Particle Filters for Mobile Robot Localization

Dieter Fox, Sebastian Thrun, Wolfram Burgard, and Frank Dellaert

1 Introduction

This chapter investigates the utility of particle filters in the context of mobile robotics. In particular, we report results of applying particle filters to the problem of mobile robot localization, which is the problem of estimating a robot's pose relative to a map of its environment. The localization problem is a key problem in mobile robotics, as it plays a fundamental role in various successful mobile robot systems; see e.g., (Cox and Wilfong 1990, Fukuda et al. 1993, Hinkel and Knieriemen 1988, Leonard et al. 1992, Rencken 1993, Simmons et al. 1997, Weiß et al. 1994) and various chapters in (Borenstein et al. 1996) and (Kortenkamp et al. 1998). Occasionally, it has been referred to as "the most fundamental problem to providing a mobile robot with autonomous capabilities" (Cox 1991).

The mobile robot localization problem comes in different flavors. The simplest localization problem—which has received by far the most attention in the literature—is *position tracking*. Here the initial robot pose is known, and localization seeks to correct small, incremental errors in a robot's odometry. More challenging is the *global localization problem*, where a robot is not told its initial pose, but instead has to determine it from scratch. The global localization problem is more difficult, since the robot's localization error can be arbitrarily large. Even more difficult is the *kidnapped robot problem* (Engelson and McDermott 1992), in which a well-localized robot is teleported to some other position without being told. This problem differs from the global localization problem in that the robot might firmly believe to be somewhere else at the time of the kidnapping. The kidnapped robot problem is often used to test a robot's ability to recover autonomously from catastrophic localization failures. Finally, there also exists the *multi-robot localization problem*, in which a team of robots seeks to localize themselves. The multi-robot localization problem is particularly interesting if robots are able to perceive each other, which introduces non-trivial statistical dependencies in the individual robots' estimates.

The beauty of particle filters is that they provide solutions to all of the problems above. Even the most straightforward implementation of particle filters exhibits excellent results for the position tracking and the global localization problem. Extensions of the basic algorithm have led to excellent results on the kidnapped robot and the multi-robot localization problem.

The power of particle filters relative to these problems stems from multiple aspects: in contrast to the widely used Kalman filters, particle filters can approximate a large range of probability distributions, not just normal distributions. Once a robot's belief is focused on a subspace of the space of all poses, particle filters are computationally efficient, since they focus their resources on regions in state space with high likelihood. Particle filters are also easily implemented as any-time filters (Dean and Boddy 1988, Zilberstein and Russell 1995), by dynamically adapting the number of samples based on the available computational resources. Finally, particle filters for localization are remarkably easy to implement, which also contributes to their popularity.

This article describes a family of methods, known as *Monte Carlo localization (MCL)* (Dellaert at al. 1999b, Fox et al. 1999b). The MCL algorithm is a particle filter combined with probabilistic models of robot perception and motion. Building on this, we will describe a variation of MCL which uses a different proposal distribution (a mixture distribution) that facilitates fast recovery from global localization failures. As we will see, this proposal distribution has a range of advantages over that used in standard MCL, but it comes at the price that it is more difficult to implement, and it requires an algorithm for sampling poses from sensor measurements, which might be difficult to obtain. Finally, we will present an extension of MCL to cooperative multi-robot localization of robots that can perceive each other during localization. All these approaches have been tested thoroughly in practice. Experimental results are provided to demonstrate their relative strengths and weaknesses in practical robot applications.

2 Monte Carlo Localization

2.1 Bayes Filtering

Particle filters have already been discussed in the introductory chapters of this book. For the sake of consistency, let us briefly derive the basics, beginning with Bayes filters. Bayes filters address the problem of estimating the state x of a dynamical system from sensor measurements. For example, in mobile robot localization the dynamical system is a mobile robot and its environment, the state is the robot's pose therein (often specified by a position in a two-dimensional Cartesian space and the robot's heading direction θ), and measurements may include range measurements, camera images, and odometry readings. Bayes filters assume that the environment is *Markov*, that is, past and future data are (conditionally) independent if one knows the current state.

The key idea of Bayes filtering is to estimate the posterior probability density over the state space conditioned on the data. In the robotics and AI literature, this posterior is typically called the *belief*. Throughout this chapter, we will use the following notation:

$$Bel(x_t) \quad = \quad p(x_t \mid d_{0\ldots t})$$

Here x denotes the state, x_t is the state at time t, and $d_{0\ldots t}$ denotes the data starting at time 0 up to time t. For mobile robots, we distinguish two types of data: *perceptual data* such as laser range measurements, and *odometry data* or *controls*, which carries information about robot motion. Denoting the former by y and the latter by u, we have

$$Bel(x_t) \quad = \quad p(x_t \mid y_t, u_{t-1}, y_{t-1}, u_{t-2} \ldots, u_0, y_0) \tag{2.1}$$

Without loss of generality, we assume that observations and actions occur in an alternating sequence. Note that the most recent perception in $Bel(x_t)$ is y_t, whereas the most recent controls/odometry reading is u_{t-1}.

Bayes filters estimate the belief *recursively*. The *initial* belief characterizes the *initial* knowledge about the system state. In the absence of such knowledge (e.g., global localization), it is typically initialized by a *uniform distribution* over the state space.

To derive a recursive update equation, we observe that Expression (2.1) can be transformed by Bayes rule to

$$\begin{aligned} Bel(x_t) \quad &= \quad \frac{p(y_t \mid x_t, u_{t-1}, \ldots, y_0)\, p(x_t \mid u_{t-1}, \ldots, y_0)}{p(y_t \mid u_{t-1}, \ldots, y_0)} \\ &= \quad \frac{p(y_t \mid x_t, u_{t-1}, \ldots, y_0)\, p(x_t \mid u_{t-1}, \ldots, y_0)}{p(y_t \mid u_{t-1}, d_{0\ldots t-1})} \qquad (2.2) \end{aligned}$$

The *Markov assumption* states that measurements y_t are conditionally independent of past measurements and odometry readings given knowledge of the state x_t:

$$p(y_t \mid x_t, u_{t-1}, \ldots, y_0) \quad = \quad p(y_t \mid x_t)$$

This allows us to conveniently simplify Equation (2.2):

$$Bel(x_t) \quad = \quad \frac{p(y_t \mid x_t)\, p(x_t \mid u_{t-1}, \ldots, y_0)}{p(y_t \mid u_{t-1}, d_{0\ldots t-1})}$$

To obtain our final recursive form, we now have to integrate out the pose x_{t-1} at time $t-1$, which yields

$$= \quad \frac{p(y_t \mid x_t)}{p(y_t \mid u_{t-1}, d_{0\ldots t-1})} \int p(x_t \mid x_{t-1}, u_{t-1}, \ldots, y_0)\, p(x_{t-1} \mid u_{t-1}, \ldots, y_0)\, dx_{t-1}$$

The *Markov assumption* also implies that given knowledge of x_{t-1} and u_{t-1}, the state x_t is conditionally independent of past measurements $y_1 \ldots, y_{t-1}$ and odometry readings $u_1 \ldots, u_{t-2}$ up to time $t-2$, that is:

$$p(x_t \mid x_{t-1}, u_{t-1}, \ldots, y_0) \quad = \quad p(x_t \mid x_{t-1}, u_{t-1})$$

Using the definition of the belief Bel, we obtain a recursive estimator known as *Bayes filter*:

$$\begin{aligned} Bel(x_t) &= \frac{p(y_t \mid x_t)}{p(y_t \mid u_{t-1}, d_{0\ldots t-1})} \int p(x_t \mid x_{t-1}, u_{t-1})\, Bel(x_{t-1})\, dx_{t-1} \\ &= \eta\, p(y_t \mid x_t) \int p(x_t \mid x_{t-1}, u_{t-1})\, Bel(x_{t-1})\, dx_{t-1} \qquad (2.3) \end{aligned}$$

where η is a normalizing constant. This equation is of central importance, as it is the basis for various MCL algorithms studied here.

2.2 Models of Robot Motion and Perception

In the context of mobile robot localization, Bayes filters are also known as *Markov localization* (Burgard, Fox, Hennig and Schmidt 1996, Fox at al. 1999a, Kaelbling et al. 1996, Koenig and Simmons 1996, Nourbakhsh et al. 1995, Simmons and Koenig 1995, Thrun 1998). To implement Markov localization, one needs to know three distributions: the initial belief $Bel(x_0)$ (e.g., uniform), the next state probability $p(x_t \mid x_{t-1}, u_{t-1})$ (called the *motion model*), and the perceptual likelihood $p(y_t \mid x_t)$ (called the *perceptual model*). The specific shape of these probabilities depends on the robot's odometry, and the type of sensors used for localization. Both of these models are time-invariant; we will henceforth omit the time index t.

A specific motion model (for an RWI B21 robot) is shown in Figure 1. This figure shows the probabilistic outcome of two example motion commands indicated by the lines. The grey-scale corresponds to $p(x' \mid x, a)$, projected into 2D. This specific model is the result of convolving conventional robot kinematics with two independent zero-mean random variables, one of which models noise in rotation, and one models translational noise. The model is easily coded in 20 lines of C code.

The perceptual model $p(y \mid x)$ depends on the specific sensor. If y are raw camera images, computing $p(y \mid x)$ is related to the computer graphics problem in that the appearance of an image y at pose x has to be predicted. However, $p(y \mid x)$ is considerably simpler if one uses range finders for perception. Such sensors measure the distance of the robot to nearby obstacles, using sound or structured laser light. Figure 2 illustrates the model of robot perception for a planar 2D laser range finder, which is commonly used in mobile robotics. Figure 2a shows a laser scan and a map. The specific density

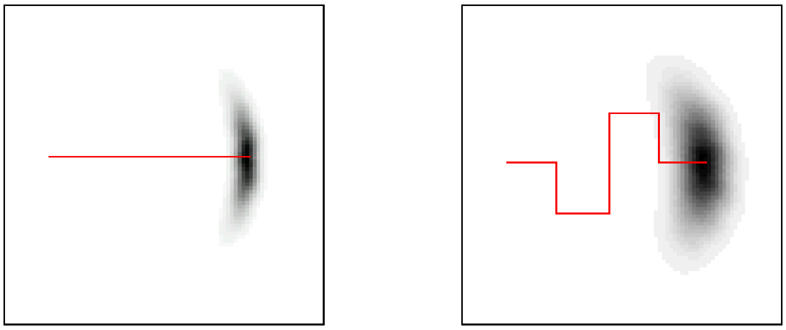

Figure 1: The density $p(y \mid x)$ after moving 40 meters (left diagram) and 80 meters (right diagram). The darker a pose, the more likely it is.

$p(y \mid x)$ is computed in two stages. First, the measurement in an ideal, noise-free environment is computed. For laser range finders, this is easily done using ray-tracing in a geometric map of the environment, such as the one shown in Figure 2a. Second, the desired density $p(y \mid x)$ is obtained as a mixture of random variables, composed of one that models the event of getting the correct reading (convolved with small Gaussian-distributed measurement noise), one for receiving a max-range reading (which occurs frequently), and one that models random noise and is exponentially distributed. Figure 2b shows a picture of $p(y \mid x)$, and Figure 2c plots $p(y \mid x)$ for the specific sensor scan y shown in Figure 2a.

2.3 Implementation as Particle Filters

If the state space is continuous, as is the case in mobile robot localization, implementing the belief update equation (2.3) is *not* a trivial matter—particularly if one is concerned about efficiency. The idea of MCL (and other particle filter algorithms) is to represent the belief $Bel(x)$ by a set of m weighted samples distributed according to $Bel(x)$:

$$Bel(x) \quad = \quad \{x^{(i)}, p^{(i)}\}_{i=1,\ldots,m}$$

Here each $x^{(i)}$ is a sample (a state), and $p^{(i)}$ are non-negative numerical factors called *importance factors*, which sum up to one. As the name suggests, the importance factors determine the weight (=importance) of each sample.

In global mobile robot localization, the *initial* belief is a set of poses drawn according to a uniform distribution over the robot's universe, annotated by the uniform importance factor $\frac{1}{m}$.

The recursive update is realized in three steps, computing the expression in (2.3) *from the right to the left*.

1. Sample a state x_{t-1} from $Bel(x_{t-1})$, by drawing a random $x_{t-1}^{(i)}$ from the

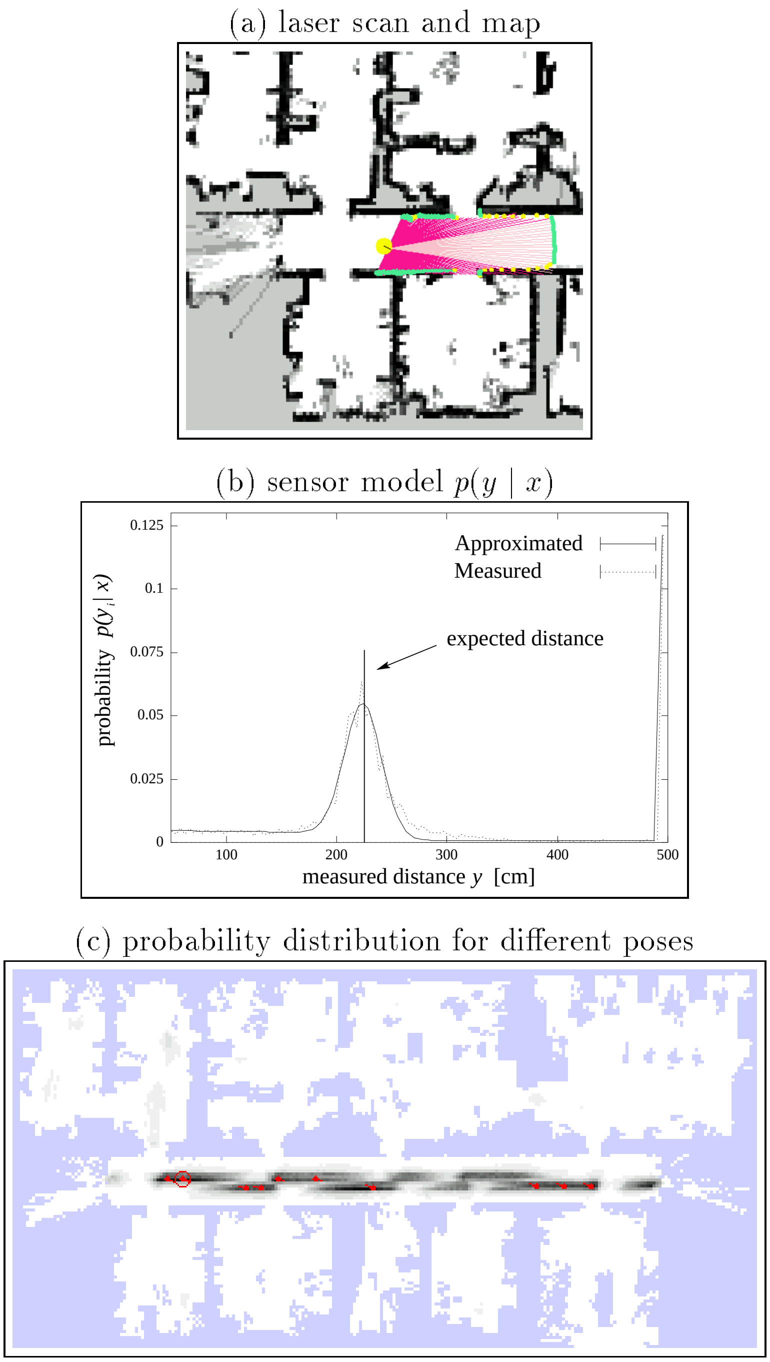

Figure 2: (a) Laser range scan, projected into a map. (b) The density $p(y \mid x)$. (c) $p(y \mid x)$ for the scan shown in (a). Based on a single sensor scan, the robot assigns high likelihood for being somewhere in the main corridor.

sample set representing $Bel(x_{t-1})$ according to the (discrete) distribution defined through the importance factors $p_{t-1}^{(i)}$.

2. Use the sample $x_{t-1}^{(i)}$ and the action u_{t-1} to sample $x_t^{(j)}$ from the distribution $p(x_t \mid x_{t-1}, u_{t-1})$. The predictive density of $x_t^{(j)}$ is now given by the product $p(x_t \mid x_{t-1}, u_{t-1})Bel(x_{t-1})$.

3. Finally, weight the sample $x_t^{(j)}$ by the (non-normalized) importance factor $p(y_t \mid x_t^{(j)})$, the likelihood of the sample $x_t^{(j)}$ given the measurement y_t.

After the generation of m samples, the new importance factors are normalized so that they sum up to 1 (hence define a probability distribution). The reader should quickly see that this procedure in fact implements (2.3), using an (approximate) sample-based representation. Obviously, our algorithm constitutes just one possible implementation of the particle filtering idea; other sampling schemes exist that further reduce variance (Kitagawa 1996). Detailed convergence results can be found in Chapters 2 and 3 of this book.

Further below, it will be convenient to notice that in this version of MCL, the proposal distribution for approximating $Bel(x_t)$ via importance sampling is given by

$$q \quad := \quad p(x_t \mid x_{t-1}, u_{t-1})Bel(x_{t-1}) \tag{2.4}$$

which is used to approximate the desired posterior

$$\frac{p(y_t \mid x_t)\; p(x_t \mid u_{t-1}, x_{t-1})\; Bel(x_{t-1})}{p(y_t \mid d_{0\ldots t-1}, u_{t-1})} \tag{2.5}$$

Consequently, the importance factors are given by the quotient

$$\begin{aligned} & [p(x_t \mid x_{t-1}, u_{t-1})Bel(x_{t-1})]^{-1}\, \frac{p(y_t \mid x_t)\; p(x_t \mid u_{t-1}, x_{t-1})\; Bel(x_{t-1})}{p(y_t \mid d_{0\ldots t-1}, u_{t-1})} \\ \propto \quad & p(y_t \mid x_t) \end{aligned} \tag{2.6}$$

2.4 Robot Results

MCL has been at the core of our robot navigation software. It is more efficient and accurate than any of our previous algorithms. We thoroughly tested MCL in a range of real-world environments, applying it to at least three different types of sensors (cameras, sonar, and laser proximity data). Our experiments have been carried out using several B21, B18 Pioneer, Scout, and XR4000 robots, two of which are shown in Figure 3. These robots were equipped with arrays of sonar sensors (from 7 to 24), one or two laser range finders, and in

Figure 3: Two of the robots used for testing: RHINO (left) and MINERVA (center and right), which successfully guided thousands of people through crowded museums.

the case of Minerva, the robot shown in center and right of Figure 3, a B/W camera pointed at the ceiling.

A typical example of MCL is shown in Figure 4. This example illustrates MCL in the context of localizing a mobile robot globally in an office environment. This robot is equipped with sonar range finders, and it is also given a map of the environment. In Figure 4a, the robot is globally uncertain; hence the samples are spread uniformly trough the free-space (projected into 2D). Figure 4b shows the sample set after approximately 1 meter of robot motion, at which point MCL has disambiguated the robot's position up to a single symmetry. Finally, after another 2 meters of robot motion the ambiguity is resolved, and the robot knows where it is. The majority of samples is now centered tightly around the correct position, as shown in Figure 4c.

2.5 Comparison to Grid-Based Localization

To elucidate the advantage of particle filters over alternative representations, we are particularly interested in grid-based representations, which are at the core of an alternative family of Markov localization algorithms (Fox et al. 1998). The algorithm described in (Fox et al. 1998) relies on a fine-grained grid approximation of the belief $Bel()$, using otherwise identical sensor and motion models. Figure 5 plots the localization accuracy for grid-based localization as a function of the grid resolution. Note that the results in Figure 5 were not generated in real-time. As shown there, the accuracy increases with the resolution of the grid, both for sonar (solid line) and for laser data (dashed line). However, grid sizes beyond 8 cm do not permit updating in real-time, even when highly efficient, selective update schemes are used (Fox et al. 1998).

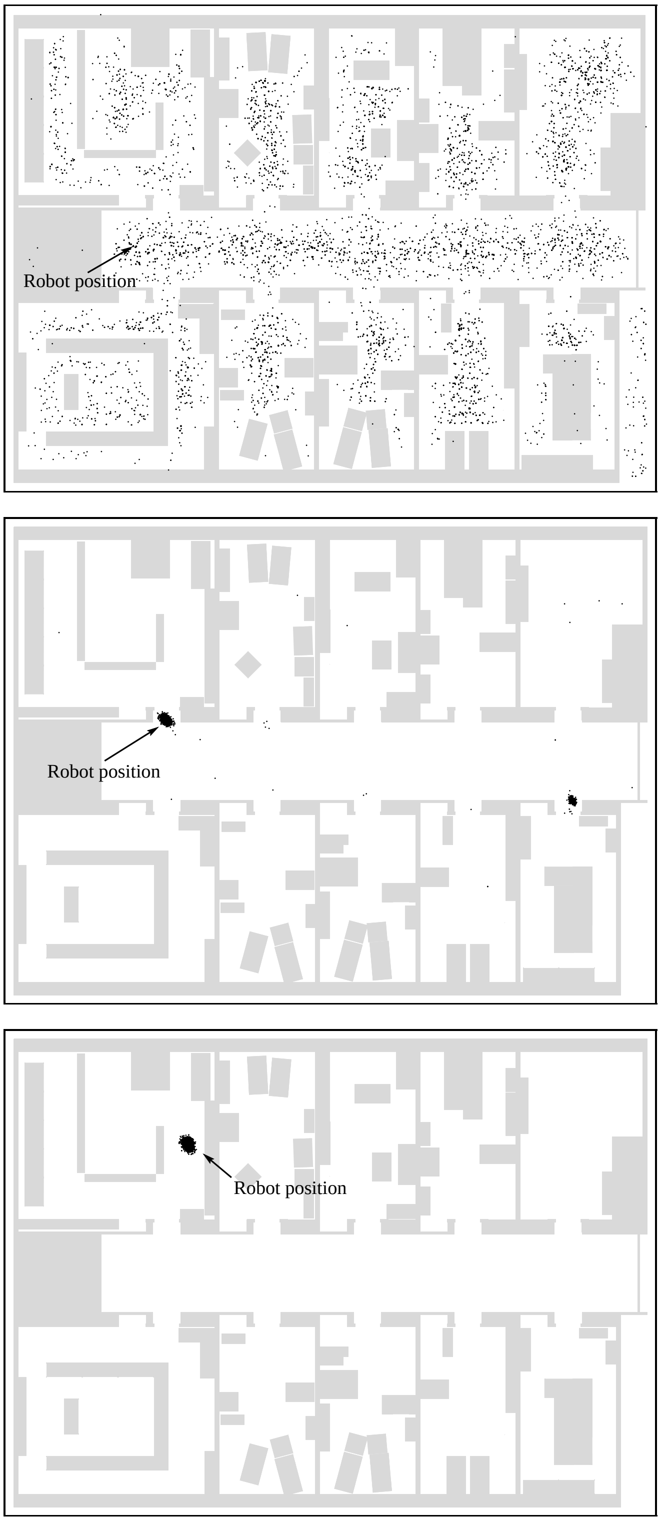

Figure 4: Global localization of a mobile robot using MCL (10,000 samples).

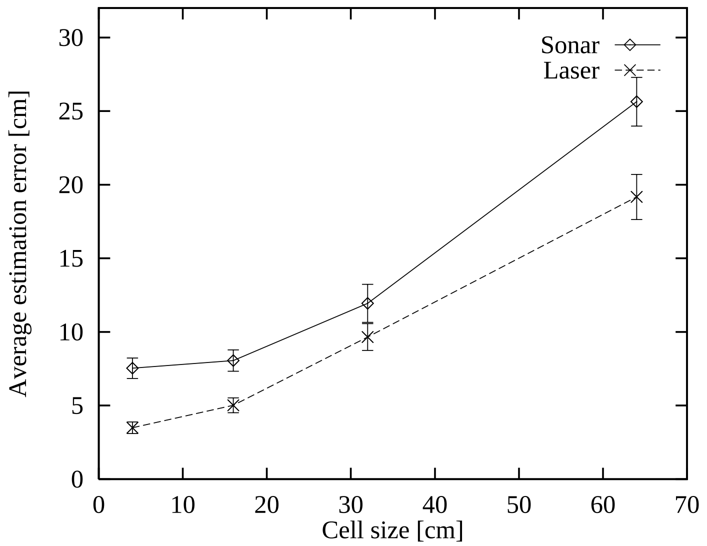

Figure 5: Accuracy of grid-based Markov localization using different spatial resolutions.

Results for MCL with fixed sample set sizes are shown in Figure 6. These results have been generated using real-time conditions, where large sample sizes (> 1,000 samples) result in loss of sensor data due to time constraints. Here very small sample sets are disadvantageous, since they infer too large an error in the approximation. Large sample sets are also disadvantageous, since processing them requires too much time and fewer sensor items can be processed in real-time. The "optimal" sample set size, according to Figure 6, is somewhere between 1,000 and 5,000 samples. Grid-based localization, to reach the same level of accuracy, has to use grids with 4cm resolution—which is infeasible given even our fastest computers we currently have.

In comparison, the grid-based approach, with a resolution of 20 cm, requires almost exactly ten times as much memory when compared to MCL with 5,000 samples. During global localization, integrating a single sensor scan requires up to 120 seconds using the grid-based approach, whereas MCL consumes consistently less than 3 seconds under otherwise equal conditions. This illustrates that particle filters are clearly superior over grid-based representations, which previously was among the best known algorithms for the global localization problem.

Similar results were obtained using a camera as the primary sensor for localization (Dellaert et al. 1999a). To test MCL under extreme circumstances, we evaluated it using data collected in a populated museum. During a two-week exhibition, our robot Minerva (Figure 3) was employed as a tour-guide in the Smithsonian's Museum of Natural History, during which it traversed more than 44km (Thrun et al. 1999). To aid localization, Minerva is equipped with a camera pointed towards the ceiling. Using this camera, the brightness

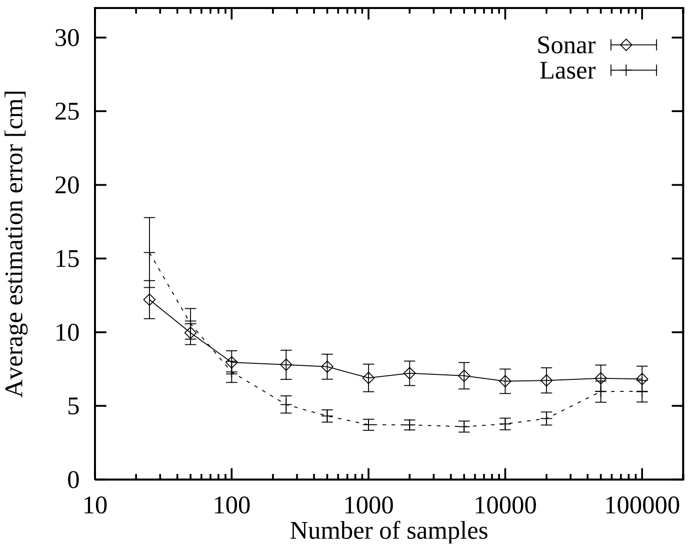

Figure 6: Accuracy of MCL for different numbers of samples (log scale).

of a small patch of the ceiling directly above the robot is measured, and compared to a large-scale mosaic of the museum's ceiling obtained beforehand (Dellaert et al. 1999c), shown in Figure 7. This constitutes the likelihood model. The data used here is among the most difficult data sets in our possession, as the robot traveled with speeds of up to 163 cm/sec. Whenever it entered or left the carpeted area in the center of the museum, it crosses a 2cm bump which introduced significant errors in the robot's odometry.

When *only* using vision information, grid-based localization fatally failed to track the robot. This is because the enormous computational overhead makes it impossible to incorporate sufficiently many images. MCL, however, succeeded in globally localizing the robot, and tracking the robot's position. Figure 8 shows an example of global localization with MCL. In the beginning the robot starts with 2,000 uniformly distributed samples representing the absolute uncertainty about the robots position. After incorporating 15 images (first diagram), the samples are still scattered over the whole area but already started to concentrate on several locations. After incorporating 38 images, most of the ambiguities are resolved and the samples are concentrated on a small number of peaks (second diagram). Finally, after 126 iterations, the robot is highly certain about its position (third diagram), which is represented by a concentration of the samples of the true location of the robot.

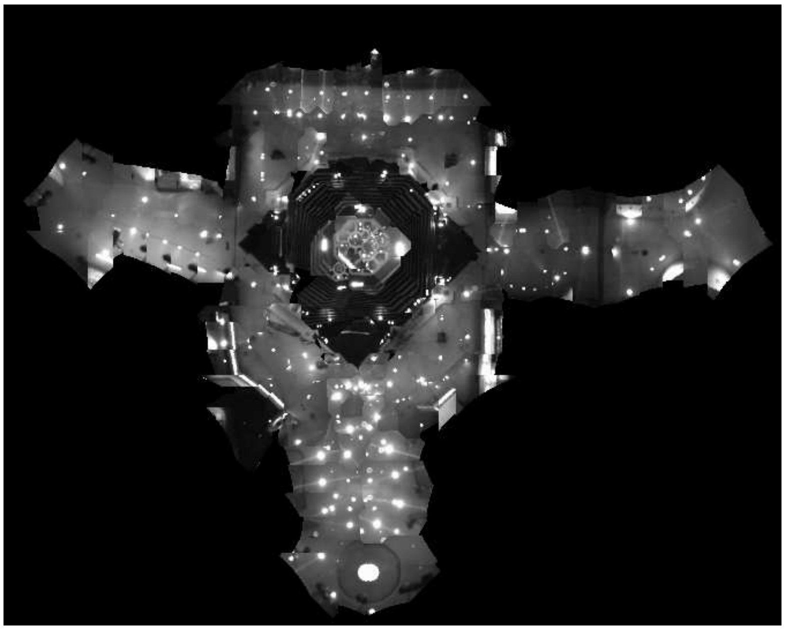

Figure 7: Ceiling map of the National Museum of American History, which was used as the perceptual model in navigating with a vision sensor.

3 MCL with Mixture Proposal Distributions

3.1 The Need For Better Sampling

As noticed by several authors (Doucet 1998, Lenser and Veloso 2000, Liu and Chen 1998, Pitt and Shephard 1999), the basic particle filter performs poorly if the proposal distribution, which is used to generate samples, places too little samples in regions where the desired posterior $Bel(x_t)$ is large.

This problem has indeed great practical importance in the context of MCL, as the following example illustrates. The solid curve in Figure 9 shows the accuracy MCL achieves after 100 steps, using $m = 1,000$ samples. These results were obtained in simulation, enabling us to vary the amount of perceptual noise from 50% (on the right) to 1% (on the left); in particular, we simulated a mobile robot localizing an object in 3D space from mono-camera imagery. It appears that MCL works best for 10% to 20% perceptual noise. The degradation of performance towards the right, when there is *high* noise, barely surprises. The less accurate a sensor, the larger an error one should expect. However, MCL also performs poorly when the noise level is too small. In other words, MCL with accurate sensors may perform *worse* than MCL with inaccurate sensors. This finding is a bit counter-intuitive in that it suggests that MCL only works well in specific situations, namely those where the sensors possess the "right" amount of noise.

At first glance, one might attempt to fix the problem by using a perceptual

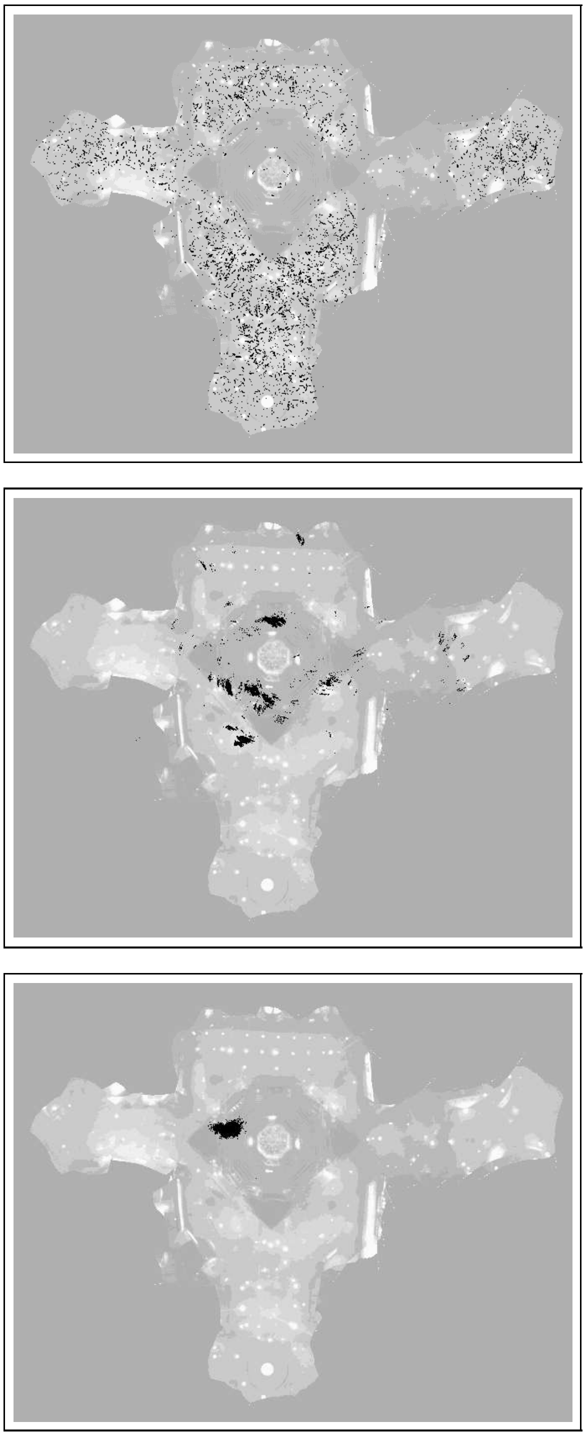

Figure 8: Global localization of a mobile robot using a camera pointed at the ceiling.

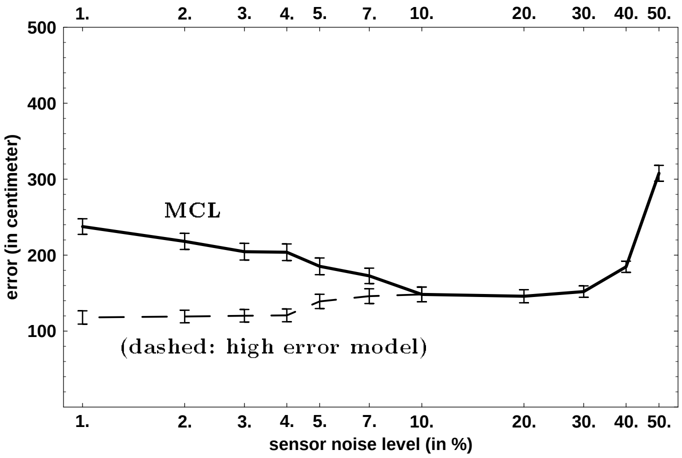

Figure 9: Solid curve: error of MCL after 100 steps, as a function of the sensor noise. 95% confidence intervals are indicated by the bars. Note that this function is *not* monotonic, as one might expect. Dashed curve: Same experiment with high-error model.

likelihood $p(y_t \mid x_t)$ that overestimates the sensor noise. In fact, such a strategy partially alleviates the problem: The dashed curve in Figure 9b shows the accuracy if the error model assumes a fixed 10% noise (shown there only for smaller "true" error rates). While the performance is better, this is hardly a principled way of fixing the problem. The overly pessimistic sensor model is inaccurate, throwing away precious information in the sensor readings. In fact, the resulting belief is not any longer a posterior, even if infinitely many samples were used. As we will see below, a mathematically sound method exists that produces much better results.

To analyze the problem more thoroughly, we first notice that the true goal of Bayes filtering is to calculate the product distribution specified in Equation (2.5). Thus, the optimal proposal distribution would be this product distribution. However, sampling from this distribution directly is too difficult. As noticed above, MCL samples instead from the proposal distribution q defined in Equation (2.4), and uses the importance factors (2.6) to account for the difference. It is well-known from the statistical literature (Doucet 1998, Pitt and Shephard 1999, Liu and Chen 1998, Tanner 1993) that the divergence between (2.5) and (2.4) determines the convergence speed. This difference is accounted by the perceptual density $p(y_t \mid x_t)$: If the sensors are entirely uninformative, this distribution is flat and (2.5) equals (2.4). For low-noise sensors, however, $p(y_t \mid x_t)$ is typically quite narrow, hence MCL converges slowly. Thus, the error in Figure 9 is in fact caused by two different types of errors: one arising from the limitation of the sensor data (=noise), and one that arises from the mismatch of (2.5) and (2.4) in MCL.

This suggests to use different proposal distributions for sampling that can accommodate highly accurate sensors.

3.2 An Alternative Proposal Distribution

To alleviate this problem, one can use a different proposal distribution, one that samples according to the most recent sensor measurement y_t (see also (Lenser and Veloso 2000, Thrun et al. 2000)). The key idea is to sample x_t directly from a distribution that is proportional to the perceptual likelihood $p(y_t \mid x_t)$:

$$\bar{q} \;:=\; \frac{p(y_t \mid x_t)}{\pi(y_t)} \qquad \text{with} \qquad \pi(y_t) \;=\; \int p(y_t \mid x_t)\, dx_t \tag{3.1}$$

This new proposal distribution possesses orthogonal limitations from the one described above, in that it generates samples that are highly consistent with the most recent sensor measurement but ignorant of the belief $Bel(x_{t-1})$ and the control u_{t-1}.

The importance factors for these samples can be calculated in three ways. Recall that our goal is to sample from the product distribution

$$\frac{p(y_t \mid x_t)\; p(x_t \mid u_{t-1}, x_{t-1})\; Bel(x_{t-1})}{p(y_t \mid d_{0\ldots t-1}, u_{t-1})} \;=\; \frac{p(y_t \mid x_t)\; p(x_t \mid d_{0\ldots t-1}, u_{t-1})}{p(y_t \mid d_{0\ldots t-1}, u_{t-1})} \tag{3.2}$$

Approach 1 (proposed by Arnaud Doucet, personal communication): The idea is to draw random pairs $\langle x_t^{(i)}, x_{t-1}^{(i)} \rangle$ by sampling $x_t^{(i)}$ as described above, and $x_{t-1}^{(i)}$ by drawing from $Bel(x_{t-1})$. Obviously, the combined proposal distribution is then given by

$$\frac{p(y_t \mid x_t^{(i)})}{\pi(y_t)} \times Bel(x_{t-1}^{(i)}) \tag{3.3}$$

and hence the importance factors are given by the quotient

$$\begin{aligned} & \left[\frac{p(y_t \mid x_t^{(i)})}{\pi(y_t)} \times Bel(x_{t-1}^{(i)}) \right]^{-1} \frac{p(y_t \mid x_t^{(i)})\; p(x_t^{(i)} \mid u_{t-1}, x_{t-1}^{(i)})\; Bel(x_{t-1}^{(i)})}{p(y_t \mid d_{0\ldots t-1}, u_{t-1})} \\ = \;& \frac{p(x_t^{(i)} \mid u_{t-1}, x_{t-1}^{(i)})\; \pi(y_t)}{p(y_t \mid d_{0\ldots t-1}, u_{t-1})} \\ \propto \;& p(x_t^{(i)} \mid u_{t-1}, x_{t-1}^{(i)}) \end{aligned} \tag{3.4}$$

This approach is mathematically more elegant than the two alternatives described below, in that it avoids the need to transform sample sets into densities (which will be the case below). We have not yet implemented this approach, hence are unable to comment on how well it works in practice. However, in the context of global mobile robot localization, we suspect the importance factor $p(x_t^{(i)} \mid u_{t-1}, x_{t-1}^{(i)})$ will be zero for many pose pairs $\langle x_t^{(i)}, x_{t-1}^{(i)} \rangle$.

Approach 2 An alternative approach uses forward sampling and kd-trees to generate an approximate density of $p(x_t \mid d_{0\ldots t-1}, u_{t-1})$. This density is then used in a second phase to calculate the desired importance factor. More specifically, Equations (3.1) and (3.2) suggest that the importance factors of a sample $x_t^{(i)}$ can be written as

$$\left[\frac{p(y_t \mid x_t^{(i)})}{\pi(y_t)}\right]^{-1} \frac{p(y_t \mid x_t^{(i)})\, p(x_t^{(i)} \mid d_{0\ldots t-1}, u_{t-1})}{p(y_t \mid d_{0\ldots t-1}, u_{t-1})}$$

$$\propto \quad p(x_t^{(i)} \mid d_{0\ldots t-1}, u_{t-1}) \tag{3.5}$$

Computing these importance factors is *not* trivial, since $Bel(x_{t-1})$ is represented by a set of samples. The "trick" here is to employ a two-staged approach, which first approximates $p(x_t \mid d_{0\ldots t-1}, u_{t-1})$ and then uses this approximate density to calculate the desired importance factors.

The following algorithm implements this alternative importance sampler:

1. Generate a set of samples $x_t^{(j)}$, by first sampling from $Bel(x_{t-1}^{(j)})$ and then sampling from $p(x_t^{(j)} \mid u_{t-1}, x_{t-1}^{(j)})$ as described above. Obviously, these samples approximate $p(x_t^{(j)} \mid d_{0\ldots t-1}, u_{t-1})$.

2. Transform the resulting sample set into a kd-tree (Bentley 1980, Moore 1990). The tree generalizes samples to arbitrary poses $x_t^{(j)}$ in pose space, which is necessary to calculate the desired importance factors.

3. Finally, sample $x_t^{(i)}$ from our proposal distribution $\frac{p(y_t \mid x_t^{(i)})}{\pi(y_t)}$. Weight each such sample by an importance factor that is proportional to its probability under the previously generated density tree.

This approach avoids the danger of generating pairs of poses $\langle x_t^{(i)}, x_{t-1}^{(i)}\rangle$ with zero probability under $p(x_t^{(i)} \mid u_{t-1}, x_{t-1}^{(i)})$. However, it involves an explicit forward sampling phase.

Approach 3 The third approach combines the best of both worlds, in that it avoids the explicit forward-sampling phase of the second approach, but also generates importance factors that are large. In particular, this approach transforms the initial belief $Bel(x_{t-1})$ into a kd-tree. It then generates samples $x_t^{(i)}$ according to

$$\frac{p(y_t \mid x_t)}{\pi(y_t)} \tag{3.6}$$

For each such sample $x_t^{(i)}$, it generates a sample $x_{t-1}^{(i)}$ according to

$$\frac{p(x_t^{(i)} \mid u_{t-1}, x_{t-1})}{\pi(x_t^{(i)} \mid u_{t-1})} \tag{3.7}$$

where

$$\pi(x_t^{(i)} \mid u_{t-1}) \quad = \quad \int p(x_t^{(i)} \mid u_{t-1}, x_{t-1}) \, dx_{t-1} \tag{3.8}$$

Each of these combined samples $\langle x_t^{(i)}, x_{t-1}^{(i)} \rangle$ is, thus, sampled from the joint distribution

$$\frac{p(y_t \mid x_t^{(i)})}{\pi(y_t)} \times \frac{p(x_t^{(i)} \mid u_{t-1}, x_{t-1}^{(i)})}{\pi(x_t^{(i)} \mid u_{t-1})} \tag{3.9}$$

The importance factor is calculated as follows:

$$\begin{aligned}
&\left[\frac{p(y_t \mid x_t^{(i)})}{\pi(y_t)} \times \frac{p(x_t^{(i)} \mid u_{t-1}, x_{t-1}^{(i)})}{\pi(x_t^{(i)} \mid u_{t-1})} \right]^{-1} \frac{p(y_t \mid x_t^{(i)}) \; p(x_t^{(i)} \mid x_{t-1}^{(i)}, u_{t-1}) \; Bel(x_{t-1}^{(i)})}{p(y_t \mid d_{0\ldots t-1})} \\
= \quad & \frac{\pi(y_t) \; \pi(x_t^{(i)} \mid u_{t-1}) \; Bel(x_{t-1}^{(i)})}{p(y_t \mid d_{0\ldots t-1})} \\
\propto \quad & \pi(x_t^{(i)} \mid u_{t-1}) \; Bel(x_{t-1}^{(i)})
\end{aligned} \tag{3.10}$$

where $Bel(x_{t-1}^{(i)})$ is calculated using the kd-tree representing this belief density. The only complication arises from the need to calculate $\pi(x_t^{(i)} \mid u_{t-1})$, which depends on both $x_t^{(i)}$ and u_{t-1}. Luckily, in mobile robot localization, $\pi(x_t^{(i)} \mid u_{t-1})$ can safely be assumed to be a constant—even though this assumption is not valid in general. This leads to the following Monte Carlo algorithm:

1. Sample a pose $x_t^{(i)}$ from a proposal distribution that is proportional to $P(y_t \mid x_t)$.
2. For this $x_t^{(i)}$, sample a pose $x_{t-1}^{(i)}$ from a distribution that is proportional to $P(x_t^{(i)} \mid u_{t-1}, x_{t-1})$.
3. Set the importance factor to a value proportional to the posterior probability of $x_{t-1}^{(i)}$ under the density tree that represents $Bel(x_{t-1})$.

3.3 The Mixture Proposal Distribution

Neither proposal distribution alone—the original distribution q described in (2.4) and the alternative distribution $\bar{q}$ given in (3.1)—is satisfactory. The original MCL proposal distribution fails if the perceptual likelihood is too peaked. The alternative proposal distribution, however, only considers the most recent sensor measurement, hence is prone to failure when the sensors err.

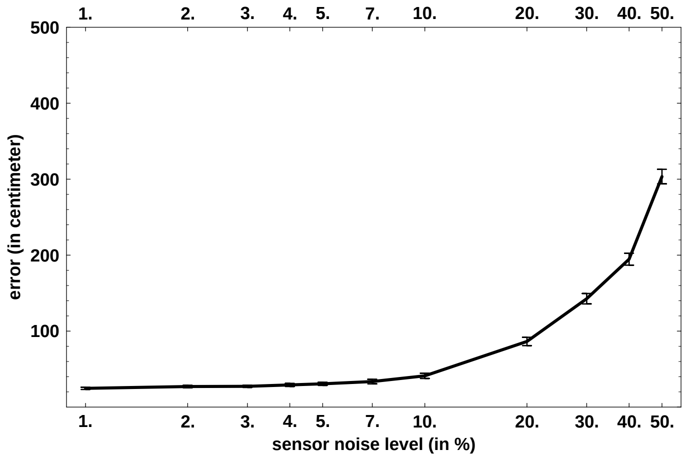

Figure 10: Error of MCL with mixture proposal distribution as a function of the sensor noise. Compare this curve with Figure 9.

A mixture of both proposal distributions gives excellent results:

$$(1-\phi)q \ + \ \phi\bar{q} \tag{3.11}$$

Here ϕ (with $0 \leq \phi \leq 1$) denotes the *mixing ratio* between regular and dual MCL. Figure 10 shows performance results of MCL using this mixture proposal distribution, using a fixed mixing ratio $\phi = 0.1$. All data points are averaged over 1,000 independent experiments. Comparison with Figure 9 suggests that this proposal distribution is uniformly superior to regular MCL, and in certain cases reduces the error by more than an order of magnitude.

These results have been obtained with the third method for calculating importance factors described in the previous section. In our simulation experiments, we found that the second approach yields slightly worse results, but the difference was not significant at the 95% confidence level. As noted above, we have not yet implemented the first approach. In our robot results below, we use the second method for calculating importance factors.

3.4 Robot Results

A series of experiments was conducted, carried out both in simulation and using physical robot hardware, to elucidate the difference between MCL with the standard and the mixture proposal distribution. We found that the modified proposal distribution scales much better to small sample set sizes than conventional MCL. Figure 11 plots the error of both MCL algorithms for different error levels, using $m = 50$ samples only. With 50 samples, the computational load is 0.126% on a 500MHz Pentium Computer—meaning that the

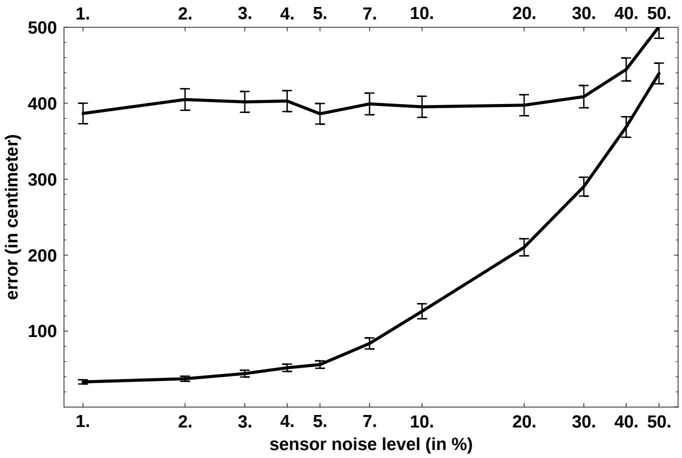

Figure 11: Error of MCL (top curve) and hybrid MCL (bottom curve) with 50 samples (instead of 1,000) for each belief state.

algorithm is approximately 800 times faster than real-time. While MCL with the standard proposal distribution basically fails under this circumstances to track the robot's position, our extended approach gives excellent results, which are only slightly inferior to those obtained with 1,000 sample.

The following experiment evaluates MCL with mixture proposal distribution in the context of the kidnapped robot problem. This MCL algorithm addresses the issue of recovery from a kidnapping, in that it generates samples that are consistent with momentary sensor readings. Our approach was tested using laser range data recorded during the two-week deployment of the robot Minerva. Figure 12 shows part of the map of the museum and the path of the robot used for this experiment. To enforce the kidnapped robot problem, we repeatedly introduced errors into the odometry information. These errors made the robot lose track of its position with probability of 0.01 when advancing one meter. These errors where synthetic; however, they accurately modeled the effect of kidnapping a robot to a random location.

Figure 13 shows comparative results for three different approaches. The error is measured by the percentage of time, during which the estimated position deviates more than 2 meters from the reference position. Obviously, using the mixture proposal distribution yields significantly better results, even if the basic proposal distribution is mixed with 5% random samples (as suggested in (Fox et al. 1999b) to alleviate the kidnapped robot problem). The mixture proposal distribution reduces the error rate of localization by as much as 70% more than MCL if the standard proposal distribution is employed; and 32% when compared to the case where the standard proposal distribution is mixed with a uniform distribution. These results are significant at the 95% confidence level, evaluated over actual robot data.

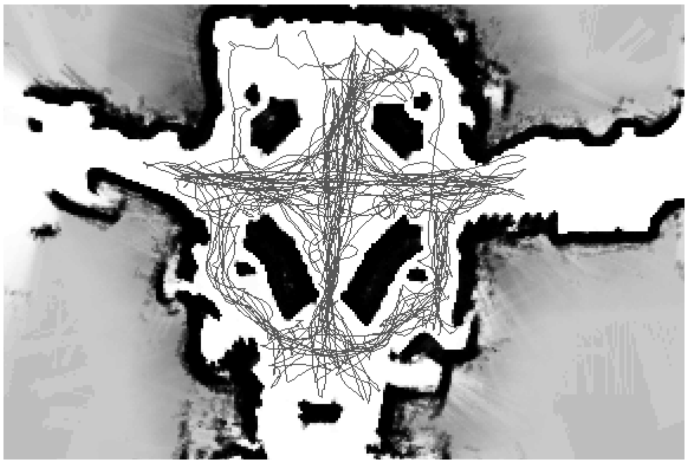

Figure 12: Part of the map of the Smithsonian's Museum of National History, and path of the robot.

We also compared MCL with different proposal distributions in the context of visual localization, using only camera imagery obtained with the robot Minerva during public museum hours. The specific image sequence is of extremely poor quality, as people often intentionally covered the camera with their hand and placed dirt on the lens. Figure 14 shows the localization error obtained when using vision only (calculated using the localization results from the laser as ground truth). The data covers a period of approximately 4,000 seconds, during which MCL processes a total of 20,740 images. After approximately 630 seconds, a drastic error in the robot's odometry leads to a loss of the position (which is an instance of the kidnapped robot problem). As the two curves in Figure 14 illustrate, the regular MCL sampler (dashed curve) is unable to recover from this event, whereas MCL with mixture proposal distribution (solid curve) recovers quickly. These result are not statistically significant in that only a single run is considered, but they confirm our findings with laser range finders. Together, our result suggest that the mixture distribution drastically increases the robustness of the statistical estimator for mobile robot localization.

4 Multi-Robot MCL

4.1 Basic Considerations

The final section of this chapter briefly addresses the multi-robot localization problem. As mentioned in the introduction, multi-robot localization involves a team of robots which simultaneously seek to determine their poses in a known environment. This problem is particularly interesting if robots can

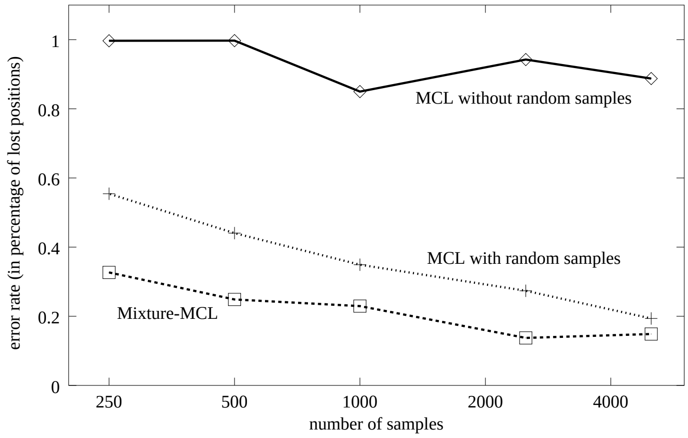

Figure 13: Performance of MCL with the conventional (top curve) and mixture proposal distribution (bottom curve), evaluated for the kidnapped robot problem in the Smithsonian museum. The middle curve reflects the performance of MCL with a small number of random samples added in the resampling step, as suggested in (Fox et al. 2000) as a means to recover from localization failures. The error rate is measured in percentage of time during which the robot lost track of its position.

sense each other during localization. The ability to detect each other can significantly speed up learning; however, it also creates dependencies in the pose estimates of individual robots that pose major challenges for the design of the estimator.

Formally speaking, the multi-robot localization problem is the problem of estimating a posterior density over a product space $X = \bigotimes_{i=1}^{N} X^i$, where X^i describes the position of the i-th robot. Every time a robot senses, it obtains information about the relative poses of all other robots, either by detecting nearby robots, or by not detecting them, which also provides information about other robots' poses. Let $r_t^{i,j}$ denote the random variable that models the detection of robot j by robot i at time t ($i \neq j$). Thus, the variable $r_t^{i,j}$ either takes the value *not detected* or it contains a relative distance and bearing of robot j relative to robot i. The multi-robot localization problem, thus, extends the single robot localization problem by additional observations $r_t^{i,j}$; which are modeled using a time-invariant sensor model $p(x^i \mid r^{i,j}, x^j)$ (time index omitted as above).

The first and most important thing to notice is that the multi-robot localization problem is very hard, and in fact, we only know of a rudimentary solution which, while exhibiting reasonable performance in practice, possesses clear limitations. What makes this problem hard is the fact that the random variables $r_t^{i,j}$ introduce dependencies in the robots' beliefs. Thus, ideally one would like to estimate the posterior over the joint distribution $X = \bigotimes_{i=1}^{N} X^i$. However, such calculations cannot be carried out locally (a desirable prop-

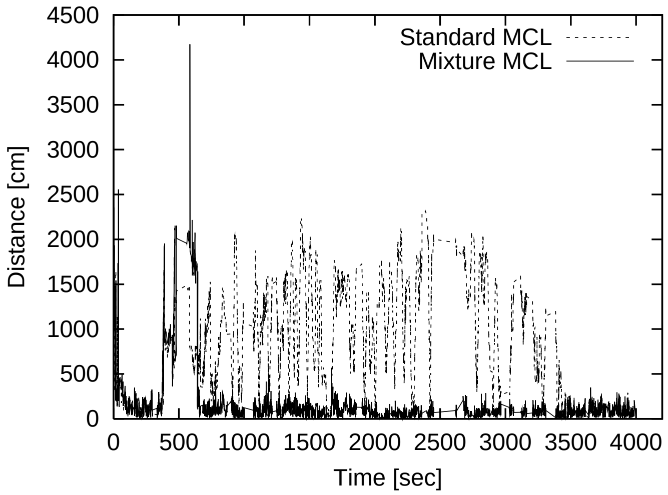

Figure 14: MCL with the standard proposal distribution (dashed curve) compared to MCL with the new mixture distribution (solid line). Shown here is the error for a 4,000-second episode of camera-based localization in the Smithsonian museum.

erty of autonomous robots) and, more importantly, the size of X increases exponentially with the number of robots N. The latter is not much of a problem if all robots are well-localized; however, during global localization large subspaces of X would have to be populated with samples, rendering particle filters hopefully inefficient for this difficult problem.

Our approach basically ignores these non-trivial interdependencies and instead represents the belief at time t by the product of its marginals

$$Bel(x_t) \quad = \quad \prod_{i=1}^{N} Bel(x_t^i) \tag{4.1}$$

Thus, our representation effectively makes a (false) independence assumption—see (Boyen and Koller 1998) for an idea how to overcome this independence assumption while still avoiding the exponential death of the full product space.

When a robot detects another robot, the observation is folded into a robot's current belief, and the result is used to update the belief of the other robots. More specifically, suppose robot i detects robot j at time t. Then j's belief is updated according to

$$Bel(x_t^j) \quad = \quad \int p(x_t^j \mid x_{t-1}^i, r_{t-1}^{i,j})\; Bel(x_{t-1}^i)\; dx_{t-1}^i \;\; Bel(x_{t-1}^j) \tag{4.2}$$

The derivation of this formula is analogous to the derivation of Bayes filters above and can be found in (Fox et al. 2000). By symmetry, the same detection is be used to constrain the i-th robot's position based on the belief of the j-the robot.

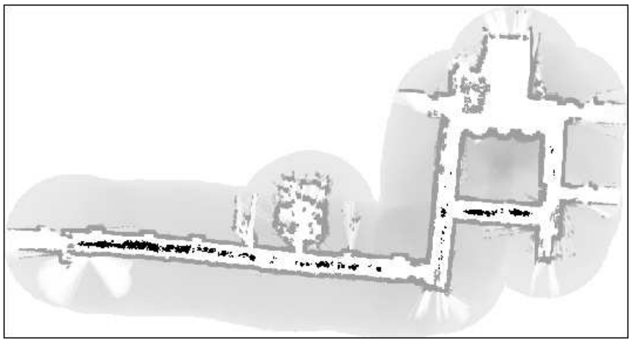

Figure 15: Sample set representing a robot's belief.

Clearly, our update rule assumes independence. Hence, when applied more than once it can lead to repetitive use of the same evidence, which will make our robots more confident than warranted by the data. Unfortunately, we are not aware of a good "fix" to this problem that would maintain the same computational efficiency as our approach. To reduce this effect, our current algorithm only processes positive sightings, that is, the event of *not* seeing another robot has no effect. Additionally, repetitive sightings in short time intervals are ignored. Nevertheless, the occasional transfer from one robot to another can have a substantial effect on each robot's ability to localize.

The implementation of the multi-robot localization algorithm as a distributed particle filter requires some thought. This is because under our factorial representation, each robot maintains its own, local sample set. When one robot detects another, both sample sets have to be synchronized according to Equation (4.2). Note that this equation requires the multiplication of two *densities* which means that we have to establish a correspondence between the individual samples of robot j and the density representing robot i's belief about the position of robot j. However, both of these densities are themselves represented by sample sets, and with probability one no two samples in these sets are the same. To solve this problem, our approach transforms sample sets into density functions using *density trees* (Koller and Fratkina 1998, Moore et al. 1997, Omohundro 1991). Density trees are continuations of sample sets which approximate the underlying density using a variable-resolution piecewise constant density.

Figure 16 shows such a tree, which corresponds to a robot's estimate of another robot's location. Together with Figure 15, it shows a map of our testing environment along with a sample set obtained during global localization. The resolution of the tree is a function of the densities of the samples: the more samples exist in a region of space, the more fine-grained the tree representation. The tree enables us to integrate the detection into the sample set of the detected robot using importance sampling for each individual

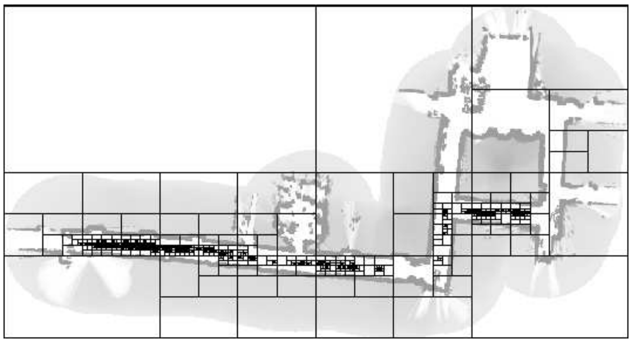

Figure 16: Tree representation extracted from the sample set.

sample $\langle x, w \rangle$:

$$w \quad \longleftarrow \quad \alpha \int p(x_t^j \mid x_{t-1}^i, r_{t-1}^i) \; Bel(x_{t-1}^i) \; dx_{t-1}^i \tag{4.3}$$

4.2 Robot Results

Multi-Robot MCL has been tested using two RWI Pioneer robots, equipped with a camera and a laser range finder for detection (see (Fox et al. 2000) for details). In particular, our implementation detects robots visually, and uses a laser range finder to determine the relative distance and bearing. The perceptual models $p(x^i \mid r^{i,j}, x^j)$ were estimated from data collected in a separate training phase, where the exact location of each robot was known. After training, the mean error of the distance estimation was 48.26 cm, and the mean angular error was 2.2 degree. Additionally, there was a 6.9% chance of erroneously detecting a robot (false positive).

Figure 17 plots the localization error as a function of time, averaged over ten experiments involving physical robots in the environment shown in Figure 15. The ability of the robots to detect each other clearly reduces the time required for global localization. Obviously, the overuse of evidence, while theoretically present, appears not to harm the robots' ability to localize themselves. We attribute this finding to the fact that our multi-robot MCL is highly selective when incorporating relative information. These findings were confirmed in systematic simulation experiments (Fox et al. 2000) involving larger groups of robots in a range of different environments.

5 Conclusion

This chapter has surveyed a family of particle filters for mobile robot localization, commonly known as *Monte Carlo localization* (MCL). MCL algorithms provide efficient and robust solutions for a range of mobile robot localization

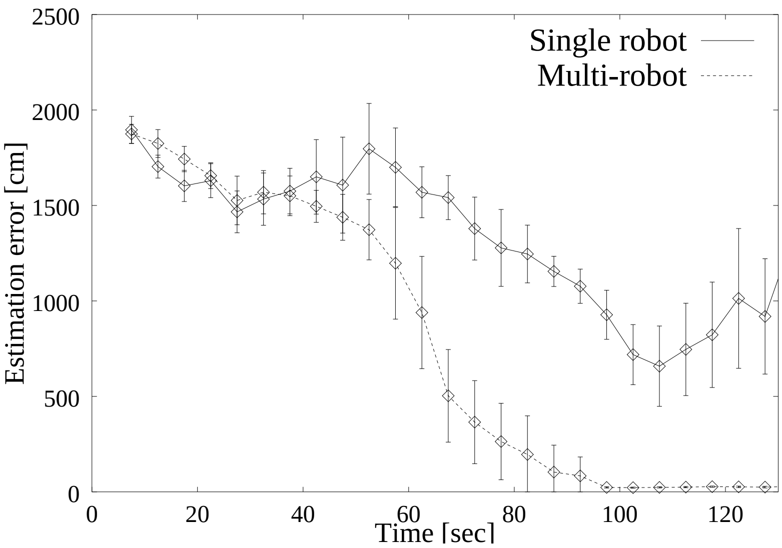

Figure 17: Empirical results comparing single robot MCL and multi robot MCL.

problems, such as position tracking, global localization, robot kidnapping, and multi-robot localization.

This chapter investigated three variants of the basic algorithm: The basic MCL algorithm, which has been applied with great success to global localization and tracking, followed by an extension that uses a more sensible proposal distribution, which overcomes certain limitations of MCL such as poor performance when sensors are too accurate, and suboptimal recovery from robot kidnapping. Finally, the paper proposed an extension to multi-robot localization, where a distributed factorial representation was employed to estimate the joint posterior.

For all these algorithms, we obtained favorable results in practice. In fact, an elaborate experimental comparison with our previous best method, a version of Markov localization that uses fine-grained grid representations (Fox at al. 1999a), showed that MCL is an order of magnitude more efficient and accurate than the grid-based approach.

The derivation of all these algorithms is based on a collection of independence assumptions, ranging from a static world assumption to the assumption that the joint belief space of multiple robots can be factorized into independent components that are updated locally on each robot. Clearly, in most application domains all of these independence assumptions are violated. Robot environments, for example, are rarely static. Relaxing those assumptions is a key goal of current research, with enormous potential benefits for robot practitioners.

Acknowledgments

The authors like to thank Hannes Kruppa for his contributions to the multi-robot MCL approach, and him and the members of CMU's Robot Learning Lab for stimulating discussions that have influenced this research. We are also indebted to Nando de Freitas and Arnaud Doucet for their insightful comments on an earlier draft of this paper that substantially improved the presentation of the material.

This research is sponsored by the National Science Foundation (CAREER grant number IIS-9876136 and regular grant number IIS-9877033), and by DARPA-ATO via TACOM (contract number DAAE07-98-C-L032) and DARPA-ISO via Rome Labs (contract number F30602-98-2-0137), which is gratefully acknowledged. The views and conclusions contained in this document are those of the author and should not be interpreted as necessarily representing official policies or endorsements, either expressed or implied, of the United States Government or any of the sponsoring institutions.

References

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

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

Boyen, X. and Koller, D. (1998). Tractable inference for complex stochastic processes, *Proceedings of Uncertainty in Artificial Intelligence*, Madison, WI, pp. 33–42.

Burgard, W., Fox, D., Hennig, D. and Schmidt, T. (1996). Estimating the absolute position of a mobile robot using position probability grids, *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, AAAI, AAAI Press/MIT Press, Menlo Park.

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

Cox, I. and Wilfong, G. (eds) (1990). *Autonomous Robot Vehicles*, Springer Verlag.

Dean, T. L. and Boddy, M. (1988). An analysis of time-dependent planning, *Proceeding of Seventh National Conference on Artificial Intelligence AAAI-92*, AAAI, AAAI Press/The MIT Press, Menlo Park, CA, pp. 49–54.

Dellaert, F., Burgard, W., Fox, D. and Thrun, S. (1999a). Using the condensation algorithm for robust, vision-based mobile robot localization, *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, IEEE, Fort Collins, CO.

Dellaert, F., Fox, D., Burgard, W. and Thrun, S. (1999b). Monte carlo localization for mobile robots, *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Dellaert, F., Thorpe, C. and Thrun, S. (1999c). Mosaicing a large number of widely dispersed, noisy, and distorted images: A bayesian approach, *Technical Report CMU-RI-TR-99-34*, Carnegie Mellon University, Pittsburgh, PA.

Doucet, A. (1998). On sequential simulation-based methods for bayesian filtering, *Technical Report CUED/F-INFENG/TR 310*, Cambridge University, Department of Engineering, Cambridge, UK.

Engelson, S. and McDermott, D. (1992). Error correction in mobile robot map learning, *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, Nice, France, pp. 2555–2560.

Fox, D., Burgard, W. and Thrun, S. (1998). Active Markov localization for mobile robots, *Robotics and Autonomous Systems* **25**: 195–207.

Fox, D., Burgard, W. and Thrun, S. (1999a). Markov localization for mobile robots in dynamic environments, *Journal of Artificial Intelligence Research* **11**: 391–427.

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

Fox, D., Burgard, W., Kruppa, H. and Thrun, S. (2000). A probabilistic approach to collaborative multi-robot localization, *Autonomous Robots*: 8(3).

Fukuda, T., Ito, S., Oota, N., Arai, F., Abe, Y., Tanake, K. and Tanaka, Y. (1993). Navigation system based on ceiling landmark recognition for autonomous mobile robot, *Proc. Int'l Conference on Industrial Electronics Control and Instrumentation (IECON'93)*, Vol. 1, pp. 1466 – 1471.

Fung, R. and Del Favero, B. (1994). Backward simulation in bayesian networks, *Proceedings of the Conference on Uncertainty in AI (UAI)*.

Hinkel, R. and Knieriemen, T. (1988). Environment perception with a laser radar in a fast moving robot, *Proceedings of Symposium on Robot Control*, Karlsruhe, Germany, pp. 68.1–68.7.

Kaelbling, L., Cassandra, A. and Kurien, J. (1996). Acting under uncertainty: Discrete bayesian models for mobile-robot navigation, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems.*

Kitagawa, G. (1996). Monte carlo filter and smoother for non-gaussian nonlinear state space models, *Journal of Computational and Graphical Statistics* **5**(1): 1–25.

Koenig, S. and Simmons, R. (1996). Passive distance learning for robot navigation, *in* L. Saitta (ed.), *Proceedings of the Thirteenth International Conference on Machine Learning.*

Koller, D. and Fratkina, R. (1998). Using learning for approximation in stochastic processes, *Proc. of the International Conference on Machine Learning (ICML).*

Kortenkamp, D., Bonasso, R. and Murphy, R. (eds) (1998). *AI-based Mobile Robots: Case studies of successful robot systems*, MIT Press, Cambridge, MA.

Lenser, S. and Veloso, M. (2000). Sensor resetting localization for poorly modelled mobile robots, *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, San Francisco, CA.

Leonard, J., Durrant-Whyte, H. and Cox, I. (1992). Dynamic map building for an autonomous mobile robot, *International Journal of Robotics Research* **11**(4): 89–96.

Liu, J. and Chen, R. (1998). Sequential monte carlo methods for dynamic systems, *Journal of the American Statistical Association*: 93.

Moore, A., Schneider, J. and Deng, K. (1997). Efficient locally weighted polynomial regression predictions, *Proc. of the International Conference on Machine Learning (ICML).*

Moore, A. W. (1990). *Efficient Memory-based Learning for Robot Control*, PhD thesis, Trinity Hall, University of Cambridge, England.

Nourbakhsh, I., Powers, R. and Birchfield, S. (1995). DERVISH an office-navigating robot, *AI Magazine* **16**(2): 53–60.

Omohundro, S. M. (1991). Bumptrees for efficient function, constraint, and classification learning, *in* R. P. Lippmann, J. E. Moody and D. S. Touretzky (eds), *Advances in Neural Information Processing Systems 3*, Morgan Kaufmann.

Pitt, M. and Shephard, N. (1999). Filtering via simulation: auxiliary particle filter, *Journal of the American Statistical Association*. Forthcoming.

Rencken, W. (1993). Concurrent localisation and map building for mobile robots using ultrasonic sensors, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Yokohama, Japan, pp. 2129–2197.

Simmons, R. and Koenig, S. (1995). Probabilistic robot navigation in partially observable environments, *Proceedings of IJCAI-95*, IJCAI, Inc., Montreal, Canada, pp. 1080–1087.

Simmons, R., Goodwin, R., Haigh, K., Koenig, S. and O'Sullivan, J. (1997). A layered architecture for office delivery robots, *Proceedings of the First International Conference on Autonomous Agents*, Marina del Rey, CA.

Tanner, M. (1993). *Tools for Statistical Inference*, Springer Verlag, New York. 2nd edition.

Thrun, S. (1998). Bayesian landmark learning for mobile robot localization, *Machine Learning* **33**(1): 41–76.

Thrun, S., Bennewitz, M., Burgard, W., Cremers, A., Dellaert, F., Fox, D., Hähnel, D., Rosenberg, C., Roy, N., Schulte, J. and Schulz, D. (1999). MINERVA: A second generation mobile tour-guide robot, *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Thrun, S., Fox, D. and Burgard, W. (2000). Monte carlo localization with mixture proposal distributions, *Proceedings of the AAAI National Conference on Artificial Intelligence*, AAAI, Austin, TX.

Weiß, G., Wetzler, C. and von Puttkamer, E. (1994). Keeping track of position and orientation of moving indoor systems by correlation of range-finder scans, *Proceedings of the International Conference on Intelligent Robots and Systems*, pp. 595–601.

Zilberstein, S. and Russell, S. (1995). Approximate reasoning using anytime algorithms., *in* S. Natarajan (ed.), *Imprecise and Approximate Computation*, Kluwer Academic Publishers, Dordrecht.