAM techniques, and will ultimately deepen our understanding as to how to build detailed, accurate maps in situations with high degrees of ambiguity,

Acknowledgments

We are indebted to Kevin Murphy, Nando de Freitas, Michael Stevens and the members of CMU's Robot Learning Lab for many insightful discussions on this topic. This research is sponsored by by DARPA's MARS Program (contracts N66001-01-C-6018 and NBCH1020014) and the National Science Foundation (CAREER grant number IIS-9876136 and regular grant number IIS-9877033), all of which is gratefully acknowledged. The authors also gratefully acknowledge the Fannie and John Hertz Foundation for their support of Michael Montemerlo's graduate research.

References

[1] A. Araneda. tatistical inference in mapping and localization for a mobile robot. In J. M. Bernardo, M.J. Bayarri, J.O. Berger, A. P. Dawid, D. Heckerman, A.F.M. Smith, and M. West, editors, *Bayesian Statistics 7*. Oxford University Press, Oxford, UK, 2003.

[2] D. Avots, E. Lim, R. Thibaux, and S. Thrun. A probabilistic technique for simultaneous localization and door state estimation with mobile robots in dynamic environments. In *Proceedings of the Conference on Intelligent Robots and Systems (IROS)*, Lausanne, Switzerland, 2002.

[3] T. Bailey. *Mobile Robot Localisation and Mapping in Extensive Outdoor Environments*. PhD thesis, University of Sydney, Sydney, NSW, Australia, 2002.

[4] Y. Bar-Shalom and T. E. Fortmann. *Tracking and Data Association*. Academic Press, 1988.

[5] Y. Bar-Shalom and X.-R. Li. *Estimation and Tracking: principles, Techniques, and Software*. YBS, Danvers, MA, 1998.

[6] J.L. Bentley. Multidimensional divide and conquer. *Communications of the ACM*, 23(4):214–229, 1980.

[7] J. Borenstein, B. Everett, and L. Feng. *Navigating Mobile Robots: Systems and Techniques*. A. K. Peters, Ltd., Wellesley, MA, 1996.

[8] M. Bosse, J. Leonard, and S. Teller. Large-scale CML using a network of multiple local maps. In J. Leonard, J.D. Tardós, S. Thrun, and H. Choset, editors, *Workshop Notes of the ICRA Workshop on Concurrent Mapping and Localization for Autonomous Mobile Robots (W4)*, Washington, DC, 2002. ICRA Conference.

[9] M. Bosse, P. Newman, M. Soika, W. Feiten, J. Leonard, and S. Teller. An atlas framework for scalable mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2003.

[10] W. Burgard, A.B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. Experiences with an interactive museum tour-guide robot. *Artificial Intelligence*, 114(1-2):3–55, 1999.

[11] J.A. Castellanos, J.M.M. Montiel, J. Neira, and J.D. Tardós. The SPmap: A probabilistic framework for simultaneous localization and map building. *IEEE Transactions on Robotics and Automation*, 15(5):948–953, 1999.

[12] I.J. Cox and J.J. Leonard. Modeling a dynamic environment using a Bayesian multiple hypothesis approach. *Artificial Intelligence*, 66:311–344, 1994.

[13] I.J. Cox and G.T. Wilfong, editors. *Autonomous Robot Vehicles*. Springer Verlag, 1990.

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

[15] M. C. Deans and M. Hebert. Experimental comparison of techniques for localization and mapping using a bearing-only sensor. In *Proceedings of the International Symposium on Experimental Robotics (ISER)*, Sant'Angelo d'Ischia, Italy, 2002.

[16] F. Dellaert, S.M. Seitz, C. Thorpe, and S. Thrun. EM, MCMC, and chain flipping for structure from motion with unknown correspondence. *Machine Learning*, 50(1-2):45–71, 2003.

[17] G. Dissanayake, P. Newman, S. Clark, H.F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localisation and map building (SLAM) problem. *IEEE Transactions of Robotics and Automation*, 2001. In Press.

[18] A. Doucet, J.F.G. de Freitas, and N.J. Gordon, editors. *Sequential Monte Carlo Methods In Practice*. Springer Verlag, New York, 2001.

[19] A Doucet, N. de Freitas, K. Murphy, and S. Russell. Rao-Blackwellised particle filtering for dynamic Bayesian networks. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 176–183, Stanford, 2000.

[20] H. Durrant-Whyte, S. Majumder, S. Thrun, M. de Battista, and S. Scheding. A Bayesian algorithm for simultaneous localization and map building. In *Proceedings of the 10th International Symposium of Robotics Research (ISRR'01)*, Lorne, Australia, 2001.

[21] A. Eliazar and R. Parr. DP-SLAM: Fast, robust simultaneous localization and mapping without predetermined landmarks. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, Acapulco, Mexico, 2003. IJCAI.

[22] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24:381–395, 1981.

[23] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte carlo localization: Efficient position estimation for mobile robots. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, Orlando, FL, 1999. AAAI.

[24] Zoubin Ghahramani. Learning dynamic Bayesian networks. *Lecture Notes in Computer Science*, 1387, 1998.

[25] J. Guivant and E. Nebot. Optimization of the simultaneous localization and map building algorithm for real time implementation. *IEEE Transactions of Robotics and Automation*, May 2001. In press.

[26] J.-S. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. In *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, 2000.

[27] D. Hähnel, D. Fox, W. Burgard, and S. Thrun. A highly efficient FastSLAM algorithm for generating cyclic maps of large-scale environments from raw laser range measurements. In *Proceedings of the Conference on Intelligent Robots and Systems (IROS)*, 2003.

[28] M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. In *European Conference on Computer Vision*, pages 343–356, 1996.

[29] A.M. Jazwinsky. *Stochastic Processes and Filtering Theory*. Academic, New York, 1970.

[30] S. J. Julier and J. K. Uhlmann. A new extension of the Kalman filter to nonlinear systems. In *Proceedings of AeroSense: The 11th International Symposium on Aerospace/Defence Sensing, Simulation and Controls*, Orlando, FL, 1997.

[31] R. E. Kalman. A new approach to linear filtering and prediction problems. *Trans. ASME, Journal of Basic Engineering*, 82:35–45, 1960.

[32] K. Konolige and K. Chou. Markov localization using correlation. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, Stockholm, Sweden, 1999. IJCAI.

[33] D. Kortenkamp, R.P. Bonasso, and R. Murphy, editors. *AI-based Mobile Robots: Case studies of successful robot systems*, Cambridge, MA, 1998. MIT Press.

[34] J. Leonard, J.D. Tardós, S. Thrun, and H. Choset, editors. *Workshop Notes of the ICRA Workshop on Concurrent Mapping and Localization for Autonomous Mobile Robots (W4)*. ICRA Conference, Washington, DC, 2002.

[35] J.J. Leonard and H.J.S. Feder. A computationally efficient method for large-scale concurrent mapping and localization. In J. Hollerbach and D. Koditschek, editors, *Proceedings of the Ninth International Symposium on Robotics Research*, Salt Lake City, Utah, 1999.

[36] R. Li, F. Ma, F. Xu, L. Matthies, C. Olson, and Y. Xiong. Large scale mars mapping and rover localization using descent and rover imagery. In *Proceedings of the ISPRS 19th Congress, IAPRS Vol. XXXIII*, Amsterdam, 2000.

[37] J. Liu and R. Chen. Sequential monte carlo methods for dynamic systems. *Journal of the American Statistical Association*, 93:1032–1044, 1998.

[38] Y. Liu and S. Thrun. Results for outdoor-SLAM using sparse extended information filters. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2003.

[39] D. B. Lomet and B. Salzberg. The hb-tree: A multiattribute indexing method. *ACM Transactions on Database Systems*, 15(4):625–658, 1990.

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

[41] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equations of state calculations by fast computing machine. *Journal of Chemical Physics*, 21:1087–1091, 1953.

[42] T. Minka. *A family of algorithms for approximate Bayesian inference*. PhD thesis, MIT Media Lab, Cambridge, AM, 2001.

[43] M. Montemerlo. FastSLAM: A factored solution to the simultaneous localization and mapping problem with unknown data association. Technical Report CMU-RI-03-28, Carnegie Mellon University, Robotics Institute, Pittsburgh, PA, 2003.

[44] M. Montemerlo and S. Thrun. Simultaneous localization and mapping with unknown data association using FastSLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2003.

[45] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, Edmonton, Canada, 2002. AAAI.

[46] 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 Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, Acapulco, Mexico, 2003. IJCAI.

[47] M. Montemerlo, W. Whittaker, and S. Thrun. Conditional particle filters for simultaneous mobile robot localization and people-tracking. In *IEEE International Conference on Robotics and Automation (ICRA)*, Washington, DC, 2002. ICRA.

[48] A.W. Moore. Very fast EM-based mixture model clustering using multiresolution kd-trees. In *Advances in Neural Information Processing Systems (NIPS)*, Cambridge, MA, 1998. MIT Press.

[49] H. P. Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, 9(2):61–74, 1988.

[50] P. Moutarlier and R. Chatila. An experimental system for incremental environment modeling by an autonomous mobile robot. In *1st International Symposium on Experimental Robotics*, Montreal, June 1989.

[51] P. Moutarlier and R. Chatila. Stochastic multisensory data fusion for mobile robot location and environment modeling. In *5th Int. Symposium on Robotics Research*, Tokyo, 1989.

[52] K. Murphy. Bayesian map learning in dynamic environments. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, 1999.

[53] J. Neira and J.D. Tardós. Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on Robotics and Automation*, 17(6):890–897, 2001.

[54] P. Newman. *On the Structure and Solution of the Simultaneous Localisation and Map Building Problem*. PhD thesis, Australian Centre for Field Robotics, University of Sydney, Sydney, Australia, 2000.

[55] J. Nieto, J. Guivant, E. Nebot, and S. Thrun. Real time data association for FastSLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2003.

[56] M.A. Paskin. Thin junction tree filters for simultaneous localization and mapping. Technical Report UCB/CSD-02-1198, University of California, Berkeley, CA, 2002.

[57] M.A. Paskin. Thin junction tree filters for simultaneous localization and mapping. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, Acapulco, Mexico, 2003. IJCAI.

[58] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers, San Mateo, CA, 1988.

[59] O. Procopiuc, P.K. Agarwal, L. Arge, and J.S. Vitter. Bkd-tree: A dynamic scalable kd-tree. Submitted for publication, 2002.

[60] D.B. Reid. An algorithm for tracking multiple targets. *IEEE Transactions on Aerospace and Electronic Systems*, AC-24:843–854, 1979.

[61] D.B. Rubin. Using the SIR algorithm to simulate posterior distributions. In M.H. Bernardo, K.M. an DeGroot, D.V. Lindley, and A.F.M. Smith, editors, *Bayesian Statistics 3*. Oxford University Press, Oxford, UK, 1988.

[62] S. Scheding, E.M. Nebot, M. Stevens, and H.F. Durrant-Whyte. Experiments in autonomous underground guidance. In R. Harrigan and M. Jamshidi, editors, *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1898–1903, Albuquerque NM, 1997. ICRA.

[63] H Shatkay and L. Kaelbling. Learning topological maps with weak local odometric information. In *Proceedings of IJCAI-97*. IJCAI, Inc., 1997.

[64] R.C. Smith and P. Cheeseman. On the representation and estimation of spatial uncertainty. *International Journal of Robotics Research*, 5(4):56–68, 1986.

[65] S. Thrun. A probabilistic online mapping algorithm for teams of mobile robots. *International Journal of Robotics Research*, 20(5):335–363, 2001.

[66] S. Thrun. Robotic mapping: A survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002.

[67] S. Thrun, W. Burgard, and D. Fox. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, San Francisco, CA, 2000. IEEE.

[68] 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).

[69] S. Thrun, D. Hähnel, D. Ferguson, M. Montemerlo, R. Triebel, W. Burgard, C. Baker, Z. Omohundro, S. Thayer, and W. Whittaker. A system for volumetric robotic mapping of abandoned mines. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2003.

[70] S. Thrun, D. Koller, Z. Ghahramani, H. Durrant-Whyte, and A.Y. Ng. Simultaneous mapping and localization with sparse extended information filters. In J.-D. Boissonnat, J. Burdick, K. Goldberg, and S. Hutchinson, editors, *Proceedings of the Fifth International Workshop on Algorithmic Foundations of Robotics*, Nice, France, 2002.

[71] J. Uhlmann, M. Lanzagorta, and S. Julier. The NASA mars rover: A testbed for evaluating applications of covariance intersection. In *Proceedings of the SPIE 13th Annual Symposium in Aerospace/Defence Sensing, Simulation and Controls*, 1999.

[72] R. van der Merwe, N. de Freitas, A. Doucet, and E. Wan. The unscented particle filter. In *Advances in Neural Information Processing Systems 13*, 2001.

[73] S. Williams, G. Dissanayake, and H.F. Durrant-Whyte. Towards terrain-aided navigation for underwater robotics. *Advanced Robotics*, 15(5), 2001.

Appendix A: Derivation of the EKF Update Equations

The derivation of measurement update equations (53) through (55) will be essential for our proof of convergence. We begin by noting that the exponent of (52) is a function quadratic in the target variable θ_{n_t}. Denoting the (negative) exponent by J_t, we obtain the following derivatives:

$$\begin{aligned}
J_t &= \tfrac{1}{2}(z_t - \hat{z}_t^{[m]} - G_\theta(\theta_{n_t} - \mu_{n_t,t-1}^{[m]}))^T R_t^{-1}(z_t - \hat{z}_t^{[m]} - G_\theta(\theta_{n_t} - \mu_{n_t,t-1}^{[m]})) \\
&\quad + \tfrac{1}{2}(\theta_{n_t} - \mu_{n_t,t-1}^{[m]})^T \Sigma_{n_t,t-1}^{[m]-1}(\theta_{n_t} - \mu_{n_t,t-1}^{[m]}) \\
\frac{\partial J_t}{\partial \theta_{n_t}} &= -G_\theta^T R_t^{-1}(z_t - \hat{z}_t^{[m]} - G_\theta(\theta_{n_t} - \mu_{n_t,t-1}^{[m]})) + \Sigma_{n_t,t-1}^{[m]-1}(\theta_{n_t} - \mu_{n_t,t-1}^{[m]}) \\
\frac{\partial^2 J_t}{\partial \theta_{n_t}^2} &= G_\theta^T R_t^{-1} G_\theta + \Sigma_{n_t,t-1}^{[m]-1} \qquad (82)
\end{aligned}$$

The new covariance $\Sigma_{n_t,t}^{[m]}$ given by inverse of second derivative

$$\Sigma_{n_t,t}^{[m]} = \left(G_\theta^T R_t^{-1} G_\theta + \Sigma_{n_t,t-1}^{[m]-1}\right)^{-1} \qquad (83)$$

As we will show further below, this expression is equivalent to (55). The new mean $\mu_{n_t,t}^{[m]}$ is the maximum of the quadratic function, which can be obtained by setting the first derivative to zero:

$$\left.\frac{\partial J_t}{\partial \theta_{n_t}}\right|_{\theta_{n_t}=\mu_{n_t,t}^{[m]}} = 0 \qquad (84)$$

The maximum is thus attained at

$$G_\theta^T R_t^{-1}\left(z_t - \hat{z}_t^{[m]} - G_\theta(\mu_{n_t,t}^{[m]} - \mu_{n_t,t-1}^{[m]})\right) = \Sigma_{n_t,t-1}^{[m]-1}(\mu_{n_t,t}^{[m]} - \mu_{n_t,t-1}^{[m]}) \qquad (85)$$

Reordering the terms gives us

$$G_\theta^T R_t^{-1}(z_t - \hat{z}_t^{[m]}) = \underbrace{(\Sigma_{n_t,t-1}^{[m]-1} + G_\theta^T R_t^{-1} G_\theta)}_{\Sigma_{n_t,t}^{[m]-1}}(\mu_{n_t,t}^{[m]} - \mu_{n_t,t-1}^{[m]}) \qquad (86)$$

and hence

$$\mu_{n_t,t-1}^{[m]} = \mu_{n_t,t-1}^{[m]} + \underbrace{\Sigma_{n_t,t}^{[m]} G_\theta^T R_t^{-1}}_{K_t^{[m]}}(z_t - \hat{z}_t^{[m]}) \qquad (87)$$

To show that this is equivalent to (54), all we have to show is that the $K_t^{[m]} = \Sigma_{n_t,t}^{[m]} G_\theta^T R_t^{-1}$ is equivalent to the expression in (53). This follows from a straightforward set of algebraic transformations:

$$\begin{aligned}
K_t^{[m]} &= \Sigma_{n_t,t}^{[m]} G_\theta^T R_t^{-1} \\
&= \Sigma_{n_t,t}^{[m]} G_\theta^T R_t^{-1}\left(G_\theta \Sigma_{n_t,t-1}^{[m]} G_\theta^T + R_t\right)\left(G_\theta \Sigma_{n_t,t-1}^{[m]} G_\theta^T + R_t\right)^{-1} \\
&= \Sigma_{n_t,t}^{[m]}\left(G_\theta^T R_t^{-1} G_\theta \Sigma_{n_t,t-1}^{[m]} G_\theta^T + G_\theta^T\right)\left(G_\theta \Sigma_{n_t,t-1}^{[m]} G_\theta^T + R_t\right)^{-1} \\
&= \Sigma_{n_t,t}^{[m]}\left(G_\theta^T R_t^{-1} G_\theta + \Sigma_{n_t,t}^{[m]-1}\right)\Sigma_{n_t,t-1}^{[m]} G_\theta^T\left(G_\theta \Sigma_{n_t,t-1}^{[m]} G_\theta^T + R_t\right)^{-1} \\
&= \Sigma_{n_t,t}^{[m]} \Sigma_{n_t,t}^{[m]-1} \Sigma_{n_t,t-1}^{[m]} G_\theta^T\left(G_\theta \Sigma_{n_t,t-1}^{[m]} G_\theta^T + R_t\right)^{-1} \\
&= \Sigma_{n_t,t-1}^{[m]} G_\theta^T\left(G_\theta \Sigma_{n_t,t-1}^{[m]} G_\theta^T + R_t\right)^{-1} \qquad (88)
\end{aligned}$$

This establishes the correctness of (54). To show (55), we restate the covariance (83) in its incremental form. In particular, we apply the *matrix inversion lemma*

$$(C^{-1} + BAB^T)^{-1} \;\; = \;\; C - CB\left(A^{-1} + B^T CB\right)^{-1} B^T C \tag{89}$$

to the expression (83)

$$\begin{aligned}
\Sigma_{n_t,t}^{[m]} \;\; &= \;\; \left(G_\theta^T R_t^{-1} G_\theta + \Sigma_{n_t,t-1}^{[m]-1}\right)^{-1} \\
&= \;\; \Sigma_{n_t,t-1}^{[m]} - \Sigma_{n_t,t-1}^{[m]} G_\theta^T \left(G_\theta \Sigma_{n_t,t-1}^{[m]} G_\theta^T + R_t\right)^{-1} G_\theta \Sigma_{n_t,t-1}^{[m]} \\
&= \;\; \left[I - \Sigma_{n_t,t-1}^{[m]} G_\theta^T \left(G_\theta \Sigma_{n_t,t-1}^{[m]} G_\theta^T + R_t\right)^{-1} G_\theta\right] \Sigma_{n_t,t-1}^{[m]} \\
&= \;\; \left(I - K_t^{[m]} G_\theta\right) \Sigma_{n_t,t-1}^{[m]}
\end{aligned} \tag{90}$$

This shows the correctness of (55).

Appendix B: Convergence

The proof of convergence is carried out through a series of lemmas. For that, it will be convenient to formulate elements of the FastSLAM 2.0 algorithm for LG-SLAM, exploiting the specific definitions of the functions g and h in Equations (77) and (78). In particular, we have

$$\hat{s}_t^{[m]} \;\; = \;\; h(s_{t-1}^{[m]}, u_t) \;\; = \;\; s_{t-1}^{[m]} + u_t \tag{91}$$

$$\hat{z}_t^{[m]} \;\; = \;\; g(\mu_{n_t,t-1}^{[m]}, \hat{s}_t^{[m]}) \;\; = \;\; \mu_{n_t,t-1}^{[m]} - \hat{s}_t^{[m]} \;\; = \;\; \mu_{n_t,t-1}^{[m]} - s_{t-1}^{[m]} - u_t \tag{92}$$

$$G_\theta \;\; = \;\; \nabla_{\theta_{n_t}} g(\theta_{n_t}, s_t)\Big|_{s_t=\hat{s}_t^{[m]};\theta_{n_t}=\mu_{n_t,t-1}^{[m]}} \;\; = \;\; I \tag{93}$$

$$G_s \;\; = \;\; \nabla_{s_t} g(\theta_{n_t}, s_t)\big|_{s_t=\hat{s}_t^{[m]};\theta_{n_t}=\mu_{n_t,t-1}^{[m]}} \;\; = \;\; -I \tag{94}$$

$$Q_t^{[m]} \;\; = \;\; R_t + G_\theta \Sigma_{n_t,t-1}^{[m]} G_\theta^T \;\; = \;\; R_t + \Sigma_{n_t,t-1}^{[m]} \tag{95}$$

These equations follow directly from the more general definitions in the algorithm FastSLAM 2.0 (Equations (36) through (39) and (42)). FastSLAM 2.0's sampling rule for the t-th pose, stated in its general form in (40) and (41), is now conveniently written as follows:

$$\begin{aligned}
\mu_{s_t}^{[m]} \;\; &= \;\; \Sigma_{s_t}^{[m]} G_s^T Q_t^{[m]-1}(z_t - \hat{z}_t^{[m]}) + \hat{s}_t^{[m]} \\
&= \;\; -\Sigma_{s_t}^{[m]}(R_t + \Sigma_{n_t,t-1}^{[m]})^{-1}(z_t - \mu_{n_t,t-1}^{[m]} + s_{t-1}^{[m]} + u_t) + s_{t-1}^{[m]} + u_t
\end{aligned} \tag{96}$$

$$\Sigma_{s_t}^{[m]} \;\; = \;\; \left[G_s^T Q_t^{[m]-1} G_s + P_t^{-1}\right]^{-1} \;\; = \;\; \left[(R_t + \Sigma_{n_t,t-1}^{[m]})^{-1} + P_t^{-1}\right]^{-1} \tag{97}$$

Similarly, the mean update for the observed feature, which in FastSLAM 2.0 is attained via Equations (53) and (54), can be written as follows for LG-SLAM:

$$\mu_{n_t,t}^{[m]} \;\; = \;\; \mu_{n_t,t-1}^{[m]} + \Sigma_{n_t,t-1}^{[m]}(R_t + \Sigma_{n_t,t-1}^{[m]})^{-1}(z_t - \mu_{n_t,t-1}^{[m]} + s_{t-1}^{[m]} + u_t) \tag{98}$$

For our proof of the Theorem, it will be useful to introduce error variables for the estimates of the robot pose, and the feature locations, respectively.

$$\alpha_t^{[m]} \;\; = \;\; s_t^{[m]} - s_t \tag{99}$$

$$\beta_{n,t}^{[m]} \;\; = \;\; \mu_{n,t}^{[m]} - \theta_n \tag{100}$$

There variables measure the absolute error of a particle's estimates of the robot's pose and the features in the map. We will refer to the known feature as *anchoring feature*. The first result characterizes the effect of map errors β on the pose the pose error α:

Lemma 1. *If the error $\beta_{n_t,t}^{[m]}$ of the observed feature z_t at time t is smaller in magnitude than the robot pose error $\alpha_t^{[m]}$, $\alpha_t^{[m]}$ shrinks in expectation as a result of this measurement. Conversely, if $\beta_{n_t,t}^{[m]}$ is larger than the pose error $\alpha_t^{[m]}$, the latter may increase, but in expectation will not exceed $\beta_{n_t,t}^{[m]}$.*

Proof of Lemma 1. The expected error of the robot pose sample at time t is given by

$$E[\alpha_t^{[m]}] \;=\; E[s_t^{[m]} - s_t] \;=\; E[s_t^{[m]}] - E[s_t] \tag{101}$$

The first term obtained via the sampling distribution (96), and the second term is obtained from linear motion model (78):

$$\begin{aligned} E[\alpha_t^{[m]}] &= E\left[-\Sigma_{s_t}^{[m]}(R_t + \Sigma_{n_t,t-1}^{[m]})^{-1}(z_t - \mu_{n_t,t-1}^{[m]} + s_{t-1}^{[m]} + u_t) + s_{t-1}^{[m]} + u_t\right] - E[u_t + s_{t-1}] \\ &= -\Sigma_{s_t}^{[m]}(R_t + \Sigma_{n_t,t-1}^{[m]})^{-1}(E[z_t] - \mu_{n_t,t-1}^{[m]} + s_{t-1}^{[m]} + u_t) + \underbrace{s_{t-1}^{[m]} - s_{t-1}}_{\alpha_{t-1}^{[m]}} \end{aligned} \tag{102}$$

The last transformation exploited the linearity of the expectation. We note that in LG-SLAM (77) and (78), the expectation $E[z_t] = \theta_{n_t} - E[s_t] = \theta_{n_t} - u_t - s_{t-1}$. With that, the expression in the brackets becomes

$$\begin{aligned} E[z_t] - \mu_{n_t,t-1}^{[m]} + s_{t-1}^{[m]} + u_t &= \theta_{n_t} - u_t - s_{t-1} - \mu_{n_t,t-1}^{[m]} + s_{t-1}^{[m]} + u_t \\ &= s_{t-1}^{[m]} - s_{t-1} + \theta_{n_t} - \mu_{n_t,t-1}^{[m]} \\ &= \alpha_{t-1}^{[m]} - \beta_{n_t,t-1}^{[m]} \end{aligned} \tag{103}$$

Plugging this back into (102) and subsequently substituting $\Sigma_{s_t}^{[m]}$ according to (97) gives us:

$$\begin{aligned} E[\alpha_t^{[m]}] &= \alpha_{t-1}^{[m]} + \Sigma_{s_t}^{[m]}(R_t + \Sigma_{n_t,t-1}^{[m]})^{-1}(\beta_{n_t,t-1}^{[m]} - \alpha_{t-1}^{[m]}) \\ &= \alpha_{t-1}^{[m]} + \left[(R_t + \Sigma_{n_t,t-1}^{[m]})^{-1} + P_t^{-1}\right]^{-1}(R_t + \Sigma_{n_t,t-1}^{[m]})^{-1}(\beta_{n_t,t-1}^{[m]} - \alpha_{t-1}^{[m]}) \\ &= \alpha_{t-1}^{[m]} + \left[I + (R_t + \Sigma_{n_t,t-1}^{[m]})P_t^{-1}\right]^{-1}(\beta_{n_t,t-1}^{[m]} - \alpha_{t-1}^{[m]}) \end{aligned} \tag{104}$$

Since R_t, $\Sigma_{n_t,t-1}^{[m]}$, and P_t^{-1} are all positive semidefinite, the inverse of $I + (R_t + \Sigma_{n_t,t-1}^{[m]})P_t^{-1}$ is a contraction matrix. This observation effectively proves Lemma 1. In particular, the expected pose error $\alpha_{t-1}^{[m]}$ shrinks in magnitude if $\beta_{n_t,t}^{[m]}$ is smaller in magnitude than $\alpha_{t-1}^{[m]}$. Conversely, if $\alpha_{t-1}^{[m]}$ is smaller in magnitude than $\beta_{n_t,t}^{[m]}$, Equation (104) then suggests that $\alpha_{t-1}^{[m]}$ will increase in expectation, but by a value that is proportional to this difference. This ensures that $\alpha_{t-1}^{[m]}$ will not exceed the error $\beta_{n_t,t}^{[m]}$ in expectation. *qed.*

Of particular interest is the result of observing the anchoring feature. Without loss of generality, we assume that this feature is θ_1.

Lemma 2. *If the robot observes the anchoring feature, its pose error will shrink in expectation.*

Proof of Lemma 2. For the anchoring feature θ_1, we can exploit the fact that $\Sigma_{1,t}^{[m]} = \beta_{1,t}^{[m]} = 0$. The lemma now follows directly from Equation (104),

$$\begin{aligned} E[\alpha_t^{[m]}] &= \alpha_{t-1}^{[m]} + \left[I + (R_t + 0)P_t^{-1}\right]^{-1}(0 - \alpha_{t-1}^{[m]}) \\ &= \alpha_{t-1}^{[m]} - \left[I + R_t P_t^{-1}\right]^{-1}\alpha_{t-1}^{[m]} \end{aligned} \tag{105}$$

Thus, whenever the robot sees its anchoring feature its position error $\alpha_{t-1}^{[m]}$ is expected to shrink. The only exception arises when the error is already zero, in which case it remains zero in expectation. *qed.*

Finally, a lemma similar to Lemma 1 can be stated on the effect of pose errors α on map errors β:

Lemma 3. *If the pose error $\alpha_{t-1}^{[m]}$ is smaller than the error $\beta_{n_t,t}^{[m]}$ of the observed feature z_t in magnitude, observing z_t shrinks the feature error $\beta_{n_t,t}^{[m]}$ in expectation. Conversely, if $\alpha_{t-1}^{[m]}$ is larger than the feature error $\beta_{n_t,t}^{[m]}$, the latter may increase, but in expectation will not exceed $\alpha_{t-1}^{[m]}$.*

Proof of Lemma 3. This proof is analogous to that of Lemma 1. From (98) it follows that the expected feature error after updating is:

$$\begin{aligned} E[\beta_{n,t}^{[m]}] &= E[\mu_{n,t}^{[m]} - \theta_n] \quad = \quad E[\mu_{n,t}^{[m]}] - \theta_n \\ &= E\left[\mu_{n_t,t-1}^{[m]} + \Sigma_{n_t,t-1}^{[m]}(R_t + \Sigma_{n_t,t-1}^{[m]})^{-1}(z_t - \mu_{n_t,t-1}^{[m]} + s_{t-1}^{[m]} + u_t)\right] - \theta_n \\ &= \mu_{n_t,t-1}^{[m]} + \Sigma_{n_t,t-1}^{[m]}(R_t + \Sigma_{n_t,t-1}^{[m]})^{-1}(E[z_t] - \mu_{n_t,t-1}^{[m]} + s_{t-1}^{[m]} + u_t) - \theta_n \end{aligned} \tag{106}$$

Equation (103) enables us to rewrite this as follows:

$$\begin{aligned} E[\beta_{n,t}^{[m]}] &= \mu_{n_t,t-1}^{[m]} + \Sigma_{n_t,t-1}^{[m]}(R_t + \Sigma_{n_t,t-1}^{[m]})^{-1}(\alpha_{t-1}^{[m]} - \beta_{n_t,t-1}^{[m]}) - \theta_n \\ &= \beta_{n,t-1}^{[m]} + \Sigma_{n_t,t-1}^{[m]}(R_t + \Sigma_{n_t,t-1}^{[m]})^{-1}(\alpha_{t-1}^{[m]} - \beta_{n_t,t-1}^{[m]}) \\ &= \beta_{n,t-1}^{[m]} + (\Sigma_{n_t,t-1}^{[m]-1}R_t)^{-1}(\alpha_{t-1}^{[m]} - \beta_{n_t,t-1}^{[m]}) \end{aligned} \tag{107}$$

As in the proof of Lemma 1, $\Sigma_{n_t,t-1}^{[m]-1}$ and R_t are both positive semidefinite, the inverse of $\Sigma_{n_t,t-1}^{[m]-1}R_t$ is a contraction matrix, which immediately proves the lemma. *qed.*

The Theorem now follows from our three lemmas and the specific equations regarding the evolution of expected errors over time.

Proof of Theorem. Let $\hat{\beta}_t^{[m]}$ denote feature error that is largest in magnitude among all feature errors at time t.

$$\hat{\beta}_t^{[m]} = \underset{\beta_{n,t}^{[m]}}{\mathrm{argmax}}\ |\beta_{n,t}^{[m]}| \tag{108}$$

Lemma 3 suggests that this error may increase in expectation, but only if the absolute robot pose error $\alpha_{t-1}^{[m]}$ exceeds this error in magnitude. However, in expectation this will only be the case for a limited number of iterations. In particular, Lemma 1 guarantees that $\alpha_{t-1}^{[m]}$ may only shrink in expectation. Furthermore, Lemma 2 states that every time the anchoring feature is observed, this error will shrink by a finite amount, regardless of the magnitude of $\hat{\beta}_t^{[m]}$. Hence, $\alpha_{t-1}^{[m]}$ will ultimately become smaller in magnitude (and in expectation) than the largest feature error. Once this has happened, Lemma 3 states that the latter will shrink in expectation every time the feature is observed whose error is largest. It is now easy to see that both $\hat{\beta}_t^{[m]}$ and $\alpha_{t-1}^{[m]}$ converge to zero: Observing the anchoring feature induces a finite reduction as stated in Equation (105). To increase $\alpha_{t-1}^{[m]}$ to its old value in expectation, the total feature error must shrink in expectation, according to Equation (107). This leads to an eternal shrinkage of the total feature error down to zero. Since this error is an upper bound for the expected pose error (see Lemma 1), we also have convergence in expectation for the robot pose error. *qed.*

Theorem 1 trivially implies following corollary, which characterizes the convergence for more than one particle.

Corollary 1. *FastSLAM 2.0 converges in expectation for LG-SLAM with if all features are observed infinitely often and if the location of one feature is known in advance.*