Robust Monte Carlo Localization for Mobile Robots

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

April 2000

CMU-CS-00-125

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

This research is sponsored by the National Science Foundation (and 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.

Abstract

Mobile robot localization is the problem of determining a robot's pose from sensor data. Monte Carlo Localization is a family of algorithms for localization based on particle filters, which are approximate Bayes filters that use random samples for posterior estimation. Recently, they have been applied with great success for robot localization. Unfortunately, regular particle filters perform poorly in certain situations. Mixture-MCL, the algorithm described here, overcomes these problems by using a "dual" sampler, integrating two complimentary ways of generating samples in the estimation. To apply this algorithm for mobile robot localization, a kd-tree is learned from data that permits fast dual sampling. Systematic empirical results obtained using data collected in crowded public places illustrate superior performance, robustness, and efficiency, when compared to other state-of-the-art localization algorithms.

Keywords: Mobile Robots, Particle Filters, Localization, Kernel Density Trees, Sampling, Monte Carlo Algorithms, Randomize Algorithms, Bayes Filters

1 Introduction

Mobile robot localization is the problem of estimating a robot's pose (location, orientation) relative to its environment. The localization problem is a key problem in mobile robotics, since it plays a pivotal role in various successful mobile robot systems (see e.g., [7, 22, 27, 39, 50, 53, 61] and various chapters in [2, 36]). Occasionally, it has been referred to as " the most fundamental problem to providing a mobile robot with autonomous capabilities" [6].

The mobile robot localization problem comes in many different flavors [2, 21]. The most simple localization problem—which has received by far the most attention in the literature—is *position tracking* [2, 62, 52, 61]. Here the initial robot pose is known, and the problem is to compensate small, incremental errors in a robot's odometry. Algorithms for position tracking often make restrictive assumptions on the size of the error and the shape of the robot's uncertainty, required by a range of existing localization algorithms. More challenging is the *global localization problem* [4], 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 error in the robot's estimate cannot be assumed to be small. Here a robot should be able to handle multiple, distinct hypotheses. Even more difficult is the *kidnapped robot problem* [17], in which a well-localized robot is tele-ported to some other place 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 from catastrophic localization failures. Finally, all these problems are particularly hard in dynamic environments, e.g., if robots operate in the proximity of people who corrupt the robot's sensor measurements [3, 59].

The vast majority of existing algorithms only address the position tracking problem (see e.g., the review [2]). The nature of small, incremental errors makes algorithms such as Kalman filters [25, 31, 41, 55] applicable, which have been successfully applied in a range of fielded systems (e.g., [38]). Kalman filters estimate posterior distributions of robot poses conditioned on sensor data. Exploiting a range of restrictive assumptions—such as Gaussian-distributed noise and Gaussian-distributed initial uncertainty—they represent posteriors by Gaussians, exploiting an elegant and highly efficient algorithm for incorporating new sensor data. However, the restrictive nature of belief representations makes them inapplicable to global localization problems.

Possibly the most powerful family of global localization algorithm to date is *Markov localization*—a generalization of Kalman filters—which has been proposed by a range of researchers [5, 21, 26, 30, 34, 47, 48, 54, 58]. Just like Kalman filters, these algorithms estimate posterior distributions over robot poses. The key distinguishing factor is that these distributions are approximated by piecewise constant functions instead of Gaussians, enabling them to represent highly multi-modal distributions. As a result, these algorithms have been applied successfully to global localization problems. However, the piecewise constant representation can impose a significant computational burden, especially if one is interested in high resolution. To overcome this limitation, researchers have proposed selective updating algorithms [21] and tree-based representations that dynamically change their resolution [4]. While these algorithms work well in real-time, they nevertheless suffer two limitations: First, they are computationally very expensive, and second, the accuracy is limited by the resolution of the approximation, which typically lacks behind that of Kalman filters when a robot is well-localized (see [24] for an empirical comparison).

Monte Carlo localization (MCL) [10, 18] is a novel mobile robot localization algorithm which overcomes many of these problems; in particular, it solves the global localization and kidnapped robot problem, and it is an order of magnitude more efficient and accurate than the best existing Markov localization algorithm. MCL and Kalman filters share the same mathematical foundations. Just like Kalman filters and Markov localization, the MCL algorithm computes approximate posteriors over robot poses conditioned on sensor data. The key idea of MCL, however, is to approximate these densities through samples (also called: *particles*), drawn according to the posterior distribution over robot poses. In other words, rather than approximating posteriors in close form, which typically requires some parametric model such as Gaussians, MCL simply represents the posteriors by a random collection of weighted particles which approximates the desired distribution [51].

The idea of estimating state recursively using particles is not new, although most work on this topic is very recent. In the statistical literature, it is known as *particle filters* [14, 15, 40, 49], and recently computer vision researchers have proposed the same algorithm under the name of *condensation algorithm* [29]. MCL, which applies this idea to mobile robot localization, has originally been proposed in [10, 18]. Within the context of localization, the particle representation has a range of advantages over previous approaches:

1. As we will see below, particles focus computational resources in areas that are most relevant, by sampling in proportion to the posterior likelihood. This results in improved efficiency over previous methods.

2. Particle filters are universal density approximators, weakening the restrictive assumptions on the shape of the posterior density when compared to previous approaches.

3. Particle filters are easily implemented as *any-time algorithms* [8, 63], which are algorithms that can output an answer at any time, but where the quality of the answer depends on the time spent computing it. By controlling the number of samples online, particle filters can adapt to the available computational resources. The same code can, thus, be executed on computers with vastly different speed without modification of the basic code.

4. Finally, participle filters are surprisingly easy to implement, which makes them an attractive paradigm for mobile robot localization. Consequently, it has already been adopted by several research teams [13, 37], who have extended the basic paradigm in interesting new ways.

However, there are pitfalls, too, arising from the stochastic nature of the approximation. Some of these pitfalls are obvious: For example, if the sample set size is small, a well-localized robot might lose track of its position just because MCL fails to generate a sample in the right location. The regular MCL algorithm is also unfit for the kidnapped robot problem, since there might be no surviving samples nearby the robot's new pose after it has been kidnapped. Somewhat counter-intuitive is the fact that the basic algorithm degrades poorly when sensors are too accurate. In the extreme, regular MCL will fail with perfect, noise-free sensors. All these problems possess fixes, such as augmenting the sample set through uniformly distributed samples [18], generating samples consistent with the most recent sensor reading [37], or assuming a higher level of sensor noise than actually is the case. While these extensions yield improved performance, they are mathematically questionable. In particular, these extensions do not approximate the correct density; which makes the interpretation of their results difficult.

This paper presents an extension of MCL closely related to [37], called *MCL with mixture proposal distribution* (in short: *Mixture-MCL*). Mixture-MCL addresses all these problems in a way that is mathematically sound. The key idea is to modify the way samples are generated in MCL (and particle filters). Mixture-MCL combines regular MCL sampling with a "dual" of MCL, which basically inverts MCL's sampling process. More specifically, while regular MCL first guesses a new pose using odometry, then uses the sensor measurements to assess the "importance" of this sample, dual MCL guesses poses using the most recent sensor measurement, then uses odometry to assess the compliance of this guess with the robot's previous belief and odometry data. Neither of these sampling methodologies alone is sufficient; in combination, however, they work very well. In particular, Mixture-MCL works well if the sample set size is small (e.g., 50 samples), it recovers faster from robot kidnapping than any previous variation of MCL, and it also works well when sensor models are too narrow for regular MCL. Thus, from a performance point of view, Mixture-MCL is uniformly superior to regular MCL and particle filters.

The key *disadvantage* of Mixture-MCL is a requirement for a sensor model that permits fast sampling of poses. While in certain cases, such a model can trivially be obtained, in others, such as the navigation domains studied here and in [21], it cannot. To overcome this difficulty, our approach uses sufficient statistics and density trees to learn a sampling model from data. More specifically, in a pre-processing phase sensor readings are mapped into a set of discriminating features, and potential robot poses are then drawn randomly using trees generated. Once the tree has been constructed, dual sampling can be done very efficiently.

Empirical results are presented both using a robot simulator and data collected by a physical robot. Simulation is used since it allows us to systematically vary key parameters such as the sensor noise, thereby enabling us to characterize the degradation of MCL in extreme situations. To verify the experimental findings obtained with simulation, Mixture-MCL is also applied to two extensive data set gathered in a public museum (a Smithsonian museum in Washington, DC), where during a two-week period in the Fall 1998 our mobile robot Minerva gave interactive tours to thousands of visitors [59]. One of the data set comprises laser range data, where a metric map of the museum is used for localization [59]. The other data set contains image segments recorded with a camera pointed towards the museum's ceiling, using a large-scale ceiling mosaic for cross-referencing the robot's position [11]. In the past, these data have been used as benchmark, since localization in this crowded and feature-impoverished museum is a challenging problem. Our experiments suggest that our new MCL algorithm is highly efficient and accurate, and it works well in cases where regular MCL (and other localization algorithms) fail.

The remainder of this article is organized as follows. Section 2 introduces the regular MCL algorithm, which includes a mathematical derivation from first principles and an experimental characterization of MCL in practice. The section also compares MCL with grid-based Markov localization [21], one of the most powerful localization algorithms capable of global localization. Section 3 presents examples where regular MCL performs poorly, along with a brief analysis of the underlying causes. This section is followed by the description of dual MCL and Mixture-MCL in Section 4. Section 5 describes our approach to learning trees for efficient sampling in dual MCL. Experimental results are given in Section 6. Finally, the paper is concluded by a description of related work in Section 7, and a discussion of the strengths and weaknesses of Mixture-MCL (Section 8).

2 Monte Carlo Localization

2.1 Bayes Filtering

MCL is a recursive Bayes filter that estimates the posterior distribution of robot poses conditioned on sensor data. Bayes filters address the problem of estimating the state x of a dynamical system (partially observable Markov chain) 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 Cartesian x-y 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 Markov assumption will be made more explicit below.

The key idea of Bayes filtering is to estimate a probability density over the state space conditioned on the data. This posterior is typically called the *belief* and is denoted

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

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*, which carries information about robot motion. Denoting the former by o (for *observation*) and the latter by a (for *action*), we have

$$Bel(x_t) \quad = \quad p(x_t|o_t, a_{t-1}, o_{t-1}, a_{t-2} \ldots, o_0) \tag{1}$$

Without loss of generality, we assume that observations and actions arrive in an alternating sequence. Notice that we will use a_{t-1} to refer to the odometry reading that measures the motion that occurred in the time interval $[t-1; t]$, to illustrate that the motion is the result of the control action asserted at time $t-1$.

Bayes filters estimate the belief *recursively*. The *initial* belief characterizes the *initial* knowledge about the system state. In the absence of such, it is typically initialized by a *uniform distribution* over the state space. In mobile robot localization, a uniform initial distribution corresponds to the global localization problem, where the initial robot pose is unknown.

To derive a recursive update equation, we observe that Expression (1) can be transformed by Bayes rule to

$$Bel(x_t) \quad = \quad \eta \; p(o_t|x_t, a_{t-1}, \ldots, o_0) \; p(x_t|, a_{t-1}, \ldots, o_0) \tag{2}$$

where η is a normalization constant. As noticed above, Bayes filters rest on the assumption that future data is independent of past data given knowledge of the current state—an assumption typically referred to as the *Markov assumption*. Put mathematically, the Markov assumption implies

$$p(o_t|x_t, a_{t-1}, \ldots, o_0) \quad = \quad p(o_t|x_t) \tag{3}$$

and hence our target expression (2) can be simplified to:

$$Bel(x_t) \quad = \quad \eta \; p(o_t|x_t) \; p(x_t|a_{t-1}, \ldots, o_0)$$

We will now expand the rightmost term by integrating over the state at time $t-1$:

$$\begin{aligned} & Bel(x_t) \\ = \quad & \eta \; p(o_t|x_t) \int p(x_t|x_{t-1}, a_{t-1}, \ldots, o_0) \; p(x_{t-1}|a_{t-1}, \ldots, o_0) \; dx_{t-1} \end{aligned} \tag{4}$$

Again, we can exploit the Markov assumption to simplify $p(x_t|x_{t-1}, a_{t-1}, \ldots, o_0)$:

$$p(x_t|x_{t-1}, a_{t-1}, \ldots, o_0) \quad = \quad p(x_t|x_{t-1}, a_{t-1}) \tag{5}$$

which gives us the following expression:

$$Bel(x_t) \quad = \quad \eta \; p(o_t|x_t) \int p(x_t|x_{t-1}, a_{t-1}) \; p(x_{t-1}|a_{t-1}, \ldots, o_0) \; dx_{t-1} \tag{6}$$

Substituting the basic definition of the belief Bel back into (6), we obtain the important recursive equation

$$Bel(x_t) \quad = \quad \eta \; p(o_t|x_t) \int p(x_t|x_{t-1}, a_{t-1}) \; Bel(x_{t-1}) \; dx_{t-1} \tag{7}$$

This equation is the recursive update equation in Bayes filters. Together with the initial belief, it defines a recursive estimator for the state of a partially observable system. This equation is of central importance in this paper, as it is the basis for various MCL algorithms studied here.

To implement (7), one needs to know two conditional densities: the probability $p(x_t|x_{t-1}, a_{t-1})$, which we will refer to as *next state density* or simply *motion model*, and the density $p(o_t|x_t)$, which we will call *perceptual model* or *sensor model*. Both models are typically *stationary* (also called: *time-invariant*), that is, they do not depend on the specific time t. This stationarity allows us to simplify the notation by denoting these models $p(x|x', a)$, and $p(o|x)$, respectively.

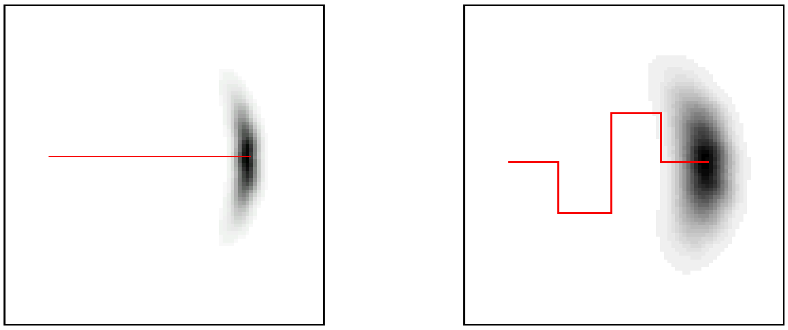


Figure 1: The density $p(x|x', a)$ after moving 40 meter (left diagram) and 80 meter (right diagram). The darker a pose, the more likely it is.

2.2 Probabilistic Models for Localization

The nature of the models $p(x|x', a)$ and $p(o|x)$ depends on the specific estimation problem. In mobile robot localization, which is the focus of this paper, both models are relatively straightforward and can be implemented in a few lines of code. The specific probabilistic models used in our implementation have been described in depth elsewhere [21]; therefore, we will only provide an informal account.

The motion model, $p(x|x', a)$, is a probabilistic generalization of robot kinematics [7]. More specifically, recall that x and x' are poses. For a robot operating in the plane, a pose is a three-dimensional variable, which comprises a robot's two-dimensional Cartesian coordinates and its heading direction (orientation). The value of a may be an odometry reading or a control command, both of which characterize the change of pose. In robotics, change of pose is called *kinematics*. The conventional kinematic equations, however, describe only the *expected* pose x that an ideal, noise-free robot would attain starting at x', and after moving as specified by a. Of course, physical robot motion is erroneous; thus, the pose x is uncertain. To account for this inherent uncertainty, the probabilistic motion model $p(x|x', a)$ describes a posterior density over possible successor x. Noise is typically modeled by zero centered, Gaussian noise that is added to the translation and rotation components in the odometry measurements [21]. Thus, $p(x|x', a)$ generalizes exact mobile robot kinematics typically described in robot textbooks [7].

Figure 1 shows two examples of $p(x|x', a)$. In both examples, the initial pose x' is shown on the left, and the solid line depicts the odometry data as measured by the robot. The grayly shaded area on the right depicts the density $p(x|x', a)$: the darker a pose, the more likely it is. A comparison of both diagrams reveals that the margin of uncertainty depends on the overall motion: Even though the *expected* posterior poses are the same for both motion segments, the uncertainty in the right diagram is larger due to the longer overall distance traversed by the robot.

For the MCL algorithm described further below, one does not need a closed-form description of the motion model $p(x|x', a)$. Instead, a *sampling model* of $p(x|x', a)$ suffices. A sampling model is a routine that accepts x' and a as an input and generates random poses x distributed according to $p(x|x', a)$. Sampling models are usually easier to code than rou-

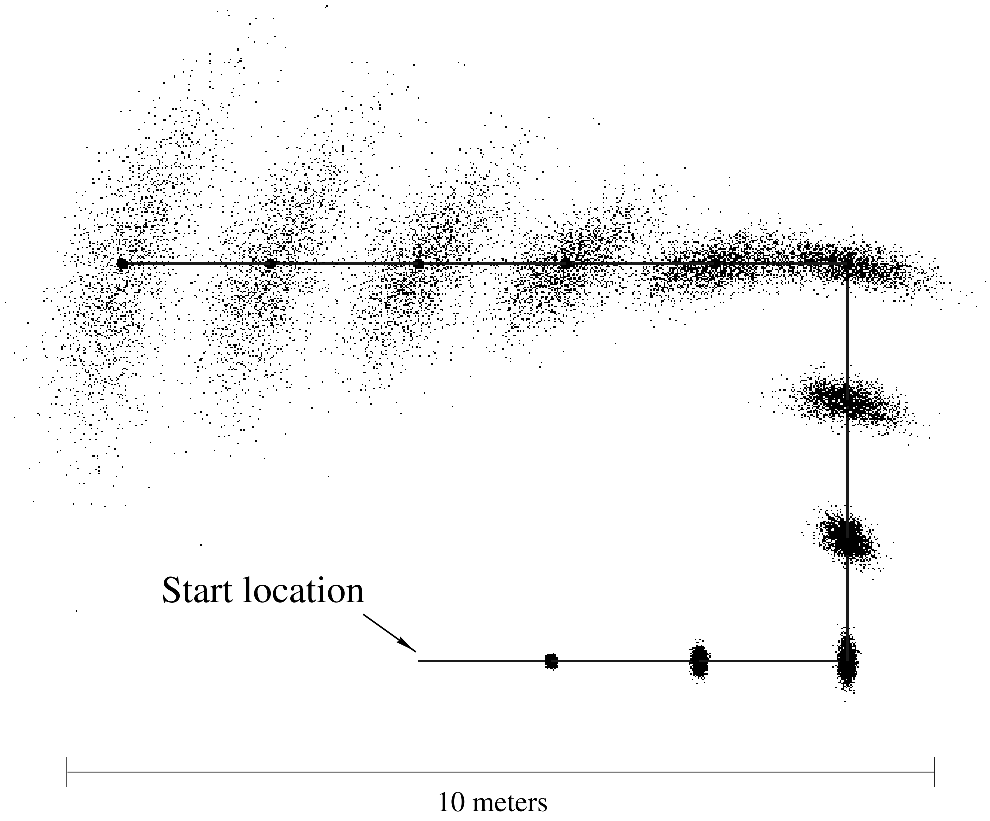


Figure 2: Sampling-based approximation of the position belief for a robot that only measures odometry. The solid line displays the actions, and the samples represent the robot's belief at different points in time.

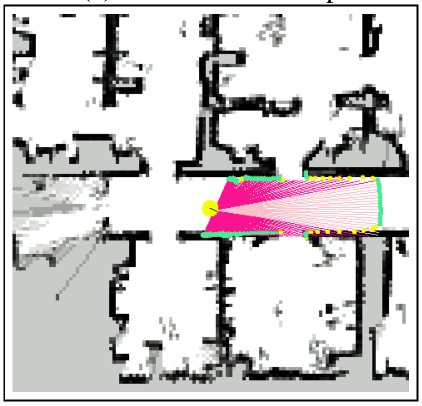
tines that compute densities in closed form. Figure 2 shows a sample model of $p(x|x', a)$, applied to a sequence of odometry measurements indicated by the solid line. As is easy to be seen, the sequence of particle sets approximates the densities of a robot that only measures odometry.

Let us now turn our attention to the perceptual model, $p(o|x)$. Mobile robots are commonly equipped with range finders, such as ultrasonic transducers (sonar sensors) or laser range finders. Figure 3a shows an example of a laser range scan, obtained with an RWI B21 robot in an environment whose approximate shape is also shown in Figure 3a. Notice that the range finder emits a plateau of light that covers a horizontal 180 degree range, for which it measures the distance to the nearest objects.

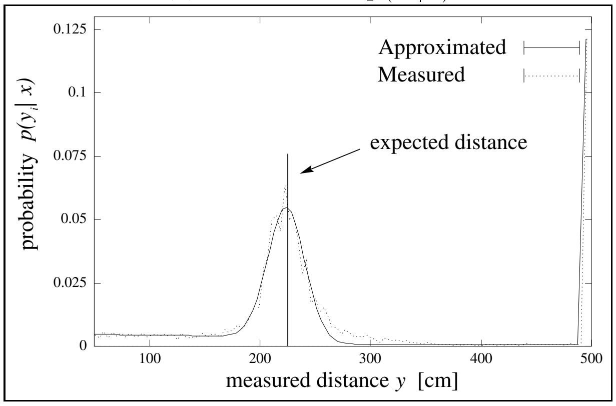
For range finders, we decompose the problem of computing $p(o|x)$ into three parts:

1. the computation of the value a noise-free sensor would generate;
2. the modeling of sensor noise; and
3. the integration of many individual sensor beams into a single density value.

(a) laser scan and map



(b) sensor model $p(o_i|x)$



(c) probability distribution for different poses

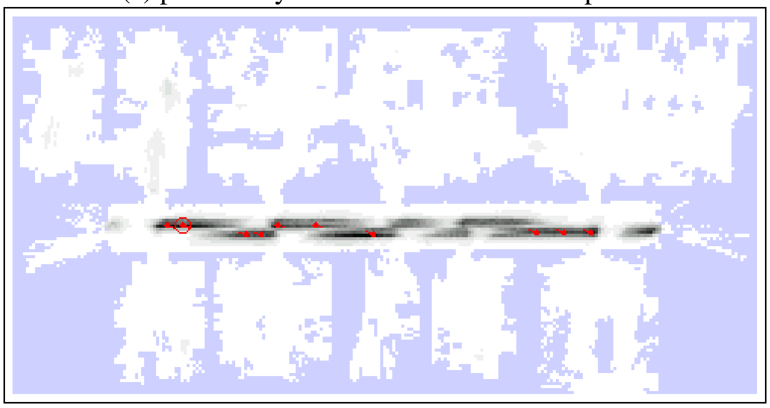


Figure 3: (a) Laser range scan, projected into a map. (b) The density $p(o|x)$, where the peak corresponds to the distance an ideal, noise-free sensor would measure. (c) $p(o|x)$ for the scan shown in (a). Based on a single sensor scan, the robot assigns high likelihood for being somewhere in the main corridor.

Assume the robot's pose is x, and let o_i denote an individual sensor beam with bearing α_i relative to the robot. Let $g(x, \alpha_i)$ denote the measurement of an ideal, noise-free sensor whose relative bearing is α_i. Since we assume that the robot is given a map of the environment such as the one shown in Figure 3a, $g(x, \alpha_i)$ can be computed using *ray tracing* [43]. It is common practice to assume that this "expected" distance $g(x, \alpha_i)$ is a sufficient statistic for the probability $p(o_i|x)$, that is

$$p(o_i|x) \quad = \quad p(o_i|g(x, \alpha_i)) \tag{8}$$

The exact density $p(o_i|x)$ is shown in Figure 3b. This density is a mixture of three densities: a Gaussian centered at $g(x, \alpha_i)$ that models the event of measuring the correct distance with small added Gaussian noise, an exponential density that models random readings as often caused by people, and a discrete large probability (mathematically modeled by a narrow uniform density) that models max-range measurements, which frequently occur when a range sensor fails to detect an object. The specific parameters of the density in Figure 3b have been estimated using an algorithm similar to EM [12, 44], which starts with a crude initial model and iteratively labels several million measurements collected in the Smithsonian museum, while refining the model. A smoothed version of these data are also shown in Figure 3b, illustrating that our probabilistic model is highly accurate.

Finally, the individual density values $p(o_i|x)$ are integrated multiplicatively, assuming conditional independence between the individual measurements:

$$p(o|x) \quad = \quad \prod_i p(o_i|x) \tag{9}$$

Clearly, this conditional independence can be violated in the presence of people (which often block more than one sensor beam). In such cases, it might be advisable to subsample the sensor readings and use a reduced set for localization [21].

Figure 3c depicts $p(o|x)$ for the sensor scan shown in Figure 3a and the map shown in grey in Figure 3c. Most of the probability mass is found in the corridor region, which reflects the fact that the specific sensor measurement is much more likely to have originated in the corridor than in one of the rooms. The probability mass roughly lies on two bands, and otherwise is spread through most of the corridor. Coincidentally, the density shown in Figure 3c is equivalent to the posterior belief of a globally localizing robot after perceiving only one sensor measurement.

2.3 Particle Approximation

If the state space is continuous, as is the case in mobile robot localization, implementing (7) 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 $Bel(x)$:

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

Here each $x^{(i)}$ is a sample (a hypothesized state), and $w^{(i)}$ are non-negative numerical factors called *importance factors*, which we assume to sum up to one. As the name suggests, the importance factors determine the weight (=importance) of each sample [51].

The initial set of samples represents the initial knowledge $Bel(x_0)$ about the state of the dynamical system. 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.

1. Sample $x_{t-1}^{(i)} \sim Bel(x_{t-1})$ from the (weighted) sample set representing $Bel(x_{t-1})$. Each such particle $x_{t-1}^{(i)}$ is distributed according to the belief distribution $Bel(x_{t-1})$.

2. Sample $x_t^{(i)} \sim p(x_t|x_{t-1}^{(i)}, a_{t-1})$. Obviously, the pair $\langle x_t^{(i)}, x_{t-1}^{(i)} \rangle$ is distributed according to the product distribution

$$q_t \quad := \quad p(x_t|x_{t-1}, a_{t-1}) \times Bel(x_{t-1}) \tag{10}$$

which is commonly called the *proposal distribution*.

3. To offset the difference between the proposal distribution and the desired distribution (c.f., Equation (7))

$$\eta\ p(o_t|x_t)p(x_t|x_{t-1}, a_{t-1})Bel(x_{t-1}) \tag{11}$$

the sample is weighted by the quotient

$$\begin{aligned} w^{(i)} &= \frac{\eta\ p(o_t|x_t^{(i)})p(x_t^{(i)}|x_{t-1}^{(i)}, a_{t-1})Bel(x_{t-1}^{(i)})}{Bel(x_{t-1}^{(i)})\ p(x_t^{(i)}|x_{t-1}^{(i)}, a_{t-1})} \\ &\propto p(o_t|x_t^{(i)}) \end{aligned} \tag{12}$$

$w^{(i)}$ denotes the new (non-normalized) importance factor for the particle $x_t^{(i)}$.

The sampling routine is repeated m times, producing a set of m weighted samples $x_t^{(i)}$. Afterwards, the importance factors are normalized so that they sum up to 1 (hence define a discrete probability distribution).

Table 1 summarizes the MCL algorithm. It is known [57] that under mild assumptions (which hold in our work), the sample set converges to the true posterior $Bel(x_t)$ as m goes to infinity, with a convergence speed in $O(\frac{1}{\sqrt{m}})$. The speed may vary by a constant factor, which can vary drastically depending on the proposal distribution. Due to the normalization, the particle filter is only asymptotically unbiased. Care has been taken if the number of samples is extremely small (e.g., less than 10), as the bias increases as the sample set size decreases. In all our implementations, however, the number of samples is large and the bias can be neglected (see [56] for an insightful analysis of bias in estimation).

2.4 Examples

We performed systematic experiments in a range of different settings to evaluate the performance of MCL in practice.

Algorithm MCL(X, a, o):

$X' = \emptyset$

for $i = 0$ *to* m *do*

generate random x *from* X *according to* $w_1, \ldots, w_m$

generate random $x' \sim p(x'|a, x)$

$w' = p(o|x', m)$

add $\langle x', w' \rangle$ *to* X'

endfor

normalize the importance factors w *in* X'

return X'

Table 1: The MCL algorithm.

Simulation. Simulation was employed for evaluation since it allows us to freely vary key parameters, such as the amount of sensor noise. Further below, we will make use of this freedom to characterize situations in which MCL performs poorly.

Figure 4 shows an example in which a simulated mobile robot localizes an object in 3D. This robot can detect the (approximate) location of the object in the image taken by its camera, but the lack of depth estimation in mono-vision makes it impossible to localize the object from a single camera image. Instead, the robot has to view the object from multiple viewpoints. However, changing the viewpoint introduces additional uncertainty, as robot motion is inaccurate. Additionally, the visual field of the robot is limited to a narrow region in front of the robot, which further complicates the object localization problem. Our noise simulation includes a simulation of measurement noise, false positive measurements (phantom detections) and false negative measurements (failures to detect the target object). To enable us to systematically vary key parameters such as the perceptual noise, our results use a mobile robot simulator that accurately models control error (from a RWI B21 robot) and noise in perception (Gaussian position noise, false negatives, and false positives). Notice that this task is more difficult than the first one, due to the impoverished nature of the robot sensors and the large number of symmetries.

Figure 4 depicts different states of global object localization. Initially, the pose of the object is unknown, as represented by the uniform sample set in Figure 4 (first diagram). As the robot turns in a wide circle, unable to see the object (despite a handful of phantom detections), the samples gradually populate the unexplored part of the state space (Figure 4, second diagram). The third diagram shows the first "correct" object sighting, which reinforces the samples that happen to be close to the correct object pose. A few measurements later, repetitive sightings of the object lead to a posterior shown in the fourth diagram of Figure 4. Figure 5 shows systematic error curves for MCL in global localization for differ-

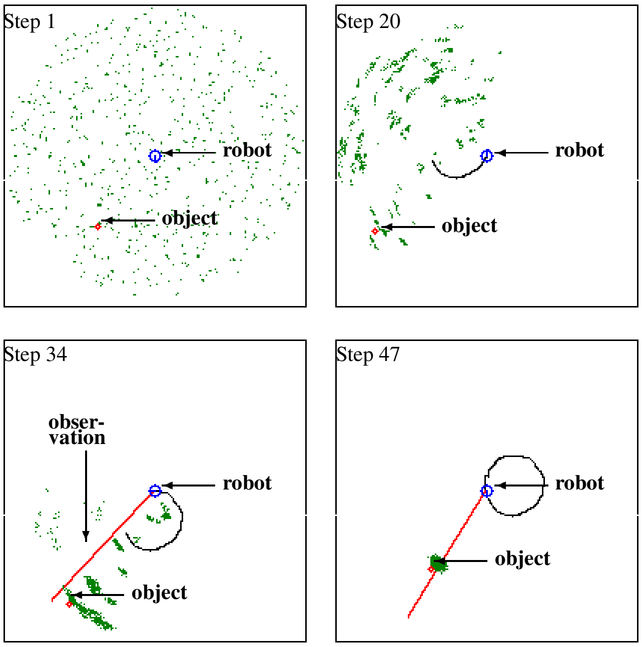


Figure 4: Successful localization sequence. Final error is 36.0 cm.

ent sample set sizes m, averaged over 1,000 individual experiments. The bars in this figure are confidence intervals at the 95% level.

The reader should notice that these results have been obtained for a perceptual noise level of 20% (for both false-negative and false-positive) and an additional position noise that is Gaussian-distributed with a variance of 10 degrees. For our existing vision system, the errors are much lower.

Robot with sonar sensors. Figure 6 shows an example of 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 6a, the robot is globally uncertain; hence the samples are spread uniformly through the free-space (projected into 2D). Figure 6b shows the sample set after approximately 2 meters 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 6c.

Of particular interest shall be a comparison of MCL to previous localization algorithms capable of global mobile robot localization. In particular, we compared MCL to grid-based Markov localization, our previous best stochastic localization algorithm and one of the very few algorithms capable of localizing a robot globally [5, 20]. The grid-based localization

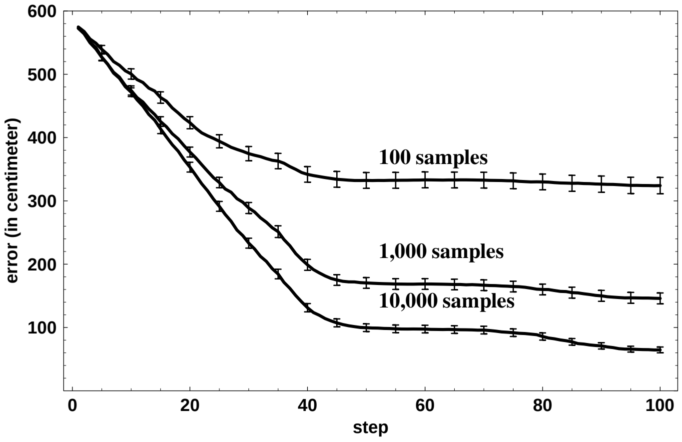
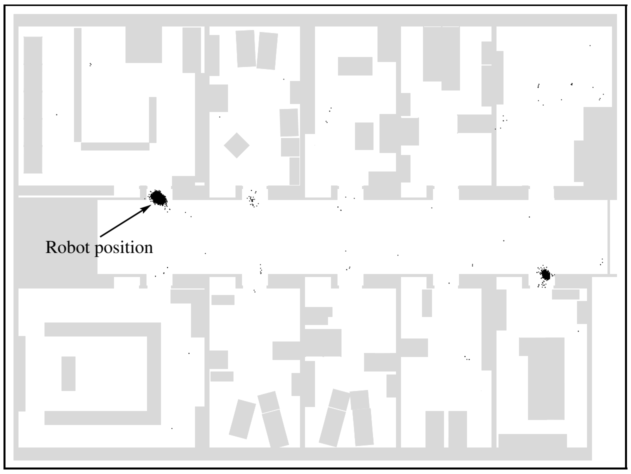


Figure 5: Average error of MCL as a function of the number of robot steps/measurements.

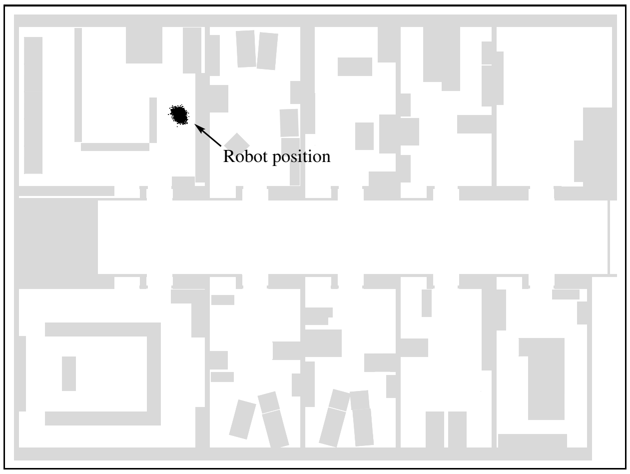
algorithm relies on a fine-grained piecewise constant approximation for the belief Bel, using otherwise identical sensor and motion models. The fact that our implementation employs identical sensor and motion models and is capable of processing the same data greatly facilitates the comparison. Figure 7a plots the localization accuracy for grid-based localization as a function of the grid resolution. Notice that the results in Figure 7a 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 in the specific testing environment, even when highly efficient selective update schemes are applied [21]. Results for MCL with fixed sample set sizes are shown in Figure 7b. 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 7b, 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 best computers.

Robot with upward-pointed camera. Similar results were obtained using a camera as the primary sensor for localization [9]. 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 8) was employed as a tour-guide in the Smithsonian's Museum of Natural History, during which it traversed more than 44km [59]. To aid localization, Minerva is equipped with a camera pointed towards the ceiling. Figure 8 shows a mosaic of the museum's ceiling. Since the ceiling height is unknown, only the center region in the

(a)



(b)



(c)

Figure 6: Global localization of a mobile robot using MCL (10,000 samples): (a) initial particle set, uniformly distributed (projected into 2D). (b) Particles after approx. 2 meters of robot motion. Due to environment symmetry, most particles are centered around two locations. (c) Particle set after moving into a room, thereby breaking the symmetry.

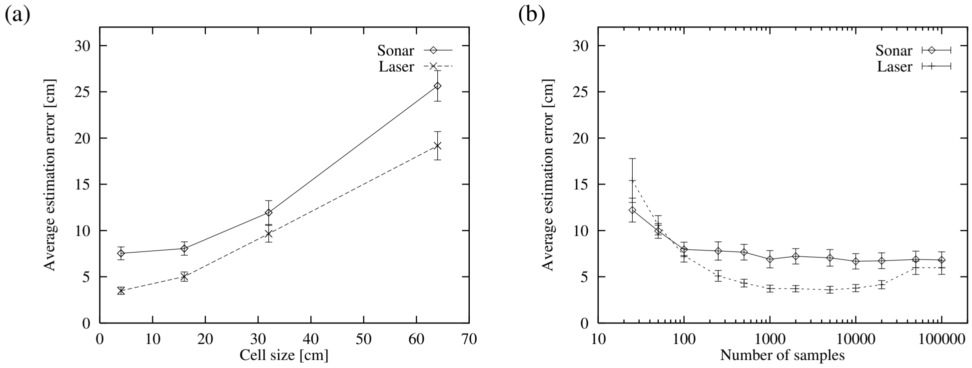


Figure 7: (a) Accuracy of grid-based Markov localization using different spatial resolutions. (b) Accuracy of MCL for different numbers of samples (log scale).

camera image is used for localization.

This data set is among the most difficult 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 crossed 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 in this specific data set. Figure 9 shows a sequence of belief distributions during localization using MCL. However, as we will see below, in a second data sequence recorded during the museum's opening hours even MCL fails to localize the robot.

3 Limitations of MCL

As noticed by several authors [14, 40, 49], 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 10 shows, for our object localization example, 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). It appears that MCL works best for 10% to 20% perceptual noise. The degradation of performance towards the right, when there is a lot of noise, hardly 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.

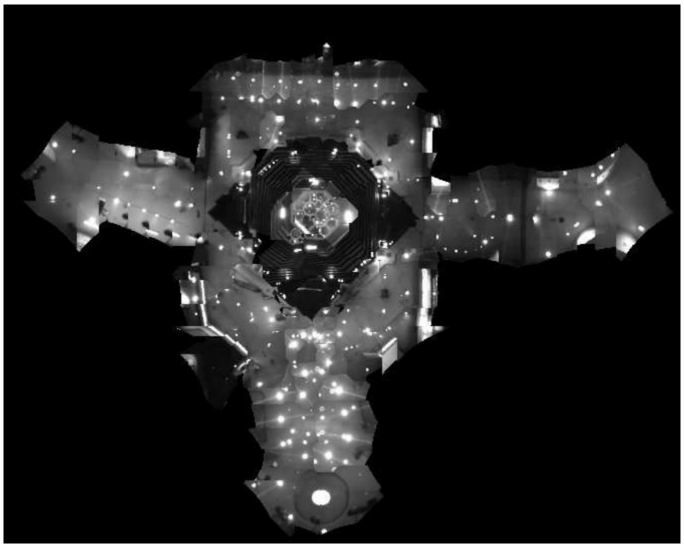
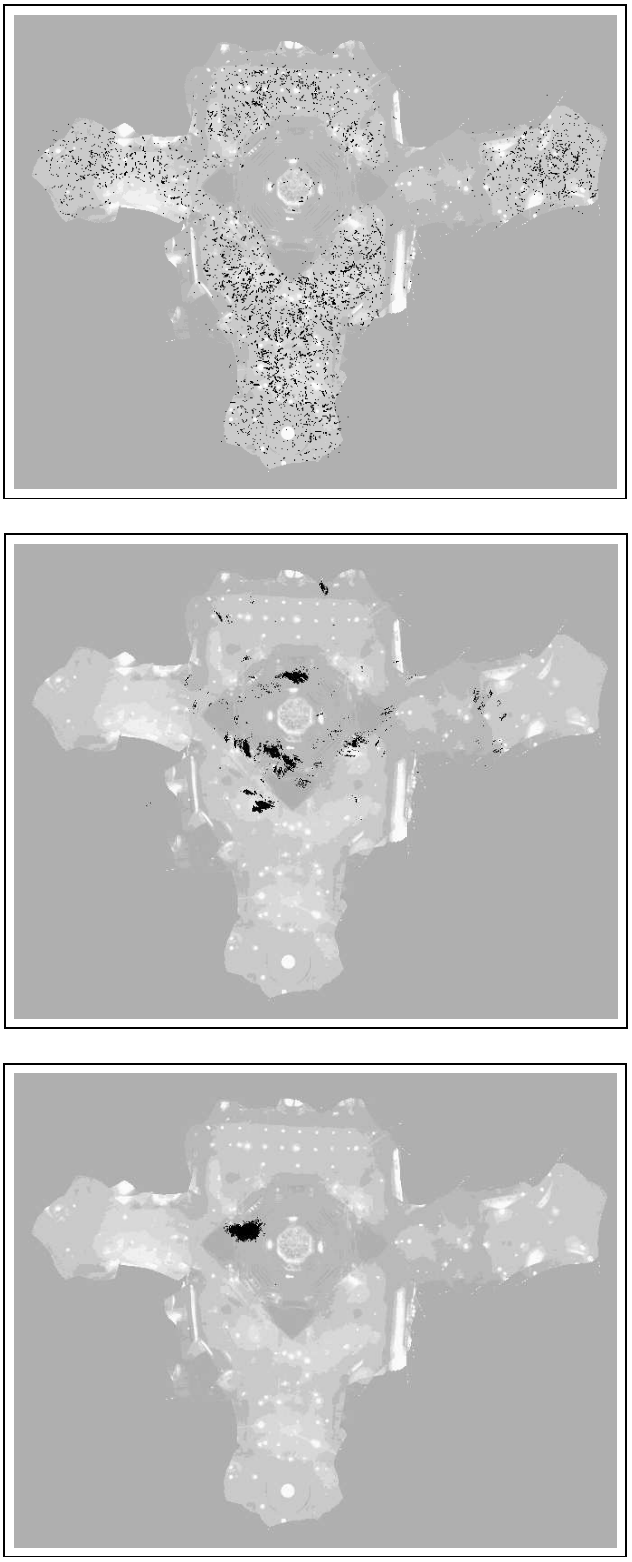


Figure 8: Ceiling map of the National Museum of American History, which was used as the perceptual model in navigating with a vision sensor.

Figure 11 depicts an example run for highly accurate sensors in which MCL fails. When the object is first sighted, none of the samples is close enough to the object's true position. As a consequence, MCL gradually removes all samples with the exception of those located in the robot's "dead spot," which is the center of its circular trajectory. Clearly, localization fails in this example, with a final error of 394 cm. Unfortunately, the more accurate the sensors, the smaller the support of $p(o|x)$, hence the more likely this problem occurs.

There are, of course, fixes. At first glance, one might add artificial noise to the sensor readings. A more sensible strategy would be to use a perceptual model $p(o_t|x_t)$ that over-estimates the actual sensor noise. In fact, such a strategy, which has been adopted in [18], partially alleviates the problem: The dashed curve in Figure 10b 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 fix. 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. Other fixes include the addition of *random* samples into the posterior [18], and the generation of samples at locations that are consistent with the sensor readings [37]—a strategy that is similar to Mixture-MCL below but, without proper sample weighting, is mathematically unmotivated. While these approaches have shown superior performance over strict MCL in certain settings, they all lack mathematical rigor. In particular, neither of them converge to the true posterior as the sample set size goes to infinity, and even with large sample set sizes they may diverge arbitrarily.

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 (11). Thus, the op-



(a) (b) (c)

Figure 9: Global localization of a mobile robot using a camera pointed at the ceiling.

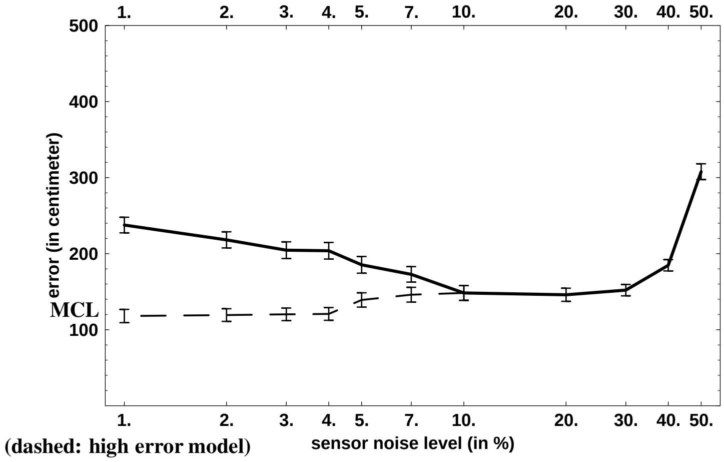


Figure 10: Solid curve: error of MCL after 100 steps, as a function of the sensor noise. 95% confidence intervals are indicated by the bars. Notice that this function is *not* monotonic, as one might expect. Dashed curve: Same experiment with high-error model.

timal 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_t defined in Equation (10), and uses the importance factors (12) to account for the difference. It is well-known from the statistical literature [14, 40, 49, 57] that the divergence between (10) and (11) determines the convergence speed. This difference is accounted by the perceptual density $p(o_t|x_t)$: If the sensors are entirely uninformative, this distribution is flat and (10) is equivalent to (11). For low-noise sensors, however, $p(o_t|x_t)$ is typically quite narrow, hence MCL converges slowly. Thus, the error in Figure 10 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 (10) and (11) in MCL. As we will show in this paper, an alternative version of MCL exists that practically eliminates the second error source, thereby enhancing the accuracy and robustness of the approach.

4 Mixture-MCL

4.1 The Dual of MCL

We will now derive an alternative version of MCL, called *dual Monte Carlo localization*. This algorithm will ultimately lead to the main algorithm proposed in this paper, the *Mixture-MCL* algorithm.

The key idea of the dual is to "invert" the sampling process, by exchanging the roles of the proposal distribution and the importance factors in MCL. More specifically, dual MCL

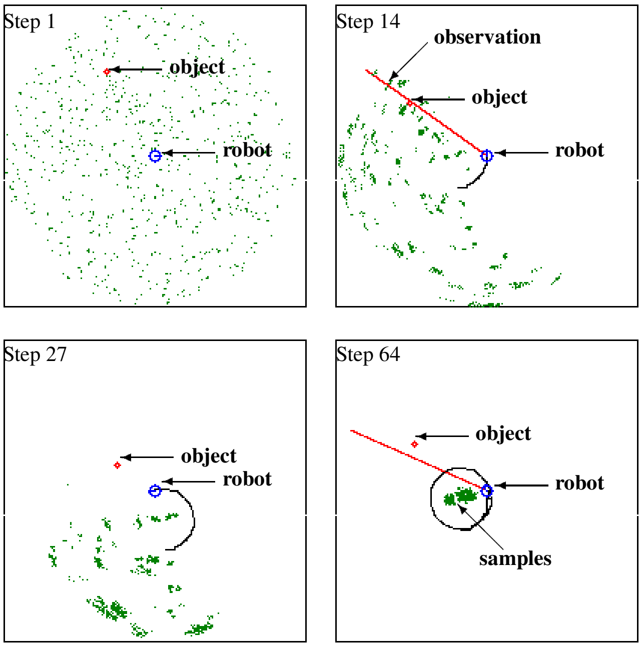


Figure 11: Unsuccessful sequence. Final error is 394 cm.

generates samples of the state $x_t^{(i)}$ by virtue of the following proposal distribution:

$$\bar{q}_t = \frac{p(o_t|x_t)}{\pi(o_t)} \quad \text{with} \quad \pi(o_t) = \int p(o_t|x_t)\, dx_t \tag{13}$$

Here the normalizer, $\pi(o_t)$, is assumed to be finite, which indeed is the case for mobile robot localization in environments of bounded size. Dual MCL can be viewed as the logical inverse of the sampling in regular MCL: Rather than guessing the state $x_t^{(i)}$ and then using the most recent observation to adjust the importance of a guess, dual MCL guesses states corresponding to the most recent observation, and adjusts the importance factor in accordance with the prior belief $Bel(x_{t-1})$. Consequently, the dual proposal distribution possesses complimentary strengths and weaknesses: while it is ideal for highly accurate sensors, its performance is negatively affected by measurement noise. The key advantage of dual MCL is that when the distribution of $p(o|x)$ is narrow—which is the case for low-noise sensors—dual sampling can be much more effective than conventional MCL.

4.2 Importance Factors

We will now provide three alternative ways to calculate the importance factors for $\bar{q}_t$. The first is mathematically the most elegant, but for reasons detailed below it is not well-suited for mobile robot localization. The other two require an additional smoothing step. Both work well for mobile robot localization.

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

$$\bar{q}_{1,t} = \frac{p(o_t|x_t)}{\pi(o_t)} \times Bel(x_{t-1}) \tag{14}$$

and hence the importance factors are given by the quotient

$$\begin{aligned} w^{(i)} &= \left[\frac{p(o_t|x_t^{(i)})}{\pi(o_t)} \times Bel(x_{t-1}^{(i)}) \right]^{-1} \frac{p(o_t|x_t^{(i)})\ p(x_t^{(i)}|x_{t-1}^{(i)}, a_{t-1})\ Bel(x_{t-1}^{(i)})}{p(o_t|a_{t-1}, \ldots, o_0)} \\ &= \frac{p(x_t^{(i)}|x_{t-1}^{(i)}, a_{t-1})\ \pi(o_t)}{p(o_t|a_{t-1}, \ldots, o_0)} \\ &\propto p(x_t^{(i)}|x_{t-1}^{(i)}, a_{t-1}) \end{aligned} \tag{15}$$

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). In the context of global mobile robot localization, the importance factor $p(x_t^{(i)}|a_{t-1}, x_{t-1}^{(i)})$ will be zero (or vanishingly small) for many pose pairs $\langle x_t^{(i)}, x_{t-1}^{(i)} \rangle$. This is because it is unlikely that *independently* drawn poses $x_t^{(i)}$ and $x_{t-1}^{(i)}$ are "consistent" with the action a_{t-1} under the motion model. We have therefore not implemented this approach. However, for other estimation problems using particle filters it might work well, which is the reason why it is described in this paper.

Approach 2 Alternatively, one may in an explicit forward phase sample $x_{t-1}^{(j)} \sim Bel(x_{t-1})$ and then $x_t^{(j)} \sim p(x_t|x_{t-1}^{(j)}, a_{t-1})$, which represents the robot's belief *before* incorporating the sensor measurement. The "trick" is then to transform the samples $x_t^{(j)}$ into a kd-tree [1, 45] that represents the density $p(x_t|a_{t-1}, d_{0\ldots t})$, which is again the pose belief just before incorporating the most recent observation o_t.

The proposal distribution $\bar{q}_{2,t}$ is equivalent to $\bar{q}_t$ in Equation (14). After the forward phase, the importance weights of our samples $x_t^{(i)} \sim \bar{q}_t$ can then calculated as follows:

$$\begin{aligned} w^{(i)} &= \left[\frac{p(o_t|x_t^{(i)})}{\pi(o_t)} \right]^{-1} \frac{p(o_t|x_t^{(i)})\ p(x_t^{(i)}|a_{t-1}, d_{0\ldots t-1})}{p(o_t|d_{0\ldots t-1}, a_{t-1})} \\ &\propto p(x_t^{(i)}|a_{t-1}, d_{0\ldots t-1}) \end{aligned} \tag{16}$$

This approach avoids the danger of generating pairs of poses $\langle x_t^{(i)}, x_{t-1}^{(i)} \rangle$ for which $w^{(i)} = 0$, but it involves an explicit forward sampling phase. The kd-tree effectively *smoothes* the resulting density, which further reduces the variance of the estimation—at the expense of introducing bias [35, 33]. However, the primary role of converting samples into kd-trees is that it facilitates the calculation of the importance weights.

Algorithm dual_MCL_3(X, a, o):

$X' = \emptyset$
for $i = 0$ *to* m *do*
  generate random $x \sim p(x|o)/\pi(o)$
  generate random $x' \sim p(x|\bar{a}, x')/\pi(x|a)$
  $w = Bel(x')$
  add $\langle x, w \rangle$ *to* X'
endfor
normalize the importance factors w *in* X'
return X'

Table 2: The dual MCL algorithm (third approach).

Approach 3 The third approach avoids the explicit forward-sampling phase of the second approach, but also tends to generate large importance factors. In particular, it transforms the initial belief $Bel(x_{t-1})$ into a kd-tree. For each sample $x_t^{(i)} \sim \bar{q}_t$, we now draw a sample $x_{t-1}^{(i)}$ from the distribution

$$\frac{p(x_t^{(i)}|a_{t-1}, x_{t-1})}{\pi(x_t^{(i)}|a_{t-1})} \tag{17}$$

where

$$\pi(x_t^{(i)}|a_{t-1}) \quad = \quad \int p(x_t^{(i)}|a_{t-1}, x_{t-1}) \, dx_{t-1} \tag{18}$$

As above, we assume that the integral $\pi(x_t^{(i)}|a_{t-1})$ is finite, which is trivially the case in the context of mobile robot localization.

In other words, our approach projects $x_t^{(i)}$ back to a possible predecessor pose $x_{t-1}^{(i)}$. Consequently, the pair of poses $\langle x_t^{(i)}, x_{t-1}^{(i)} \rangle$ is distributed according to the proposal distribution

$$\bar{q}_{3,t} \quad = \quad \frac{p(o_t|x_t^{(i)})}{\pi(o_t)} \times \frac{p(x_t^{(i)}|a_{t-1}, x_{t-1}^{(i)})}{\pi(x_t^{(i)}|a_{t-1})} \tag{19}$$

which gives rise to the following importance factor:

$$w^{(i)} \quad = \quad \left[\frac{p(o_t|x_t^{(i)})}{\pi(o_t)} \times \frac{p(x_t^{(i)}|a_{t-1}, x_{t-1}^{(i)})}{\pi(x_t^{(i)}|a_{t-1})} \right]^{-1}$$

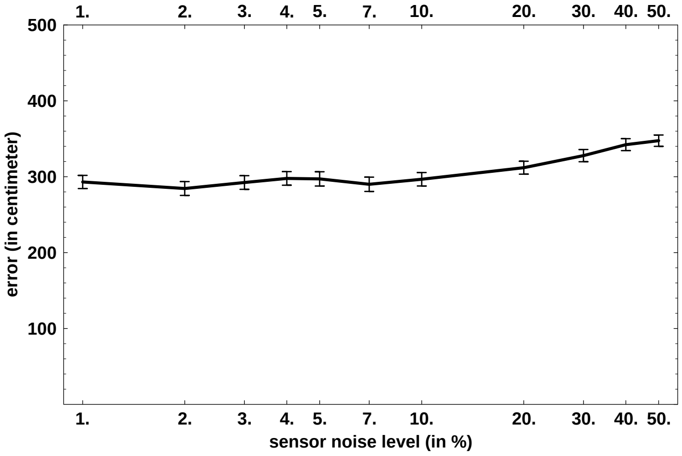


Figure 12: Error of dual MCL as a function of the sensor noise. The error appears to increase monotonically with the sensor noise, but the overall error level is high. Compare this graph to Figure 10.

$$\begin{aligned} & \frac{p(o_t|x_t^{(i)})p(x_t^{(i)}|x_{t-1}^{(i)},a_{t-1})\ Bel(x_{t-1}^{(i)})}{p(o_t|d_{0\ldots t-1})} \\ = \quad & \frac{\pi(o_t)\ \pi(x_t^{(i)}|a_{t-1})\ Bel(x_{t-1}^{(i)})}{p(o_t|d_{0\ldots t-1})} \\ \propto \quad & \pi(x_t^{(i)}|a_{t-1})\ Bel(x_{t-1}^{(i)}) \end{aligned} \tag{20}$$

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 $\pi(x_t^{(i)}|a_{t-1})$, which depends on both $x_t^{(i)}$ and a_{t-1}. Luckily, in mobile robot localization, $\pi(x_t^{(i)}|a_{t-1})$ can safely be assumed to be a constant, although this assumption may not be valid in general. Table 2 shows the algorithm that implements this specific version of dual MCL.

The reader should notice that all three approaches require a method for sampling poses from observations according to $\bar{q}_t$—which can be non-trivial in mobile robot applications. The first approach is the easiest to implement and mathematically most straightforward. However, as noted above, we suspect that it will be inefficient for mobile robot localization. The two other approaches rely on a density estimation method (such as kd-trees). The third also requires a method for sampling poses backwards in time, which further complicates its implementation. However, the superior results given below may well make this additional work (i.e., implementing the dual) worthwhile.

Unfortunately, dual MCL alone is insufficient for localization. Figure 12 depicts the performance of dual MCL using the third approach, under the same conditions that led to the MCL results shown in Figure 10. In both figures, the horizontal axis depicts the amount

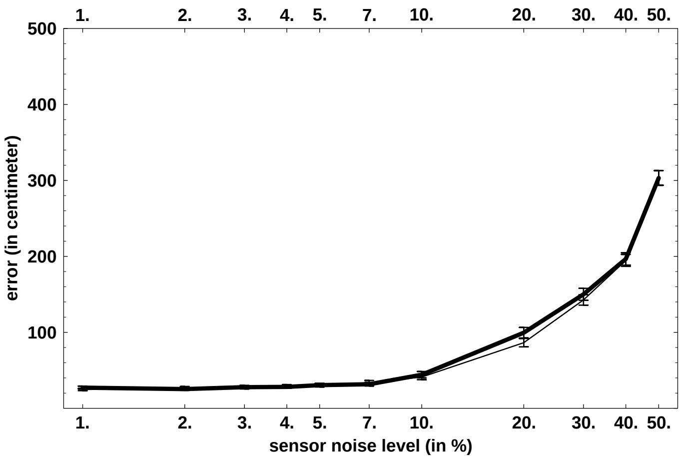


Figure 13: Error of Mixture-MCL, combining regular and dual importance sampling. Mixture-MCL outperforms both components by a large margin, and its error is monotonic in the sensor noise. Thick line: second approach, thin line: third approach for dual sampling. Compare this graph to Figures 10 and 12.

of noise in perception, and the vertical axis depicts the error in centimeters, averaged over 1,000 independent runs. Two things are remarkable: First, the accuracy is now *monotonic* in perceptual noise: More accurate sensors give better results. Second, however, the overall performance is much poorer than that of conventional MCL. The poor performance of dual MCL is due to the fact that *erroneous* sensor measurements have a devastating effect on the estimated belief, since almost all samples are generated at the "wrong" place.

4.3 The Mixture-MCL Algorithm

Neither proposal distribution alone—the original distribution q described in (10) and the alternative distribution $\bar{q}$ given in (13)—is satisfactory. The 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.

However, a mixture of both proposal distributions gives excellent results:

$$(1-\phi)q \ + \ \phi\bar{q} \tag{21}$$

Here ϕ (with $0 \leq \phi \leq 1$) denotes the *mixing ratio* between regular and dual MCL. The resulting algorithm is called *Mixture-MCL* with parameter ϕ.

Figure 13 shows performance results of MCL using this mixture proposal distribution, for the second (thick line) and third (thin line) approach. All experiments use a fixed mixing ratio $\phi = 0.1$, and are averaged over 1,000 independent experiments per data point.

Comparison with Figure 10 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. More results will be given further below.

5 Sampling From The Dual Proposal Distribution

For some sensors, sampling from the dual proposal distribution $\bar{q}$ can be far from trivial. For example, if the mobile robot uses proximity sensors and a metric map as described in Section 2.2, sampling from the inverse, $\bar{q}$ is *not* straightforward. The reader may recall that Section 2.2 outlines a closed-form routine for computing the forward model $p(o|x)$, which accepts both the pose x and an observation o as an input. However, dual MCL requires us to sample poses x from a density proportional to $p(o|x)$, given just the observation o as an input. In other words, we are in need of a routine that accepts a range scan o as input, and which generates poses x.

The key idea here is to *"learn"* a sampling model of the joint distribution $p(o, x)$ from data, such that samples of the desired proposal distribution can be generated with ease. The specific representation chosen here is again a set of kd-trees, each of which models $p(o, x)$ for a subset of "similar" observations o, and each of which recursively partitions the space of all poses in a way that makes it easy to sample from $\bar{q} = p(o|x)/\pi(o)$.

The data used to construct the trees are samples $\langle x, o\rangle$ of poses x and observations o that are distributed according to the joint distribution, $p(x, o)$. There are two ways to sample from the joint: synthetically, using the existing probabilistic models, and using the physical robot to gather data (and the probabilistic model to augment such data).

- The *synthetic sampling* scheme is relatively straightforward. To generate a single sample, joint can be done in two cascaded sampling steps

  1. a pose x is sampled from a uniform distribution, and
  2. for this pose, an observation is sampled according to $p(o|x)$.

  Sampling is repeated, until a sufficient number of samples has been generated (e.g., a million). Obviously, the resulting sample set represents the joint distribution $p(x, o)$.

- An alternative way to generate samples of the joint is to use *data collected by a physical robot*. This approach is preferable if one cannot easily sample from the perceptual model $p(o|x)$—as is actually the case in our existing software implementation (fixes such as rejection sampling [57] are inefficient). In general, sampling from the perceptual model is particularly difficult when using cameras, since this is a problem similar to the computer graphics problem. Luckily, sensor measurements o randomly collected by a robot are samples of $P(o)$, assuming that the robot is placed randomly in its environment. However, robots are usually unable to measure their poses—otherwise there would not be a localization problem.

Luckily, importance sampling [51] offers a solution. The following routine generates samples from the desired joint distribution:

1. Generate an observation o using a physical robot in a known environment with a random (but unknown) pose.
2. Generate a large number of poses x according to a uniform distribution over the set of all robot poses.
3. For each pose x, compute the *importance factor* $p(o|x)$ using the perceptual model described in Section 2.2.

Samples $\langle x, o \rangle$ generated using this approach, along with their importance factors $p(o|x)$, approximate the joint $p(o, x)$.

Equipped with samples representing the joint distribution, let us now turn our attention to learning trees that permit fast sampling from the dual proposal distribution $\bar{q}$. Again, there are multiple options to generate such trees. In our approach, the sensor measurements o are mapped into a low-dimensional feature vector. For laser range scans, a good set of features are the following three:

- The location of the center of gravity of the sensor scan, relative to the robot's local coordinate system. The location is encoded in polar coordinates, hence comprise two parameters.
- The *average* distance measurement, which is a single numerical parameter.

Together, these three values are treated like *sufficient statistics*, that is, we assume it suffices to know $f(o)$ to sample x, hence

$$\frac{p(o|x)}{\pi(o)} = \frac{p(f(o)|x)}{\pi(f(o))} \tag{22}$$

A discrete grid is then stipulated over these three values, and a kd-tree is grown for every (discrete) combination of the feature values $f(o)$. Each tree, thus, is conditioned on $f(o)$ (and hence on o). The depth of the tree depends on the total likelihood of a region in pose space: the more likely a pose given a specific observation $f(o)$, the deeper the tree (and hence the smaller the region covered by that leaf). Sampling from a tree is very efficient, since it only involves a small number of random coin flips.

Figure 14 illustrates our sampling algorithm in practice. Shown in Figure 14a is an example range scan along with an occupancy grid map [16, 46] as described in Section 2.2. From this scan, our approach extracts the three features described above (center of gravity, average distance). Figure 14b shows the tree that corresponds to these features, which partitions the state space recursively into small hyper-rectangular regions. Sampling from this tree yields sample sets like the one shown in Figure 14c.

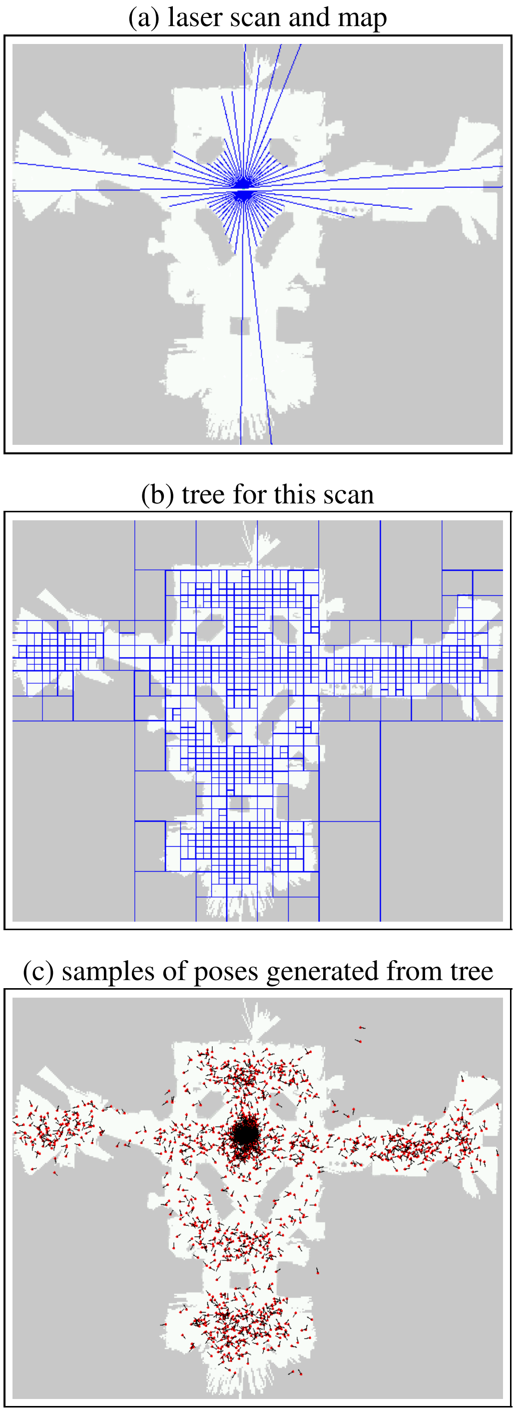


Figure 14: Sampling from a distribution proportional to $p(x|o)$: (a) example range scan and map, (b) tree that partitions the state space for this scan, and (c) samples of poses x generated from the tree.

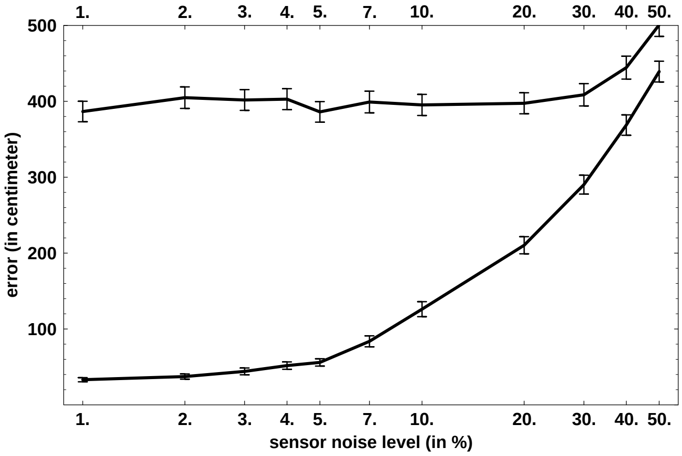


Figure 15: Error of MCL (top curve) and Mixture-MCL (bottom curve) with 50 samples (instead of 1,000) for each belief state.

6 Experimental Results

Systematic experimental results were conducted to evaluate the advantage of Mixture-MCL to regular MCL under a wide range of circumstances. The comparisons were carried out for a range of localization problems, with an emphasis on the more difficult global localization problem and the kidnapped robot problem. As above, real robot experiments were augments by systematic simulation results, where key parameters such as the amount sensor noise could easily be controlled.

Simulation Figure 13 shows the performance of Mixture-MCL, under conditions that are otherwise identical to those in Figures 10. As these results suggest, our new MCL algorithm outperforms both MCL and its dual by a large margin. At every single noise level, our new algorithm outperforms its alternatives by a factor that ranges from 1.07 (high noise level) to 9.7 (low noise level). For example, at a noise level of 1%, Mixture-MCL algorithm exhibits an average error of 24.6cm, whereas MCL's error is 238cm and that of dual MCL is 293cm. In comparison, the average error with noise-free sensors and the optimal estimator is approximately 19.5cm (it's not zero since the robot has to face the object to see it).

Mixture-MCL also degrades nicely to very small sample sets. Figure 15 plots the error of conventional MCL (top curve) and MCL with mixture proposal distribution (bottom curve) for different error levels, using $m = 50$ samples only. With 50 samples, regular MCL basically fails to track the robot's position. Mixture-MCL exhibits excellent performance, and is only slightly inferior to $m = 1,000$ samples. Viewed differently, these findings suggest that Mixture-MCL is computationally an order of magnitude more efficient than conventional MCL.

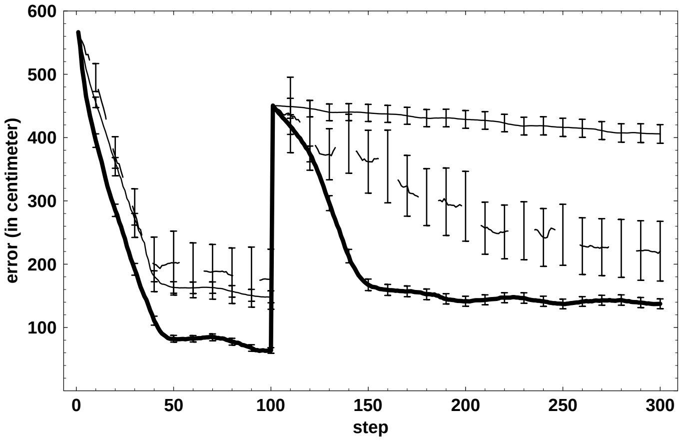


Figure 16: Kidnapped robot problem: Localization error as a function of time, for three different approaches: MCL (thin curve), MCL with added random samples (dashed curve), and Mixture-MCL (thick curve). At time $t = 100$, the robot is tele-ported to a random pose without being told. As these results suggest, Mixture-MCL is most efficient in recovering from this incident.

Finally, Mixture-MCL tends to exhibit superior performance in the kidnapped robot problem. Figure 16 shows the average localization error averaged over 1,000 simulation runs, as a function of time. The tree different curves in that figure correspond to three different algorithms: MCL (thin curve), MCL with added random samples (dashed curve), and Mixture-MCL (thick curve). At time step 100, the robot is kidnapped: it is tele-ported to a random pose without being told. As argued in the introduction, kidnapping is a way to test the ability of a localization algorithm to recover from catastrophic failures. As the results in Figure 16 suggest, Mixture-MCL recovers faster than any alternative MCL algorithm, despite the fact that we optimized parameters such as the ratio of random samples beforehand. Regular MCL fails entirely to recover from the kidnapping, since it tends to lack samples at the new robot pose. The addition of random samples overcomes this problem, but is inefficient. Mixture-MCL places samples more thoughtfully, which increases its efficiency in recovering from kidnapping.

Robot with laser range finder. Mixture-MCL has also been evaluated using data recorded by Minerva, basically confirming the findings obtained in simulation. As outlined above, the data contains logs of odometry measurements and sensor scans taken by Minerva's two laser range-finders (see [19] for details). Figure 17 shows part of the map of the museum and the path of the robot used for this evaluation.

As already reported in Section 2.4, conventional MCL reliably succeeds in localizing the robot. Thus, our attention here is to evaluate Mixture-MCL for the kidnapped robot problem. To do so, we repeatedly introduced errors into the odometry information. These

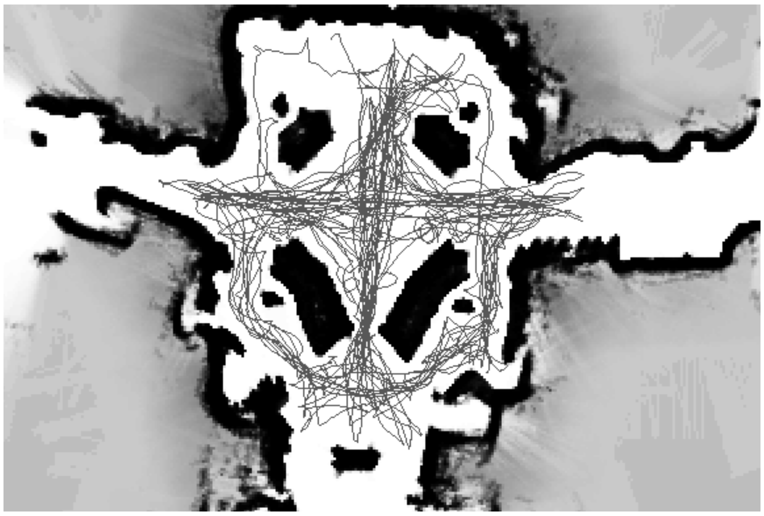


Figure 17: Part of the map of the Smithsonian Museum of National History and path of the robot.

errors made the robot lose track of its position with probability of 0.01 when advancing one meter.

Figure 18 shows comparative results for our three different approaches. The error is measured by the percentage of time, during which the estimated position deviates by more than 2 meters from the reference position. Obviously, Mixture-MCL yields significantly better results, even if the basic proposal distribution is mixed with 5% random samples. 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.

Robot with upward-pointed camera. We also compared Mixture-MCL in the context of visual localization, using only camera imagery obtained with the robot Minerva during public museum hours [9]. Notice that this data set is *not* the same as the one used above; in particular, this data set contained a “natural” odometry error that basically induces a kidnapped robot problem—however, we lack a physical explanation for this incident. Whatever the physical cause may be, this data set is authentic and illustrates the importance of recovery from global localization failure. Moreover, the image sequence used for evaluation is of extremely poor quality, as people often intentionally covered the camera with their hand and placed dirt on the lens.

Figure 19 shows two sample sets (large images), superimposed on the ceiling mosaic, which have been generated by Mixture-MCL during localization. Samples generated by the regular MCL sampler are marked by arrows. Next to these diagrams, the center regions of the most recent camera images are shown (small diagrams), which are used for generating samples in the dual filter. In Figure 19a, the most recent image suggests that the robot

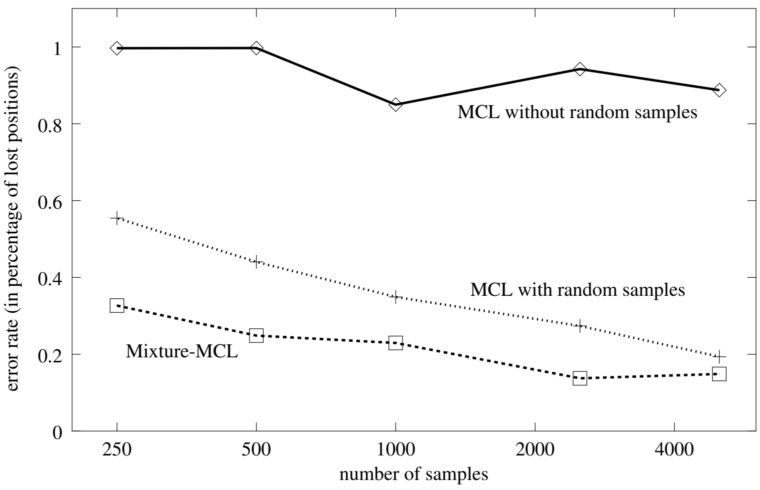


Figure 18: Performance of conventional (top curve), conventional with random samples (middle curve) and mixture (bottom curve) MCL for the kidnapped robot problem in the Smithsonian museum. The error rate is measured in percentage of time during which the robot lost track of its position.

is under a ceiling light. Consequently, the dual sampler generates samples close to light sources. In Figure 19b, the camera measurement is considerably dark, suggesting a location in the center octagon. Notice that we changed the brightness of the ceiling map to increase the visibility of the samples; the authentic ceiling map is shown in Figure 8.

Figure 20 depicts 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, the aforementioned error in the robot's odometry leads to a loss of the position. As the two curves in Figure 20 illustrate, the regular MCL sampler (dashed curve) is unable to recover from this event, whereas MCL with mixture proposal distribution (solid curve) recovers quickly. For this data set, MCL with added random sample performs similarly well as Mixture-MCL. 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 Mixture-MCL drastically increases the robustness of the statistical estimator for mobile robot localization.

7 Related Work

Mobile robot localization is a fundamental problem in mobile robotics that has received considerable attention over the past decades [2, 7, 22, 27, 36, 39, 50, 53, 61]. As argued in the introduction of this article, the vast majority of work focuses on the position tracking problem, where errors are assumed to be small. Most approaches are incapable of recovering from localization failures, though methods exist for detecting such conditions. Usually, failures of the localization component require that a robot's position is entered manually.

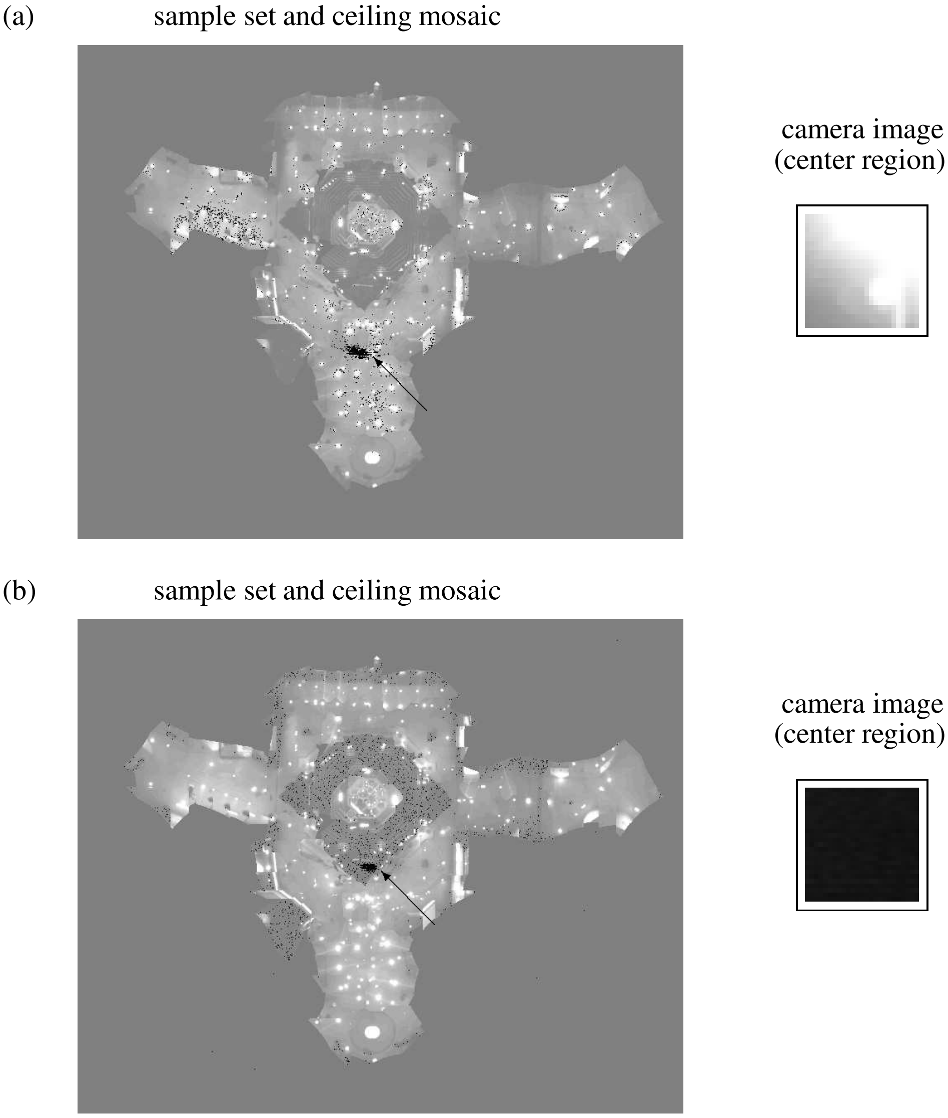


Figure 19: Two sample sets generated by Mixture-MCL, along with the most recent camera image (center region of the image only). The arrows mark the samples generated using the conventional MCL sampler, whereas the other samples are generated by the dual. In (a), the most recent camera measurement suggests that the robot is near a ceiling light, whereas in (b) the measurement suggests a location in the center octagon.

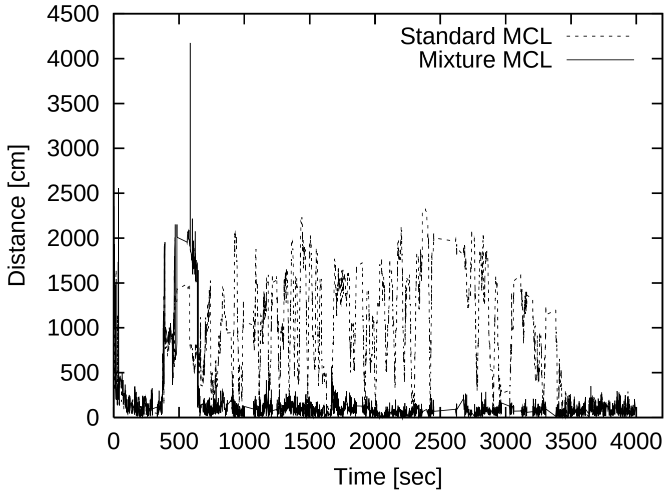


Figure 20: MCL with the standard proposal distribution (dashed curve) compared to Mixture-MCL (solid line). Shown here is the error for a 4,000-second episode of camera-based localization in the Smithsonian museum.

Approaches that solve the global localization and the kidnapped robot problem are relatively recent, and they commonly rely on Bayes filtering with multi-modal density representations, just like MCL. A recent paper [21] gives a more comprehensive overview of algorithms for mobile robot localization; we refer the interested reader to this paper.

Particle filters have become extremely popular for tracking and position estimation in the last few years, as documented by a forthcoming book on this topic [15]. Most existing research on particle filtering focuses on the mathematical foundations. Recent research, thus, has led to a range of variants of the basic particle filters. The poor performance of particle filtering in cases where the proposal distribution differs significantly from the target distribution has been observed by several authors, e.g., [14, 33, 40, 49]. Typical "fixes" involve the design of a different proposal distribution that places more weight in the tails of the distribution.

Despite their young age, particle filters have already been applied with great success to estimation and tracking problems of practical importance. In computer vision, particle filters are commonly known as *condensation algorithm*, where they have been applied with remarkable success to visual tracking problems [28, 29, 42]. To the best of our knowledge, their application to mobile robot localization has originally been proposed in [10, 18], and since been adopted (and extended) by several other researchers [13, 37]. In our own work, we recently extended the basic paradigm to collaborative localization for a whole team of mobile robots [19].

To the best of our knowledge, the idea of the dual particle filter proposed here and in [60] is new. Obviously, it works well in the context of mobile robot localization. While the aim of the paper is to evaluate the Mixture-MCL algorithm in practice, it should be straightforward to devise a proof of convergence of all three versions of Mixture-MCL, assuming convergence of kd-trees. The idea of a dual is related to a recent paper by Lenser and Veloso [37], who also propose to generate samples in accordance with the most recent sensor measurement. Like us, they evaluated their approach in the context of mobile robot localization. There are two main differences between their and our work: First, their approach generates samples that maximize the perceptual density $p(o|x)$, instead of sampling from $p(o|x)$. As a result, their approach does model sensor noise in their "sensor resetting" phase. Second, and more importantly, their approach does not take past evidence into account when generating samples from sensor readings, that is, their approach does not adjust the importance factors of samples generated by the dual in accordance with $Bel(x_{t-1})$. This is mathematically problematic, as the resulting estimate does not approximate the posterior any longer. For example, if the environment consists of disconnected components (e.g., rooms), such an approach can place non-zero likelihood behind walls that are physically impossible to traverse. Our approach can be seen as an approach that relies on the same basic idea, but is mathematically more sound in that it asymptotically approximates the desired posterior, thereby avoiding potential problems that arise from adding samples that only consider the most recent sensor measurement.

The idea of sampling from the sensor measurement (the "evidence") has also been proposed in the context of Bayes networks, in particular in the context of marginalization using Monte Carlo sampling [32]. Under the name of "arc reversal," Kanazawa and colleagues have proposed an efficient sampling algorithm that jump-starts samples at Bayes network nodes whose value is known, then propagating those samples throughout the network to obtain an estimate of the desired marginal distribution. This approach is significantly more efficient than the importance sampler in Bayes networks (which follows the causality expressed by the Bayes network), for reasons that are identical to those given here. Our approach can be viewed as implementing this idea in the context of particle filtering, using somewhat different mathematical equations to account for the differences of Bayes networks and particle filtering. Also, our approach mixes both sampling methodologies, which is essential for the superior performance of this approach, as pointed out above.

8 Conclusion

This paper introduced a new mobile robot localization algorithm, called Mixture Monte Carlo Localization. Mixture-MCL is a version of particle filtering that combines a regular sampler with its dual. By combining both, our approach overcomes a range of limitations that currently exist for different versions of MCL, such as the inability to estimate posteriors for highly accurate sensors, poor degradation to small sample sets, and the ability to recover from unexpected large state changes (robot kidnapping).

Mixture-MCL possesses a range of unique advantages over previous localization algorithms capable of global localization from ambiguous features:

1. **Efficiency.** Mixture-MCL inherits its computational efficiency from particle filters, which focus computational resources in areas that are most probable.

2. **Versatility.** It also inherits from particle filters the ability to approximate a huge range of non-parametric densities. Often, the posterior is centered on a small subspace of the state space. Mixture-MCL does not require an explicit model of this

subspace; instead, it models such subspaces implicitly by generating samples accordingly.

3. **Resource Adaptiveness.** Our implementation of Mixture-MCL is *any-time* [8, 63], in that the number of samples is determined dynamically based on the available computational time between two consecutive sensor measurements. As a consequence, the software can be run on many different computer platforms, where it adapts o the available computational resources.

4. **Robustness.** By mixing regular forward sampling with its dual, Mixture-MCL performs robustly under a range of circumstances, such as highly accurate sensors, robot kidnapping, and very small sample sets.

Extensive experimental results suggest that Mixture-MCL consistently outperforms MCL and other statistical localization algorithms by a large margin.

While this paper focuses on the localization problem, we conjecture that its basic algorithms transcend to a much broader range of state estimation problems for temporal dynamic systems. Bayes filters have been applied to estimation problems for decades, and a recent interest in Monte Carlo approximations [15, 23] suggests that the probabilistic paradigm is well-suited for a broad range of state estimation problems in noisy temporal domains. While this paper has described the limitations of particle filtering in the context of mobile robot localization, we envision that many other estimation domains might suffer similar problems that can be overcome by mixing particle filters with their duals.

Acknowledgments

The authors would like to thank Nando De Freitas and Arnaud Doucet, whose comments on a related paper substantially contributed to the presentation of the material.

References

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

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

[3] 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.

[4] W. Burgard, A. Derr, D. Fox, and A.B. Cremers. Integrating global position estimation and position tracking for mobile robots: The dynamic markov localization approach. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'98)*, 1998. To appear.

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

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

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

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

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

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

[11] F. Dellaert, C. Thorpe, and S. Thrun. 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, 1999.

[12] A.P. Dempster, A.N. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[13] J. Denzler, B. Heigl, and H. Niemann. Combining computer graphics and computer vision for probabilistic self-localization. Internal Report, 1999.

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

[15] A. Doucet, N.J. Gordon, and J.F.G. de Freitas, editors. *Sequential Monte Carlo Methods In Practice*. forthcoming, 2000.

[16] A. Elfes. *Occupancy Grids: A Probabilistic Framework for Robot Perception and Navigation*. PhD thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, 1989.

[17] S. Engelson and D. McDermott. Error correction in mobile robot map learning. In *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, pages 2555–2560, Nice, France, May 1992.

[18] 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.

[19] D. Fox, W. Burgard, H. Kruppa, and S. Thrun. Collaborative multi-robot localization. *Autonomous Robots*, 8(3), 2000. to appear.

[20] D. Fox, W. Burgard, and S. Thrun. Active markov localization for mobile robots. *Robotics and Autonomous Systems*, 25(3-4):195–207, 1998.

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

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

[23] W.R. Gilks, S. Richardson, and D.J. Spiegelhalter, editors. *Markov Chain Monte Carlo in Practice*. Chapman and Hall/CRC, 1996.

[24] J.-S. Gutmann, W. Burgard, D. Fox, and K. Konolige. An experimental comparison of localization methods. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1998.

[25] J.-S. Gutmann and C. Schlegel. Amos: Comparison of scan matching approaches for self-localization in indoor environments. In *Proceedings of the 1st Euromicro Workshop on Advanced Mobile Robots*. IEEE Computer Society Press, 1996.

[26] J. Hertzberg and F. Kirchner. Landmark-based autonomous navigation in sewerage pipes. In *Proc. of the First Euromicro Workshop on Advanced Mobile Robots*, pages 68–73, 1996.

[27] R. Hinkel and T. Knieriemen. Environment perception with a laser radar in a fast moving robot. In *Proceedings of Symposium on Robot Control*, pages 68.1–68.7, Karlsruhe, Germany, October 1988.

[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] M. Isard and A. Blake. Condensation: conditional density propagation for visual tracking. *International Journal of Computer Vision*, 1998. In press.

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

[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. Kanazawa, D. Koller, and S.J. Russell. Stochastic simulation algorithms for dynamic probabilistic networks. In *Proceedings of the 11th Annual Conference on Uncertainty in AI*, Montreal, Canada, 1995.

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

[34] S. Koenig and R. Simmons. Passive distance learning for robot navigation. In L. Saitta, editor, *Proceedings of the Thirteenth International Conference on Machine Learning*, 1996.

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

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

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

[38] J. J. Leonard and H. F. Durrant-Whyte. *Directed Sonar Sensing for Mobile Robot Navigation*. Kluwer Academic Publishers, Boston, MA, 1992.

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

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

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

[42] J. MacCormick and A. Blake. A probabilistic exclusion principle for tracking multiple objects. In *Proceedings of the International Conference on Computer Vision*, Kerkyra, Korfu, 1999.

[43] R.E. Maeder. Ray tracing and graphics extensions. *The Mathematica Journal*, 4(3), 1994.

[44] G.J. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. Wiley Series in Probability and Statistics, New York, 1997.

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

[46] H. P. Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, pages 61–74, Summer 1988.

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

[48] S. Oore, G.E. Hinton, and G. Dudek. A mobile robot that learns its place. *Neural Computation*, 9:683–699, 1997.

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

[50] W.D. Rencken. Concurrent localisation and map building for mobile robots using ultrasonic sensors. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2129–2197, Yokohama, Japan, July 1993.

[51] 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.

[52] B. Schiele and J. Crowley. A comparison of position estimation techniques using occupancy grids. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 1628–1634, San Diego, CA, May 1994.

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

[54] R. Simmons and S. Koenig. Probabilistic robot navigation in partially observable environments. In *Proceedings of IJCAI-95*, pages 1080–1087, Montreal, Canada, August 1995. IJCAI, Inc.

[55] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In I.J. Cox and G.T. Wilfong, editors, *Autonomous Robot Vehnicles*, pages 167–193. Springer-Verlag, 1990.

[56] C. Stein. Inadmissibility of the usual estimator for the mean of a multivariate normal distribution. In *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability 1*, pages 197–206. University of California Press, 1955.

[57] M.A. Tanner. *Tools for Statistical Inference*. Springer Verlag, New York, 1993. 2nd edition.

[58] S. Thrun. Bayesian landmark learning for mobile robot localization. *Machine Learning*, 33(1), 1998.

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

[60] S. Thrun, D. Fox, and W. Burgard. Monte carlo localization with mixture proposal distribution. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, Austin, TX, 2000. AAAI.

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

[62] B. Yamauchi and R. Beer. Spatial learning for navigation in dynamic environments. *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, Special Issue on Learning Autonomous Robots, 1996. also located at http://www.aic.nrl.navy.mil/∼yamauchi/.

[63] S. Zilberstein and S. Russell. Approximate reasoning using anytime algorithms. In S. Natarajan, editor, *Imprecise and Approximate Computation*. Kluwer Academic Publishers, Dordrecht, 1995.