FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem

Michael Montemerlo and **Sebastian Thrun**
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
mmde@cs.cmu.edu, thrun@cs.cmu.edu

Daphne Koller and **Ben Wegbreit**
Computer Science Department
Stanford University
Stanford, CA 94305-9010
koller@cs.stanford.edu, ben@wegbreit.com

Abstract

The ability to simultaneously localize a robot and accurately map its surroundings is considered by many to be a key prerequisite of truly autonomous robots. However, few approaches to this problem scale up to handle the very large number of landmarks present in real environments. Kalman filter-based algorithms, for example, require time quadratic in the number of landmarks to incorporate each sensor observation. This paper presents FastSLAM, an algorithm that recursively estimates the full posterior distribution over robot pose and landmark locations, yet scales logarithmically with the number of landmarks in the map. This algorithm is based on an exact factorization of the posterior into a product of conditional landmark distributions and a distribution over robot paths. The algorithm has been run successfully on as many as 50,000 landmarks, environments far beyond the reach of previous approaches. Experimental results demonstrate the advantages and limitations of the FastSLAM algorithm on both simulated and real-world data.

Introduction

The problem of *simultaneous localization and mapping*, also known as SLAM, has attracted immense attention in the mobile robotics literature. SLAM addresses the problem of building a map of an environment from a sequence of landmark measurements obtained from a moving robot. Since robot motion is subject to error, the mapping problem necessarily induces a robot localization problem—hence the name SLAM. The ability to simultaneously localize a robot and accurately map its environment is considered by many to be a key prerequisite of truly autonomous robots [3, 7, 16].

The dominant approach to the SLAM problem was introduced in a seminal paper by Smith, Self, and Cheeseman [15]. This paper proposed the use of the extended Kalman filter (EKF) for incrementally estimating the posterior distribution over robot pose along with the positions of the landmarks. In the last decade, this approach has found widespread acceptance in field robotics, as a recent tutorial paper [2] documents. Recent research has focused on scaling this approach to larger environments with more than a few hundred landmarks [6, 8, 9] and to algorithms for handling data association problems [17].

A key limitation of EKF-based approaches is their computational complexity. Sensor updates require time quadratic in the number of landmarks K to compute. This complexity stems from the fact that the covariance matrix maintained by the Kalman filters has $O(K^2)$ elements, all of which must be updated even if just a single landmark is observed. The quadratic complexity limits the number of landmarks that can be handled by this approach to only a few hundred—whereas natural environment models frequently contain millions of features. This shortcoming has long been recognized by the research community [6, 8, 14].

In this paper we approach the SLAM problem from a Bayesian point of view. Figure 1 illustrates a generative probabilistic model (dynamic Bayes network) that underlies the rich corpus of SLAM literature. In particular, the robot poses, denoted $s_1, s_2, \ldots, s_t$, evolve over time as a function of the robot controls, denoted $u_1, \ldots, u_t$. Each of the landmark measurements, denoted $z_1, \ldots, z_t$, is a function of the position θ_k of the landmark measured and of the robot pose at the time the measurement was taken. From this diagram it is evident that the SLAM problem exhibits important conditional independences. In particular, knowledge of the robot's path $s_1, s_2, \ldots, s_t$ renders the individual landmark measurements *independent*. So for example, if an oracle provided us with the exact path of the robot, the problem of determining the landmark locations could be decoupled into K independent estimation problems, one for each landmark. This observation was made previously by Murphy [12], who developed an efficient particle filtering algorithm for learning grid maps.

Based on this observation, this paper describes an efficient SLAM algorithm called *FastSLAM*. FastSLAM decomposes the SLAM problem into a robot localization problem, and a collection of landmark estimation problems that are conditioned on the robot pose estimate. As remarked in [12], this factored representation is exact, due to the natural conditional independences in the SLAM problem. FastSLAM uses a modified particle filter for estimating the posterior over robot paths. Each particle possesses K Kalman filters that estimate the K landmark locations conditioned on the path estimate. The resulting algorithm is an instance of the Rao-Blackwellized particle filter [5, 13]. A naive im-

Copyright © 2002, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

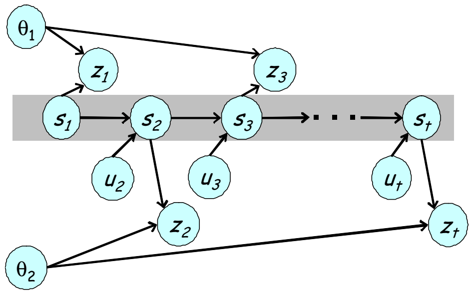

Figure 1: The SLAM problem: The robot moves from pose s_1 through a sequence of controls, $u_1, u_2, \ldots, u_t$. As it moves, it observes nearby landmarks. At time $t = 1$, it observes landmark θ_1 out of two landmarks, $\{\theta_1, \theta_2\}$. The measurement is denoted z_1 (range and bearing). At 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 and the measurements z. The gray shading illustrates a conditional independence relation.

plementation of this idea leads to an algorithm that requires $O(MK)$ time, where M is the number of particles in the particle filter and K is the number of landmarks. We develop a tree-based data structure that reduces the running time of FastSLAM to $O(M \log K)$, making it significantly faster than existing EKF-based SLAM algorithms. We also extend the FastSLAM algorithm to situations with unknown data association and unknown number of landmarks, showing that our approach can be extended to the full range of SLAM problems discussed in the literature.

Experimental results using a physical robot and a robot simulator illustrate that the FastSLAM algorithm can handle orders of magnitude more landmarks than present day approaches. We also find that in certain situations, an increased number of landmarks K leads to a mild *reduction* of the number of particles M needed to generate accurate maps—whereas in others the number of particles required for accurate mapping may be prohibitively large.

SLAM Problem Definition

The SLAM problem, as defined in the rich body of literature on SLAM, is best described as a probabilistic Markov chain. The robot's pose at time t will be denoted s_t. For robots operating in the plane—which is the case in all of our experiments—poses are comprised of a robot's x-y coordinate in the plane and its heading direction.

Poses evolve according to a probabilistic law, often referred to as the *motion model*:

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

Thus, s_t is a probabilistic function of the robot control u_t and the previous pose s_{t-1}. In mobile robotics, the motion model is usually a time-invariant probabilistic generalization of robot kinematics [1].

The robot's environment possesses K immobile landmarks. Each landmark is characterized by its location in space, denoted θ_k for $k = 1, \ldots, K$. Without loss of generality, we will think of landmarks as points in the plane, so that locations are specified by two numerical values.

To map its environment, the robot can sense landmarks. For example, it may be able to measure range and bearing to a landmark, relative to its local coordinate frame. The measurement at time t will be denoted z_t. While robots can often sense more than one landmark at a time, we follow commonplace notation by assuming that sensor measurements correspond to exactly one landmark [2]. This convention is adopted solely for mathematical convenience. It poses no restriction, as multiple landmark sightings at a single time step can be processed sequentially.

Sensor measurements are governed by a probabilistic law, often referred to as the *measurement model*:

$$p(z_t \mid s_t, \theta, n_t) \tag{2}$$

Here $\theta = \{\theta_1, \ldots, \theta_k\}$ is the set of all landmarks, and $n_t \in \{1, \ldots, K\}$ is the index of the landmark perceived at time t. For example, in Figure 1, we have $n_1 = 1, n_2 = 2$, and $n_3 = 1$, since the robot first observes landmark θ_1, then landmark θ_2, and finally landmark θ_1 for a second time. Many measurement models in the literature assume that the robot can measure range and bearing to landmarks, confounded by measurement noise. The variable n_t is often referred to as *correspondence*. Most theoretical work in the literature assumes knowledge of the correspondence or, put differently, that landmarks are uniquely identifiable. Practical implementations use maximum likelihood estimators for estimating the correspondence on-the-fly, which work well if landmarks are spaced sufficiently far apart. In large parts of this paper we will simply assume that landmarks are identifiable, but we will also discuss an extension that estimates the correspondences from data.

We are now ready to formulate the SLAM problem. Most generally, SLAM is the problem of determining the location of all landmarks θ and robot poses s_t from measurements $z^t = z_1, \ldots, z_t$ and controls $u^t = u_1, \ldots, u_t$. In probabilistic terms, this is expressed by the posterior $p(s^t, \theta \mid z^t, u^t)$, where we use the superscript ^{t} to refer to a set of variables from time 1 to time t. If the correspondences are known, the SLAM problem is simpler:

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

As discussed in the introduction, all individual landmark estimation problems are independent if one knew the robot's path s^t and the correspondence variables n^t. This conditional independence is the basis of the FastSLAM algorithm described in the next section.

FastSLAM with Known Correspondences

We begin our consideration with the important case where the correspondences $n^t = n_1, \ldots, n_t$ are known, and so is the number of landmarks K observed thus far.

Factored Representation

The conditional independence property of the SLAM problem implies that the posterior (3) can be factored as follows:

$$\begin{aligned} & p(s^t, \theta \mid z^t, u^t, n^t) \\ &= \; p(s^t \mid z^t, u^t, n^t) \prod_k p(\theta_k \mid s^t, z^t, u^t, n^t) \end{aligned} \tag{4}$$

Put verbally, the problem can be decomposed into $K+1$ estimation problems, one problem of estimating a posterior over robot paths s^t, and K problems of estimating the locations of the K landmarks conditioned on the path estimate. This factorization is exact and always applicable in the SLAM problem, as previously argued in [12].

The FastSLAM algorithm implements the path estimator $p(s^t \mid z^t, u^t, n^t)$ using a modified particle filter [4]. As we argue further below, this filter can sample efficiently from this space, providing a good approximation of the posterior even under non-linear motion kinematics. The landmark pose estimators $p(\theta_k \mid s^t, z^t, u^t, n^t)$ are realized by Kalman filters, using separate filters for different landmarks. Because the landmark estimates are conditioned on the path estimate, each particle in the particle filter has its own, local landmark estimates. Thus, for M particles and K landmarks, there will be a total of KM Kalman filters, each of dimension 2 (for the two landmark coordinates). This representation will now be discussed in detail.

Particle Filter Path Estimation

FastSLAM employs a particle filter for estimating the path posterior $p(s^t \mid z^t, u^t, n^t)$ in (4), using a filter that is similar (but not identical) to the *Monte Carlo localization (MCL)* algorithm [1]. MCL is an application of particle filter to the problem of robot pose estimation (localization). At each point in time, both algorithms maintain a set of particles representing the posterior $p(s^t \mid z^t, u^t, n^t)$, denoted S_t. Each particle $s^{t,[m]} \in S_t$ represents a "guess" of the robot's path:

$$S_t = \{s^{t,[m]}\}_m = \{s_1^{[m]}, s_2^{[m]}, \ldots, s_t^{[m]}\}_m \qquad (5)$$

We use the superscript notation $^{[m]}$ to refer to the m-th particle in the set.

The particle set S_t is calculated incrementally, from the set S_{t-1} at time $t-1$, a robot control u_t, and a measurement z_t. First, each particle $s^{t,[m]}$ in S_{t-1} is used to generate a probabilistic guess of the robot's pose at time t:

$$s_t^{[m]} \sim p(s_t \mid u_t, s_{t-1}^{[m]}), \qquad (6)$$

obtained by sampling from the probabilistic motion model. This estimate is then added to a temporary set of particles, along with the path $s^{t-1,[m]}$. Under the assumption that the set of 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), the new particle is distributed according to $p(s^t \mid z^{t-1}, u^t, n^{t-1})$. This distribution is commonly referred to as the *proposal distribution* of particle filtering.

After generating M particles in this way, the new set S_t is obtained by sampling from the temporary particle set. Each particle $s^{t,[m]}$ is drawn (with replacement) with a probability proportional to a so-called *importance factor* $w_t^{[m]}$, which is calculated as follows [10]:

$$w_t^{[m]} = \frac{\text{target distribution}}{\text{proposal distribution}} = \frac{p(s^{t,[m]} \mid z^t, u^t, n^t)}{p(s^{t,[m]} \mid z^{t-1}, u^t, n^{t-1})} \qquad (7)$$

The exact calculation of (7) will be discussed further below. The resulting sample set S_t is distributed according to an approximation to the desired pose posterior $p(s^t \mid z^t, u^t, n^t)$, an approximation which is correct as the number of particles M goes to infinity. We also notice that only the most recent robot pose estimate $s_{t-1}^{[m]}$ is used when generating the particle set S_t. This will allows us to silently "forget" all other pose estimates, rendering the size of each particle independent of the time index t.

Landmark Location Estimation

FastSLAM represents the conditional landmark estimates $p(\theta_k \mid s^t, z^t, u^t, n^t)$ in (4) by Kalman filters. Since this estimate is conditioned on the robot pose, the Kalman filters are attached to individual pose particles in S_t. More specifically, the full posterior over paths and landmark positions in the FastSLAM algorithm is represented by the sample set

$$\mathcal{S}_t = \{s^{t,[m]}, \mu_1^{[m]}, \Sigma_1^{[m]}, \ldots, \mu_K^{[m]}, \Sigma_K^{[m]}\}_m \qquad (8)$$

Here $\mu_k^{[m]}$ and $\Sigma_k^{[m]}$ are mean and covariance of the Gaussian representing the k-th landmark θ_k, attached to the m-th particle. In the planar robot navigation scenario, each mean $\mu_k^{[m]}$ is a two-element vector, and $\Sigma_k^{[m]}$ is a 2 by 2 matrix.

The posterior over the k-th landmark pose θ_k is easily obtained. Its computation depends on whether or not $n_t = k$, that is, whether or not θ_k was observed at time t. For $n_t = k$, we obtain

$$\begin{aligned} & p(\theta_k \mid s^t, z^t, u^t, n^t) & (9) \\ & \overset{\text{Bayes}}{\propto} p(z_t \mid \theta_k, s^t, z^{t-1}, u^t, n^t)\; p(\theta_k \mid s^t, z^{t-1}, u^t, n^t) & \\ & \overset{\text{Markov}}{=} p(z_t \mid \theta_k, s_t, n_t)\; p(\theta_k \mid s^{t-1}, z^{t-1}, u^{t-1}, n^{t-1}) & \end{aligned}$$

For $n_t \neq k$, we simply leave the Gaussian unchanged:

$$p(\theta_k \mid s^t, z^t, u^t, n^t) = p(\theta_k \mid s^{t-1}, z^{t-1}, u^{t-1}, n^{t-1}) \qquad (10)$$

The FastSLAM algorithm implements the update equation (9) using the extended Kalman filter (EKF). As in existing EKF approaches to SLAM, this filter uses a linearized version of the perceptual model $p(z_t \mid s_t, \theta, n_t)$ [2]. Thus, FastSLAM's EKF is similar to the traditional EKF for SLAM [15] in that it approximates the measurement model using a linear Gaussian function. We note that, with a linear Gaussian observation model, the resulting distribution $p(\theta_k \mid s^t, z^t, u^t, n^t)$ is exactly a Gaussian, even if the motion model is not linear. This is a consequence of the use of sampling to approximate the distribution over the robot's pose.

One significant difference between the FastSLAM algorithm's use of Kalman filters and that of the traditional SLAM algorithm is that the updates in the FastSLAM algorithm involve only a Gaussian of dimension two (for the two landmark location parameters), whereas in the EKF-based SLAM approach a Gaussian of size $2K+3$ has to be updated (with K landmarks and 3 robot pose parameters). This calculation can be done in constant time in FastSLAM, whereas it requires time quadratic in K in standard SLAM.

Calculating the Importance Weights

Let us now return to the problem of calculating the importance weights $w_t^{[m]}$ needed for particle filter resampling, as defined in (7):

$$w_t^{[m]} \propto \frac{p(s^{t,[m]} \mid z^t, u^t, n^t)}{p(s^{t,[m]} \mid z^{t-1}, u^t, n^{t-1})}$$

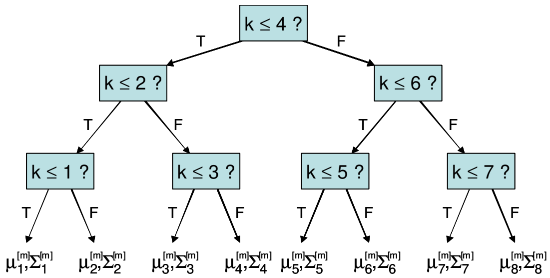

Figure 2: A tree representing $K = 8$ landmark estimates within a single particle.

$$
\begin{aligned}
&\overset{\text{Bayes}}{=} && \frac{p(z_t, n_t \mid s^{t,[m]}, z^{t-1}, u^t, n^{t-1})}{p(z_t, n_t \mid z^{t-1}, u^t, n^{t-1})} \\
& && \frac{p(s^{t,[m]} \mid z^{t-1}, u^t, n^t)}{p(s^{t,[m]} \mid z^{t-1}, u^t, n^t)} \\
&= && \frac{p(z_t, n_t \mid s^{t,[m]}, z^{t-1}, u^t, n^{t-1})}{p(z_t, n_t \mid z^{t-1}, u^t, n^{t-1})} \\
&\propto && p(z_t, n_t \mid s^{t,[m]}, z^{t-1}, u^t, n^{t-1}) \\
&= && \int p(z_t, n_t \mid \theta, s^{t,[m]}, z^{t-1}, u^t, n^{t-1}) \\
& && \quad p(\theta \mid s^{t,[m]}, z^{t-1}, u^t, n^t)\, d\theta \\
&\overset{\text{Markov}}{=} && \int p(z_t, n_t \mid \theta, s_t^{[m]}) \\
& && \quad p(\theta \mid s^{t-1,[m]}, z^{t-1}, u^{t-1}, n^{t-1})\, d\theta \\
&= && \int p(z_t \mid \theta, s_t^{[m]}, n_t)\; p(n_t \mid \theta, s_t^{[m]}) \\
& && \quad p(\theta \mid s^{t-1,[m]}, z^{t-1}, u^{t-1}, n^{t-1})\, d\theta \\
&\propto && \int p(z_t \mid \theta, s_t^{[m]}, n_t) \\
& && \quad p(\theta \mid s^{t-1,[m]}, z^{t-1}, u^{t-1}, n^{t-1})\, d\theta \\
&\overset{\text{EKF}}{\approx} && \int p(z_t \mid \theta_{n_t}^{[m]}, s_t^{[m]}, n_t)\; p(\theta_{n_t}^{[m]})\, d\theta_{n_t} \qquad (11)
\end{aligned}
$$

Here we assume that the distribution $p(n_t \mid \theta, s_t^{[m]})$ is uniform—a common assumption in SLAM. In the last line, "EKF" makes explicit the use of a linearized model as an approximation to the observation model $p(z_t \mid \theta_{n_t}^{[m]}, s_t^{[m]})$, and the resulting Gaussian posterior $p(\theta_{n_t}^{[m]})$. The final integration is easily calculated in closed form for a linear Gaussian.

Efficient Implementation

The FastSLAM algorithm, as described thus far, may require time linear in the number of landmarks K for each update iteration if implemented naively. This is because of the resampling step; every time a particle is added to $\mathcal{S}_t$, its has to be copied. Since each particle contains K landmark estimates, this copying procedure requires $O(MK)$ time. However, most of this copying can be avoided.

Our approach makes it possible to execute a FastSLAM iteration in $O(M \log K)$ time. The basic idea is that the set of Gaussians in each particle is represented by a balanced binary tree. Figure 2 shows such a tree for a single particle, in the case of 8 landmarks. The Gaussian parameters $\mu_k^{[m]}$ and $\Sigma_k^{[m]}$ are located at the leaves of the tree. Clearly, accessing each Gaussian requires time logarithmic in K.

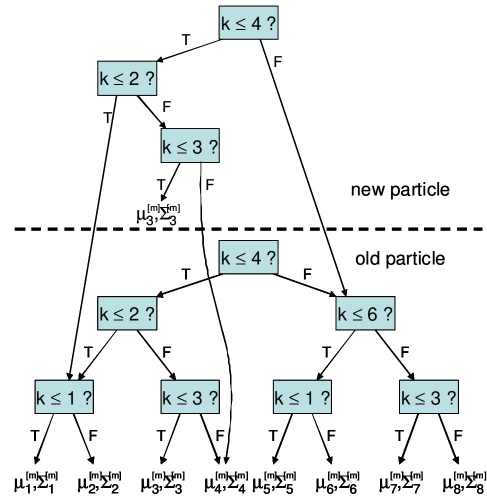

Figure 3: Generating a new particle from an old one, while modifying only a single Gaussian. The new particle receives only a partial tree, consisting of a path to the modified Gaussian. All other pointers are copied from the generating tree.

Suppose FastSLAM incorporates a new control u_t and a new measurement z_t. Each new particle in $\mathcal{S}_t$ will differ from the corresponding one in $\mathcal{S}_{t-1}$ in two ways: First, it will possess a different path estimate obtained via (6), and second, the Gaussian with index n_t will be different in accordance with (9). All other Gaussians 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 3: 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 entirely new tree, only a single path is created, leading to the Gaussian $n_t = 3$. This path is an incomplete tree. To complete the tree, for all branches that leave this path the corresponding pointers are copied 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 such an incomplete tree takes only time logarithmic in K. Moreover, accessing a Gaussian also takes time logarithmic in K, 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 $O(\log K)$. Since in each updating step M new particles are created, an entire update requires time in $O(M \log K)$.

Data Association

In many real-world problems, landmarks are not identifiable, and the total number of landmarks K cannot be obtained trivially—as was the case above. In such situations, the robot has to solve a data association problem between momentary landmarks sightings z_t and the set of landmarks

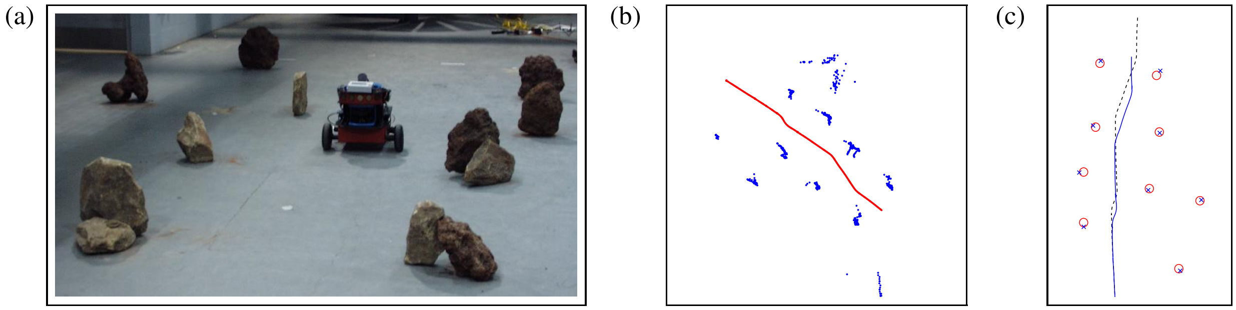

Figure 4: (a) Physical robot mapping rocks, in a testbed developed for Mars Rover research. (b) Raw range and path data. (c) Map generated using FastSLAM (dots), and locations of rocks determined manually (circles).

in the map θ. It also has to determine if a measurement corresponds to a new, previously unseen landmark, in which case the map should be augmented accordingly.

In most existing SLAM solutions based on EKFs, these problems are solved via maximum likelihood. More specifically, the probability of a data association n_t is given by

$$\begin{aligned} & p(n_t \mid z^t, u^t) \\ & \quad = \quad \int p(n_t \mid s^t, z^t, u^t)\, p(s^t \mid z^t, u^t)\, ds^t \\ & \quad \overset{\text{PF}}{\approx} \quad \sum_m p(n_t \mid s^{t,[m]}, z^t, u^t) \\ & \quad \overset{\text{Markov}}{=} \quad \sum_m p(n_t \mid s_t^{[m]}, z_t) \\ & \quad \overset{\text{Bayes}}{\propto} \quad \sum_m p(z_t \mid s_t^{[m]}, n_t) \end{aligned} \tag{12}$$

The step labeled "PF" uses the particle filter approximation to the posterior $p(s^t \mid z^t, u^t)$. The final step assumes a uniform prior $p(n_t \mid s_t)$, which is commonly used [2]. The maximum likelihood data association is simply the index n_t that maximizes (12). If the maximum value of $p(n_t \mid z^t, u^t)$—with careful consideration of all constants in (12)—is below a threshold α, the landmark is considered previously unseen and the map is augmented accordingly.

In FastSLAM, the data association is estimated on a per-particle basis: $n_t^{[m]} = \text{argmax}_{n_t}\, p(z_t|s_t^{[m]}, n_t)$. As a result, different particles may rely on different values of $n_t^{[m]}$. They might even possess different numbers of landmarks in their respective maps. This constitutes a primary difference to EKF approaches, which determine the data association only once for each sensor measurement. It has been observed frequently that false data association will make the conventional EKF approach fail catastrophically [2]. FastSLAM is more likely to recover, thanks to its ability to pursue multiple data associations simultaneously. Particles with wrong data association are (in expectation) more likely to disappear in the resampling process than those that guess the data association correctly.

We believe that, under mild assumptions (e.g., minimum spacing between landmarks and bounded sensor error), the data association search can be implemented in time logarithmic in N. One possibility is the use of kd-trees as an indexing scheme in the tree structures above, instead of the landmark number, as proposed in [11].

Experimental Results

The FastSLAM algorithm was tested extensively under various conditions. Real-world experiments were complimented by systematic simulation experiments, to investigate the scaling abilities of the approach. Overall, the results indicate favorably scaling to large number of landmarks and small particle sets. A fixed number of particles (e.g., $M = 100$) appears to work well across a large number of situations.

Figure 4a shows the physical robot testbed, which consists of a small arena set up under NASA funding for Mars Rover research. A Pioneer robot equipped with a SICK laser range finder was driven along an approximate straight line, generating the raw data shown in Figure 4b. The resulting map generated with $M = 10$ samples is depicted in Figure 4c, with manually determined landmark locations marked by circles. The robot's estimates are indicated by x's, illustrating the high accuracy of the resulting maps. FastSLAM resulted in an average residual map error of 8.3 centimeters, when compared to the manually generated map.

Unfortunately, the physical testbed does not allow for systematic experiments regarding the scaling properties of the approach. In extensive simulations, the number of landmarks was increased up to a total of 50,000, which FastSLAM successfully mapped with as few as 100 particles. Here, the number of parameters in FastSLAM is approximately 0.3% of that in the conventional EKF. Maps with 50,000 landmarks are entirely out of range for conventional SLAM techniques, due to their enormous computational complexity. Figure 5 shows example maps with smaller numbers of landmarks, for different maximum sensor ranges as indicated. The ellipses in Figure 5 visualize the residual uncertainty when integrated over all particles and Gaussians.

In a set of experiments specifically aimed to elucidate the scaling properties of the approach, we evaluated the map and robot pose errors as a function of the number of landmarks K, and the number of particles M, respectively. The results are graphically depicted in Figure 6. Figure 6a illustrates that an increase in the number of landmarks K mildly reduces the error in the map and the robot pose. This is because the larger the number of landmarks, the smaller the robot pose error at any point in time. Increasing the number of particles M also bears a positive effect on the map and pose errors, as illustrated in Figure 6b. In both diagrams, the bars correspond to 95% confidence intervals.

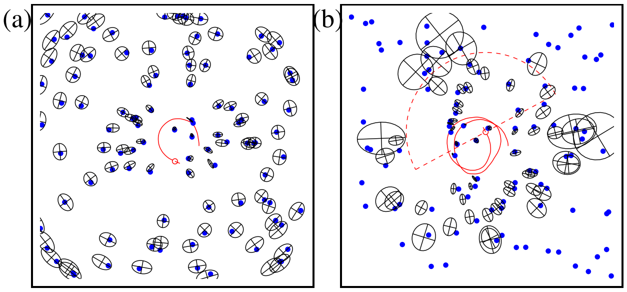

Figure 5: Maps and estimated robot path, generated using sensors with (a) large and (b) small perceptual fields. The correct landmark locations are shown as dots, and the estimates as ellipses, whose sizes correspond to the residual uncertainty.

Conclusion

We presented the FastSLAM algorithm, an efficient new solution to the concurrent mapping and localization problem. This algorithm utilizes a Rao-Blackwellized representation of the posterior, integrating particle filter and Kalman filter representations. Similar to Murphy's work [12], FastSLAM is based on an inherent conditional independence property of the SLAM problem. However, Murphy's approach maintains maps using grid positions with discrete values, and therefore scales poorly with the size of the map. His approach also did not deal with the data association problem, which does not arise in the grid-based setting.

In FastSLAM, landmark estimates are efficiently represented using tree structures. Updating the posterior requires $O(M \log K)$ time, where M is the number of particles and K the number of landmarks. This is in contrast to the $O(K^2)$ complexity of the common Kalman-filter based approach to SLAM. Experimental results illustrate that FastSLAM can build maps with orders of magnitude more landmarks than previous methods. They also demonstrate that under certain conditions, a small number of particles works well regardless of the number of landmarks.

Acknowledgments We thank Kevin Murphy and Nando de Freitas for insightful discussions on this topic. This research was sponsored by DARPA's MARS Program (Contract number N66001-01-C-6018) and the National Science Foundation (CAREER grant number IIS-9876136 and regular grant number IIS-9877033). We thank the Hertz Foundation for their support of Michael Montemerlo's graduate research. Daphne Koller was supported by the Office of Naval Research, Young Investigator (PECASE) grant N00014-99-1-0464. This work was done while Sebastian Thrun was visiting Stanford University.

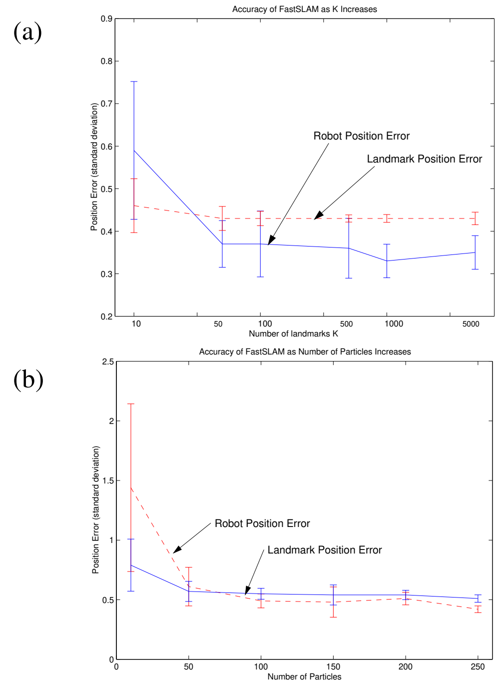

Figure 6: Accuracy of the FastSLAM algorithm as a function of (a) the number of landmarks N, and (b) the number of particles M. Large number of landmarks reduce the robot localization error, with little effect on the map error. Good results can be achieved with as few as 100 particles.

References

[1] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte Carlo localization for mobile robots. *ICRA-99*.

[2] G. Dissanayake, P. Newman, S. Clark, H.F. Durrant-Whyte, and M. Csorba. An experimental and theoretical investigation into simultaneous localisation and map building (SLAM). *Lecture Notes in Control and Information Sciences: Experimental Robotics VI*, Springer, 2000.

[3] 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.

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

[5] A Doucet, N. de Freitas, K. Murphy, and S. Russell. Rao-Blackwellised particle filtering for dynamic Bayesian networks. *UAI-2000*.

[6] J. Guivant and E. Nebot. Optimization of the simultaneous localization and map building algorithm for real time implementation. *IEEE Transaction of Robotic and Automation*, May 2001.

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

[8] J.J. Leonard and H.J.S. Feder. A computationally efficient method for large-scale concurrent mapping and localization. *ISRR-99*.

[9] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4, 1997.

[10] 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, 1953.

[11] A.W. Moore. Very fast EM-based mixture model clustering using multiresolution kd-trees. *NIPS-98*.

[12] K. Murphy. Bayesian map learning in dynamic environments. *NIPS-99*.

[13] K. Murphy and S. Russell. Rao-blackwellized particle filtering for dynamic bayesian networks. In *Sequential Monte Carlo Methods in Practice*, Springer, 2001.

[14] P. Newman. *On the Structure and Solution of the Simultaneous Localisation and Map Building Problem*. PhD thesis, Univ. of Sydney, 2000.

[15] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In *Autonomous Robot Vehnicles*, Springer, 1990.

[16] C. Thorpe and H. Durrant-Whyte. Field robots. *ISRR-2001*.

[17] S. Thrun, D. Fox, and W. Burgard. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning*, 31, 1998.