An Online Mapping Algorithm for Teams of Mobile Robots

Sebastian Thrun

October 2000

CMU-CS-00-167

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

We propose a new probabilistic algorithm for online mapping of unknown environments with teams of robots. At the core of the algorithm is a technique that combines fast maximum likelihood map growing with a Monte Carlo localizer that uses particle representations. The combination of both yields an online algorithm that can cope with large odometric errors typically found when mapping an environment with cycles. The algorithm can be implemented distributedly on multiple robot platforms, enabling a team of robots to cooperatively generate a single map of their environment. Finally, an extension is described for acquiring three-dimensional maps, which capture the structure and visual appearance of indoor environments in 3D.

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), 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.

Keywords: Bayes filters, particle filters, mobile robots, multi-robot systems, probabilistic robotics, robotic exploration, robot mapping

1 Introduction

The problem of robot mapping has received considerable attention over the past few years. Mapping is the problem of generating models of robot environments from sensor data. To map an environment, a robot has to cope with two types of sensor noise: Noise in perception (e.g., range measurements), and noise in odometry (e.g., wheel encoders). Because of the latter, the problem of mapping creates an inherent localization problem, which is the problem of determining the location of a robot relative to its own map. Because of this, the mapping problem is often referred to as the *concurrent mapping and localization problem*, the *simultaneous localization and mapping problem*, or simply *SLAM*.

The importance of the problem is due to the fact that many successful mobile robot systems require a map for their operation. Examples include the Helpmate hospital delivery robot [30], Simmons's Xavier robot [60], and various museum tour-guide robots [2, 25, 48, 65]. Consequently, over the last decade, there has been a flurry of work on map building for mobile robots (see e.g., [7, 36, 54, 67]). Some approaches seek to devise abstract, topological descriptions of robot environments [1, 8, 33, 42], whereas others generate more detailed, metric maps [7, 15, 22, 39, 44]. Naturally, the problem of mapping lends itself nicely to multi-robot solutions, where multiple robots collaborate and jointly explore an unknown environment.

In this paper, we will be interested in the problem of building detailed metric maps online, while the robot is in motion. The online characteristic is important for a range of practical problems, such as the robot exploration problem, where mapping is constantly interleaved with decision making as to where to move next. This paper addresses three important open problems in online mapping: the problem of mapping cyclic environments, the problem of generating maps with multiple robots, and the problem of generating three-dimensional maps of building interiors.

Environments with cycles are particularly difficult to map. When closing a cycle, the robot might have accrued a large odometry error. Past solutions, which are discussed in depth towards the end of this paper, either have to resort to highly restrictive assumptions on the nature of the sensor data, or require extensive batch computation to generate a map of cyclic environments. The problem is even more challenging when multiple robots jointly acquire a map. Scalable solutions must perform the computation distirbutedly.

The core of this paper is a statistical framework that combines an incremental maximum likelihood estimator with a posterior pose estimator. More specifically, maps are generated online by finding the most likely continuation of the previous maps under the most recent sensor measurement. However, while online, such a methodology is prone to failure when mapping a cycle. This is because when mapping a cycle, the error in pose estimation might be very large, requiring a backwards-correction of past estimates. To overcome this problem, we propose a second estimator that estimates the residual uncertainty in the robot's pose. When closing the cycle, this estimate is used to determine the robot's pose relative to its own map, and to correct pose estimates backwards in time to resolve inconsistencies. The resulting mapping algorithm is shown to perform favorably in a range of challenging mapping problems. Moreover, it is easily extended into a modular multi-robot mapping approach, where multiple robots integrate sensor measurements into a single, global map.

In addition, we describe an extension of the mapping algorithm that generates 3D maps, reflecting the structural and optical appearance of a building's interior. 3D maps are generated from a a panoramic camera and a laser range sensor mounted perpendicular to the robot's motion direction. They are represented in a standard virtual reality format (VRML), thereby enabling people to interactively 'fly through' an environment from a remote location.

The paper is organized as follows. We begin with introducing the basic statistical framework, which encompasses virtually all major probabilistic mapping algorithms in the literature. From there, we develop the specific set of estimators, making explicit the various assumptions and approximations necessary to obtain a working online algorithm. The algorithm is then extended into a scalable multi-robot mapping algorithm, which under specific initial conditions is shown to enable teams of robots to build accurate maps. Finally, we describe our approach to generating maps in 3D and show results of mapping a corridor segment in our university building. The paper is concluded with a discussion of limitations, related work, and research opportunities that arise from this work.

2 Mapping as Posterior Estimation

2.1 Bayes Filters

Bayes filters are at the core of the algorithm presented here. They comprise a popular class of Bayesian estimation algorithms that are in widespread use of robotics and engineering. Special versions of Bayes filters are known as Kalman filters [28], hidden Markov models [53, 52], and dynamic belief networks [9, 31, 56]. The classical probabilistic occupancy grid mapping algorithm [15, 44] is also a (binary) Bayes filter.

Bayes filtering addresses the problem of estimating an unknown quantity from sensor measurements. In its most general form, it can estimate the state of a time-varying (dynamic) system, such as a mobile robot. Let us denote the state of the environment by x; the state at time x will be denoted x_t. Furthermore, let us denote measurements by z, and the measurement at time t by z_t. In time-varying systems (and in robotic in particular), the change of state is often a function of the controls. Controls will be denoted by u. The control at time t, u_t determines the change of state in the time interval $(t, t+1]$. The set of all measurements and controls will be referred as *data* and denoted by d:

$$d^t \quad = \quad \{z_0, u_0, z_1, u_1, \ldots, z_t\} \tag{1}$$

Without loss of generality, we assume that controls and measurements are alternated. Following standard notation, we adopt the subscript $_t$ to denote to an event at time t, and the superscript ^{t} to denote to the set of all events up to and including time t.

The Bayes filter calculates the posterior probability $p(x_t|d^t)$ of the state x_t conditioned on the data leading up to time t. It does so recursively, i.e., the estimate $p(x_t|d^t)$ is calculated from the previous posterior $p(x_{t-1}|d^{t-1})$ using only the most recent control, u_{t-1}, and measurement, z_t. Once processed, data can be discarded. Hence the time required to calculate $p(x_t|d^t)$ does not depend on the size of the data set. This online property makes Bayes filters well-suited for a range of robotic estimation problem, such as the concurrent mapping and localization problem discussed here.

The recursive update equation of the Bayes filter is as follows:

$$p(x_t|d^t) \quad = \quad p(z_t|x_t) \int p(x_t|u_{t-1}, x_{t-1})\ p(x_{t-1}|d^{t-1})\ dx_{t-1} \tag{2}$$

Thus, the desired posterior $p(x_t|d^t)$ is obtained from the estimate one time step earlier, $p(x_{t-1}|d^{t-1})$, after incorporating the control u_{t-1} and the measurement z_t. The nature of the conditional probabilities $p(z_t|x_t)$ and $p(x_t|u_{t-1}, x_{t-1})$, which play an essential role in this recursive estimator, are discussed in a separate section further below.

Bayes filters owe their validity a specific assumption concerning the nature of the environment, called the *Markov assumption*. The Markov assumption states that given knowledge of the current state, the future is independent of the past. In particular, this implies that the posterior estimate $p(x_t|d^t)$ is a sufficient statistics of past data, with regards to prediction. This assumption is true for worlds in which x is the only state that affects multiple measurements. In static environments, the state comprises the robot pose and the shape of the environment itself. This suggests that the Markov assumption holds true in the concurrent mapping and localization problem. Clearly, in crowded environments the assumption is violated unless people are included in the state x. This case is therefore not addressed in the framework presented here.

To derive Bayes filters, we notice that the desired posterior can be written as

$$p(x_t|d^t) \quad = \quad p(x_t|z_t, u_{t-1}, d^{t-1}) \tag{3}$$

Applying Bayes rule leads to

$$= \quad \eta\ p(z_t|x_t, u_{t-1}, d^{t-1})\ p(x_t|u_{t-1}, d^{t-1}) \tag{4}$$

where η is a normalizer. The Markov assumption allows us to simplify the term $p(z_t|x_t, u_{t-1}, d^{t-1})$, which leads to the more convenient form

$$= \quad \eta\ p(z_t|x_t)\ p(x_t|u_{t-1}, d^{t-1}) \tag{5}$$

Next, we integrate the second term over the previous state x_{t-1}, which gives us

$$= \quad \eta\ p(z_t|x_t) \int p(x_t|u_{t-1}, d^{t-1}, x_{t-1})\ p(x_{t-1}|u_{t-1}, d^{t-1})\ dx_{t-1} \tag{6}$$

Exploiting the Markov assumption once again, we obtain the simplified form

$$= \quad \eta\ p(z_t|x_t) \int p(x_t|u_{t-1}, x_{t-1})\ p(x_{t-1}|d^{t-1})\ dx_{t-1} \tag{7}$$

which completes the derivation of the Bayes filter.

2.2 Concurrent Mapping and Localization

In concurrent mapping and localization, the state x comprises the (unknown) map of the environment and the pose of the robot. Let us denote the map by m and the pose by s. Then we have

$$x \;=\; \langle m, s\rangle \tag{8}$$

This gives rise to the following filter, which describes the problem of concurrent mapping and localization in Bayesian terminology:

$$\begin{aligned} p(s_t, m_t | d^t) \;=\; & p(z_t | s_t, m_t) \int\int p(s_t, m_t | u_{t-1}, s_{t-1}, m_{t-1}) \\ & \qquad p(s_{t-1}, m_{t-1} | d^{t-1})\; ds_{t-1}\; dm_{t-1} \end{aligned} \tag{9}$$

This equation is obtained by substituting (8) into (2).

If the world does not change over time, which is commonly assumed in the literature on concurrent mapping and localization, we have $m_t = m_{t-1}$. Hence, the term $p(s_t, m_t | u_{t-1}, s_{t-1}, m_{t-1})$ is only non-zero if $m_t = m_{t-1}$, and the integration can be omitted:

$$= \; p(z_t | s_t, m_t) \int p(s_t | u_{t-1}, s_{t-1}, m_{t-1})\; p(s_{t-1}, m_{t-1} | d^{t-1})\; ds_{t-1} \tag{10}$$

Stripping the map m of its time index leads to

$$p(s_t, m | d^t) \;=\; p(z_t | s_t, m) \int p(s_t | u_{t-1}, s_{t-1}, m)\; p(s_{t-1}, m | d^{t-1})\; ds_{t-1} \tag{11}$$

It is also common, but not essential, to assume that the effect of robot motion does not depend on the map m. This leads to the recursive estimator:

$$p(s_t, m | d^t) \;=\; p(z_t | s_t, m) \int p(s_t | u_{t-1}, s_{t-1})\; p(s_{t-1}, m | d^{t-1})\; ds_{t-1} \tag{12}$$

This important equation is at the heart of all state-of-the-art concurrent mapping and localization algorithms. Virtually all competitive algorithms can be derived from this equation.

Unfortunately, the computation of the joint posterior over poses and maps is intractable in the context of concurrent mapping and localization, due to the high dimensionality of the space of all maps. Hence, all existing approaches make further restrictive assumptions. For example, the SLAM family of approaches [12, 36, 47, 63] assumes Gaussian-distributed noise and local-linear perception and motion models. Under these assumptions, the posterior can be computed using the extended Kalman filter, transforming the exponentially hard problem into one that scales quadratically in the size of the environment [46, 47]. However, the Gaussian noise assumption has important ramifications in practice. Among others, it requires that sensor measurements can be uniquely associated with objects in the real world, a problem commonly referred to as *data association*.

The data association problem is overcome by a complimentary family of approaches, know as *EM* [59, 67] (based on Dempster's *expectation maximization* algorithm [11]). This

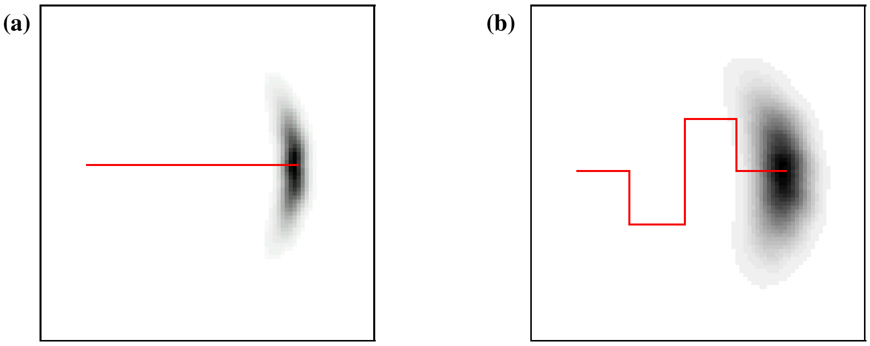


Figure 1: The motion model: Posterior distributions of the robot's pose upon executing the control illustrated by the solid line. The darker a pose, the more likely it is. Poses are projected into 2D, ignoring the robot's heading direction.

approach generates a map by likelihood maximization, treating the robot poses as latent variables. For a fixed data set, it alternates a mapping and a localization step, which after multiple iterations generates a map that locally maximizes the likelihood for unknown data association. Unfortunately, EM is not an online algorithm, hence is inapplicable to problems such as mobile robot exploration.

2.3 Probabilistic Models

To implement the recursive update equation (12), we need three quantities: a distribution $p(s_0, m)$ for initializing the filter, and the conditional distributions $p(s_t|u_{t-1}, s_{t-1})$ and $p(z_t|s_t, m)$. The latter distributions are often referred to as *motion model* and *measurement model*, respectively, since they physically model robot's motion and its sensors.

The initial distribution in mapping is straightforward. Without loss of generality, the pose s_0 is defined to be the origin of the coordinate system with a heading direction of 0 degrees, that is, $s_0 = \langle 0, 0, 0 \rangle$. The map m is initialized with a uniform prior.

The motion model, $p(s_t|u_{t-1}, s_{t-1})$, specifies the probability that the robot's pose is s_t given that it executed the control u_{t-1} at time $t-1$. Thus, the motion model is a probabilistic generalization of mobile robot kinematics. Classical kinematics specifies where one would expect a robot to see if its control and its motion were perfect. In practice, physical robots often suffer from drift an slippage, which induce deviations from the expected pose that are difficult to predict. Hence, the pose s_t is best modeled by a probability distribution over possible poses that the robot might attain after executing u_{t-1} in s_{t-1}.

In our implementation, we assume that the robot experiences translational and rotational error. For a small motion segment (e.g., the distance a robot traverses in .25 seconds), we assume that these errors can be modeled by two independent Gaussian noise variables, which are added to the commanded translation and rotation. Figure 1 shows two examples of the probabilistic motion model $p(s_t|u_{t-1}, s_{t-1})$. In both cases, the robot's pose prior to executing the control is shown on the left. The robot is then commanded to follow a straight-

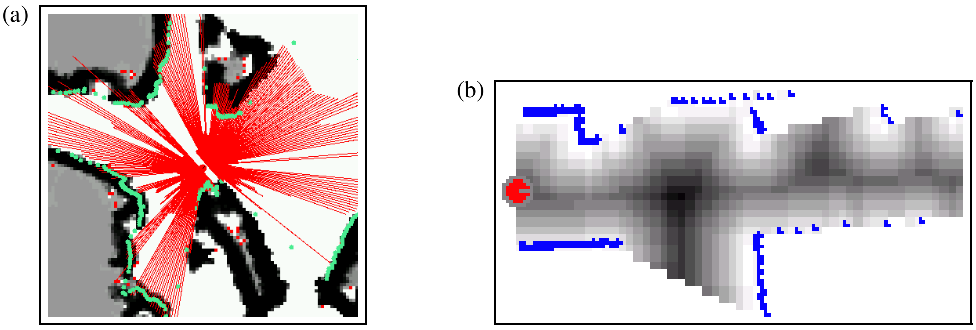


Figure 2: (a) Laser range scan projected into an occupancy grid map. (b) Probabilistic measurement model for an example range scan, which assigns likelihood in proportion to how close to the scan a non-occluded object was observed.

line motion as indicated. The posterior pose after executing the respective motion command is shown in Figure 1, as indicated by the grayly shaded area. The darker a pose, the more likely it is. Obviously, the densities in Figure 1 are only two-dimensional projections of the three-dimensional densities in pose space. A careful comparison of the two diagrams in Figure 1 shows that the control matters. Both examples show a control sequence that lead to the same expected pose, as predicted by classical kinematics. However, the path in the right diagram is longer, inducing more chances of errors as indicated by the increased margin of uncertainty.

The measurement model, $p(z_t|s_t, m)$, physically models the robot's sensors. Under the assumption that the map m and the robot pose s_t is known, the measurement model specifies the probability that z_t is measured. While the mathematical framework put forward in this paper makes no specific assumption on the nature of the sensors, we will consider robots equipped with 2D laser range finders. Such laser range finders are commonly used in mobile robotics. Figure 2a shows a range scan from a bird's eye perspective, of a range finder mounted horizontally on a mobile robot. Each line indicates the range, as measured by the sensor. Also shown is an outline of the environment, which suggests that the range measurements are of high quality.

The measurement model adopted in our implementation is a probabilistic generalization of the rich literature on scan matching [24, 39, 57]. It assumes that each scan induces a local map, which can be conceptually decomposed into three types of areas: free space, occupied space, and occluded space. The same conceptual decomposition applies to the map. Each of the measurements in a range scan z_t can thus fall into three different regions of the map. If it falls into occluded (unknown) space, the probability is uniformly high. If it coincides with occupied space, the probability is high, too. The most interesting case occurs when it falls into free-space, which can be viewed as an inconsistency. The specific probability of the measurement is then given by a function that decreases monotonically with the distance to the nearest object. The specific function in our implementation is composed by a normal (close range) and a uniform (far range) distribution. It levels off at approximately 1 meter distance, assigning equally low probability to measurements further away. The leveling is important for the gradient ascent search algorithm described further below, as it effectively removes outliers when determining the most likely robot pose.

Figure 2b shows an example. Shown there is a sensor scan (endpoints only) for a robot placed at the left of the diagram (circle). The likelihood function is shown by the grey shading: the darker a region, the smaller the likelihood for sensing an object there. Notice that occluded regions are white (and hence incur no penalty).

The likelihood computation is performed symmetrically: First, the likelihood of individual measurements in a scan are evaluated relative to the map. Then, the likelihood of obstacles in the map is calculated relative to the scan. The symmetric computation ensures that occluded measurements are matched to the 'correct' wall. The result of the calculation are a collection of probabilities $p(z_t^{[i]}|s_t, m)$, where $z_t^{[i]}$ denotes measurement likelihood obtained from the the i-th measurement or map object. The resulting probabilities are them multiplied, assuming conditional independence between the measurements:

$$p(z_t|s_t, m) \quad = \quad \prod_i p(z_t^{[i]}|s_t, m) \tag{13}$$

In our implementation, we pre-compute the calculation of the nearest neighbor in a grid (eg, 15 cm resolution). Our implementation also caches trigonometric functions that are expensive to compute, such as sine and the cosine of specific angles. As a result, calculating the measurement model $p(z_t|s_t, m)$ is extremely fast. It takes approximately one millisecond on a 500Mhz Pentium PC for a full scan.

3 Mapping Through Incremental Likelihood Maximization

3.1 Maximum Likelihood Pose Estimation

As pointed out above, computing the full posterior over maps and poses is computationally hard. Even discrete approximations scale exponentially with the size of the map. The exponential scaling can be avoided in cases where the data association is known. However, in most cases the data association is unknown and computing the full posterior online appears to be intractable given today's computers.

We will now discuss a first online algorithm for building maps. The idea is simple, and probably because of its simplicity it is very popular: *Given a scan and an odometry reading, determine the most likely pose. Then append the pose and the scan to the map, and freeze it once and forever.*

Put mathematically, the idea of the incremental maximum likelihood approach is to calculate a sequence of poses

$$\hat{s}_1, \hat{s}_2, \ldots \tag{14}$$

and corresponding maps by maximizing the marginal likelihood of the t-th pose and map relative to the $(t-1)$-th pose and map. To do so, the approach assumes we have at our fingertips a function for building maps incrementally, which requires knowledge of the robot's poses. Let us denote this function by $\hat{m}$, and call it the *incremental map updating function*:

$$\hat{m}(s^t, z^t) \quad = \quad \underset{m}{\operatorname{argmax}}\, p(m|s^t) \tag{15}$$

Functions of this type have been defined in many different ways. In occupancy grid mapping [15, 44], $\hat{m}(s^t, z^t)$ is simply the update equation for the occupancy grid map. In the work by Lu, Milios and Gutmann [40, 22], maps are collections of scans annotated by their pose. Hence, the function $\hat{m}$ simply appends a scan and a pose to the set of scans which comprise the map.

The availability of the incremental map updating function reduces the problem of estimating posteriors over the product space of maps and poses to the simpler problem of estimating poses only:

$$\begin{aligned} p(s_t, \hat{m}(s^t, z^t)|d^t) \quad &= \quad p(z_t|s_t, \hat{m}(s^{t-1}, z^{t-1})) \\ &\quad \int p(s_t|u_{t-1}, s_{t-1})\; p(s_{t-1}, \hat{m}(s^{t-1}, z^{t-1})|d^{t-1})\; ds_{t-1} \end{aligned} \tag{16}$$

This equation is obtained by substituting $\hat{m}$ for m in Equation (12). The incremental maximum likelihood approach calculates at each time step one particular pose, $\hat{s}_t$, instead of computing the full posterior. From this estimate, it uses the function $\hat{m}$ to obtain a single map. Thus, the posterior (16) can be simplified. Instead of assuming the only the posterior over maps and poses at time $t-1$ is known, we make the (much stronger) assumption that we know the pose and the map: The pose at time $t-1$ is $\hat{s}_{t-1}$, and the map is $\hat{m}(\hat{s}^{t-1}, z^t)$. Hence we can re-write (16) as follows:

$$p(s_t, \hat{m}(\hat{s}^{t-1}, s_t, z^t)|d^t) \quad = \quad p(z_t|s_t, \hat{m}(s^{t-1}, z^{t-1}))\; p(s_t|u_{t-1}, \hat{s}_{t-1}) \tag{17}$$

In particular, we notice that the integration over poses disappeared (c.f., Equation (12). The t-th pose is now obtained as the maximum likelihood estimate of (17):

$$\hat{s}_t \quad = \quad \underset{s_t}{\operatorname{argmax}}\, p(z_t|s_t, \hat{m}(\hat{s}^{t-1}, z^{t-1}))\; p(s_t|u_{t-1}, \hat{s}_{t-1}) \tag{18}$$

The so-estimated pose $\hat{s}_t$ is then used to generate a new map via the incremental map updating function $\hat{m}$, and the new map is used from that point on.

To summarize, at any point in time $t-1$ the robot is given a (non-probabilistic) estimate of its pose $\hat{s}_{t-1}$ and a map $\hat{m}(s^{t-1}, z^{t-1})$. As a new control u_{t-1} is executed and a new measurement z_t is taken, the robot determines the most likely new pose $\hat{s}_t$. It does this by trading off the consistency of the measurement with the map (first probability on the right-hand side in (18)) and the consistency of the new pose with the control action and the previous pose (second probability on the right-hand side in (18)). The result is a pose $\hat{s}_t$ which maximizes the incremental likelihood function. The map is then extended by the new measurement z_t, using the pose $\hat{s}_t$ as the pose at which this measurement was taken.

3.2 Search in Pose Space

We now discuss how to find the most likely pose $\hat{s}_t$, that is, how to compute (18). The basic problem is the continuous nature of the space of all robot poses, which makes it impossible to search this space exhaustively. In our approach, $\hat{s}_t$ is found using gradient ascent in log likelihood space. It is common practice to maximize the log likelihood, instead of the likelihood, since the log likelihood is usually mathematically easier to handle. Hence, our approach calculates

$$\hat{s}_t \quad = \quad \underset{s_t}{\operatorname{argmax}} \ln[p(z_t | s_t, \hat{m}(\hat{s}^{t-1}, z^{t-1}))\ p(s_t | u_{t-1}, \hat{s}_{t-1})] \tag{19}$$

which can be decomposed into additive terms:

$$= \quad \underset{s_t}{\operatorname{argmax}} \ln p(z_t | s_t, \hat{m}(\hat{s}^{t-1}, z^{t-1}))\ +\ \ln p(s_t | u_{t-1}, \hat{s}_{t-1}) \tag{20}$$

$$= \quad \underset{s_t}{\operatorname{argmax}} \sum_i \ln p(z_t^{[i]} | s_t, \hat{m}(\hat{s}^{t-1}, z^{t-1}))\ +\ \ln p(s_t | u_{t-1}, \hat{s}_{t-1}) \tag{21}$$

The differentiation of the argument on the right-hand side with respect to the pose s_t leads to the following gradient:

$$\nabla_{s_t} L \quad = \quad \sum_i \nabla_{s_t} \ln p(z_t^{[i]} | s_t, \hat{m}(\hat{s}^{t-1}, z^{t-1}))\ \ +\ \nabla_{s_t} \ln p(s_t | u_{t-1}, \hat{s}_{t-1}) \tag{22}$$

where L denotes the log-likelihood. The remaining gradients, $\nabla_{s_t} p(z_t^{[i]} | s_t, \hat{m}(\hat{s}^{t-1}, z^{t-1}))$ and $\nabla_{s_t} p(s_t | u_{t-1}, \hat{s}_{t-1})$, are the gradients of the measurement model, and the motion model, respectively. We notice that the motion model is differentiable, and the measurement model is piecewise differentiable. From this follows the existence of the gradients. The specific calculation of the gradient is a mechanical exercise, which is omitted here since the gradients are fairly complex.

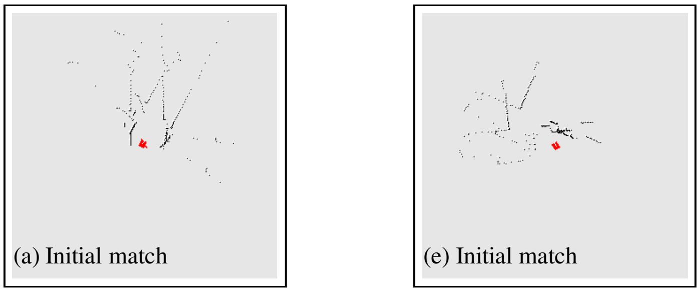
Gradient ascent then maximizes the log-likelihood by successively changing the pose s_t in the direction of the gradient.

$$s_t \quad \longleftarrow \quad s_t + \alpha \nabla_{s_t} L \tag{23}$$

Here $\alpha > 0$ is a stepsize commonly used in gradient ascent.

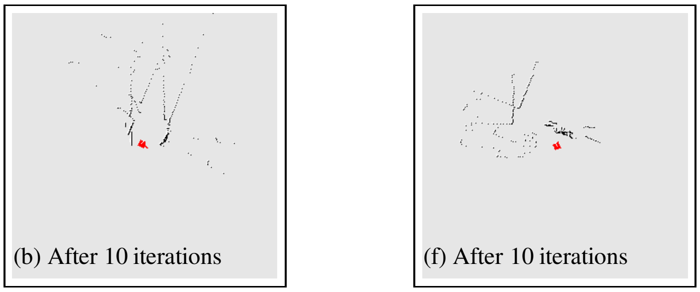
Figure 3 shows two examples of maximizing the likelihood of the relative pose of two consecutive scans through gradient ascent. Each column corresponds to one example. The rows correspond to different points in time. As can be seen, 100 iterations are sufficient to obtain a good alignment even if the initial error is huge. In practice, where misalignments are much smaller, we found that usually 20 iterations are sufficient.

In the literature, mapping with incremental likelihood maximization is an extremely popular approach—with the current algorithm being merely a variant of various known algorithms. While most approaches are not phrased in the context of statistical maximum likelihood estimation, they can easily be derived as special cases of Equation (21), under appropriate definition of the motion and the measurement model. All these approaches follow the same logic: The robot maintains a single map. When a new measurement is received, a ‘plausible’ pose is estimated and the map is then grown accordingly.



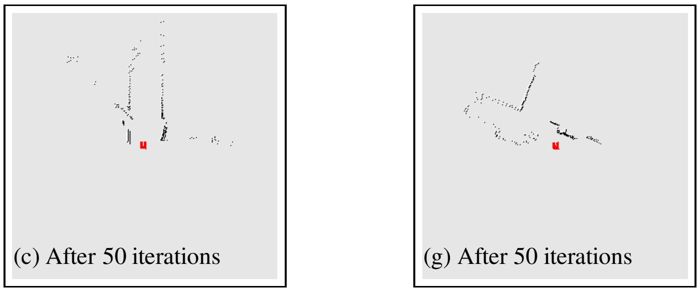
(a) Initial match

(e) Initial match



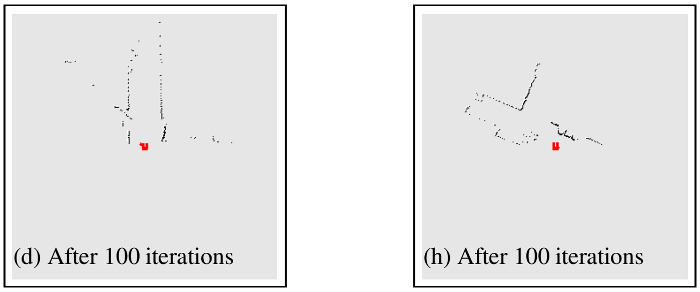
(b) After 10 iterations

(f) After 10 iterations



(c) After 50 iterations

(g) After 50 iterations



(d) After 100 iterations

(h) After 100 iterations

Figure 3: To examples of gradient ascent for aligning scans (arranged vertically). In both cases, the initial translational error is 10 cm along each axis, and the rotational error is 30 degrees. The gradient ascent algorithm safely recovers the maximum likelihood alignment.

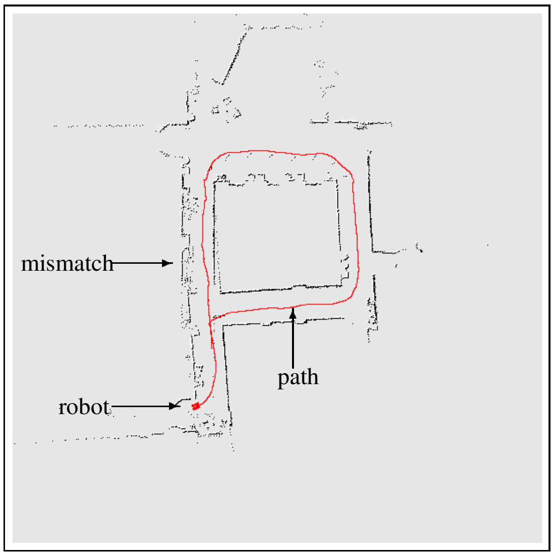


Figure 4: A typical map obtained using the most simple incremental maximum likelihood approach. While the map is locally consistent, the approach fails to close the cycle and leads to inconsistent maps. This elucidates the limitations of stepwise likelihood maximization, and illustrates the difficulty of mapping cyclic environments.

4 Incremental Mapping Using Posteriors

Unfortunately, the incremental maximum likelihood approach suffers a major drawback: It is inapplicable to cyclic environments. This limitation was recognized by Gutmann and Konolige [23] and others [39, 66]. Figure 4 depicts the map obtained in a cyclic environment. Here the map appears to be locally consistent. However, local likelihood maximization still suffers a residual error that accumulates over time. At the time the robot closes the cycle, the error has grown to a reputable amount, making it impossible to generate a map that is locally consistent. The magnitude of the error can grow without bounds, depending on the robot's odometry and the size of the cycle. Hence, the incremental maximum likelihood approach is brittle.

Technically, the inability to close cycles is caused by two factors. Firstly, the method is unable to revise poses backwards in time. This is necessary when a cycle is closed to regain local consistency. Secondly, the robot maintains only a single hypothesis as to where it is. The unawareness of its own uncertainty may prevent the robot from finding a good match when closing a cycle. Gradient methods only work well if the initial estimate is within the basin of attraction of the maximum likelihood estimate. Since the error can grow without bounds, the initial estimate might be arbitrarily far away from the best match when closing a cycle. The resulting map is then usually unusable.

4.1 Pose Posterior Estimation

To remedy this problem, we will now discuss a hybrid estimation approach. This approach extends the incremental maximum likelihood algorithm by a second estimator, which estimates the full posterior over poses (but not maps!). The posterior over poses is given by

$$p(s_t|d^t, m) \quad = \quad p(z_t|s_t, m) \int p(s_t|u_{t-1}, s_{t-1})\; p(s_{t-1}|d^{t-1}, m)\; ds_{t-1} \tag{24}$$

This equation is obtained by substituting the pose s for x in (2), and conditioning everything on the map m (except for the robot motion). It is essentially equivalent to the Markov localization algorithm, a popular algorithm for mobile robot localization with known maps [3, 27, 49, 62]. However, Markov localization assumes the availability of a complete map of the environment, which is not assumed here.

The key reason for memorizing the full posterior $p(s_t|d^t, m)$, instead of the maximum likelihood estimate $\hat{s}_t$ only, it our desire to accommodate the increasing residual uncertainty when moving into unmapped terrain. As the robot traverses a cycle, the posterior grows increasingly large, and the posterior characterizes this uncertainty.

Formally, the posterior is obtained by substituting the maximum likelihood map $\hat{m}(s^{t-1}, z^{t-1})$ into Equation (24), which gives us

$$\begin{aligned} p(s_t|d^t, \hat{m}(s^{t-1}, z^{t-1})) \quad = \quad & p(z_t|s_t, \hat{m}(s^{t-1}, z^{t-1})) \\ & \int p(s_t|u_{t-1}, s_{t-1})\; p(s_{t-1}|d^{t-1}, \hat{m}(s^{t-1}, z^{t-1}))\; ds_{t-1} \end{aligned} \tag{25}$$

Notice the difference between this estimator and the one in (17). Equation (17) computes the posterior over poses and maps; here we assume knowledge of the map and compute the posterior over poses only. This frees us of the necessity to calculate the most likely pose and instead enables us to maintain a full posterior over all poses. On the other hand, calculating (25) requires integration, whereas (17) does not. Luckily, the integration is well-understood in the rich literature on Markov localization, and below we will adopt one of the representations developed there.

Knowledge of the posterior gives us a better way to calculate the maximum likelihood pose $\bar{s}_t$. In particular, $\bar{s}_t$ is obtained by maximizing the posterior (25), instead of the marginal posterior specified in Equation (17):

$$\bar{s}_t \quad = \quad \underset{s_t}{\operatorname{argmax}}\, p(s_t|d^t, \hat{m}(s^{t-1}, z^{t-1})) \tag{26}$$

The resulting pose is generally the same when mapping unexplored terrain. However, it may differ when closing a cycle, at which point the robot might have to compensate a large pose error.

This consideration points out the key advantage of maintaining the full posterior, instead of the most likely pose only. Our approach fundamentally addresses one of the two problems discussed above, namely that of errors growing without bounds when closing a cycle. The

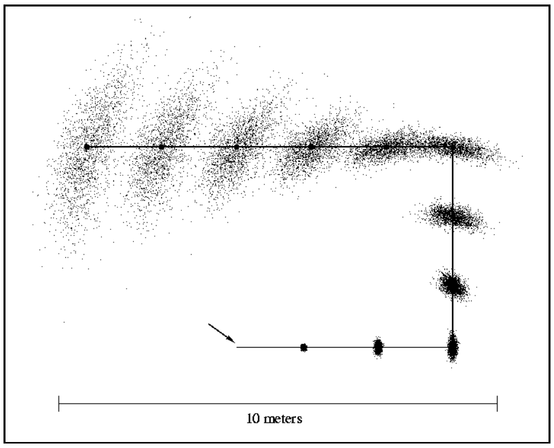


Figure 5: Sample-based approximation for the posterior over poses. Here each density is represented by a set of samples, weighted by numerical importance factors. The Monte Carlo localization algorithm is used to generate the sample sets.

posterior estimator can accommodate such errors. In particular, when closing a cycle the posterior provides a spectrum of hypotheses as to where the robot might be. The sample representation, described in turn, provides a set of potential seeds for the gradient ascent search, which makes it possible to find best matches when closing arbitrary large cycles.

4.2 Approximation Using Monte Carlo Localization

Our algorithm implements the posterior calculation using particle filters [13, 14, 38, 51]. Particle filters apply Rubin's idea of importance sampling [55] to Bayes filters. The resulting algorithm is known in computer vision as *condensation algorithm* [26], and in mobile robotics as *Monte Carlo localization* [10, 17, 34]. A similar algorithm has been proposed in the context of Bayes networks as [29] *survival of the fittest*.

Particle filters represent the posteriors by a set of particles (samples). Each particle is a pose that represents a 'guess' as to where the robot might be. Particle filters weigh each particle by a non-negative numerical factor, commonly referred to as *importance factor*. Figure 5 shows an example of the particle representation in the context of mobile robot localization. Shown there is a sequence of posterior beliefs for a robot that follows a U-shaped trajectory. Each of the sample sets is an approximation of densities (of the type shown in Figure 1).

The general algorithm is depicted in Table 1. When transitioning from time $t-1$ to time t, new particles are generated by sampling poses from the current posterior and guessing a successor pose using the motion model. The weight of the particle is then obtained using the measurement model. Particle filters have been shown to approximate the Bayes filter update equation given in (2). As argued in the statistical literature, the particle representation can approximate almost arbitrary posteriors at a convergence rate of $\frac{1}{\sqrt{N}}$ [64]. It is convenient

```
**Algorithm particle_filter**($X_{t-1}, u_{t-1}, z_t$):

    for i = 1 to n to
        sample $x_{t-1}^{\langle i \rangle}$ from $X_{t-1}$ according to the importance factors $w_{t-1}^{\langle j \rangle}$ in $X_{t-1}$
        sample $x_t^{\langle i \rangle} \sim p(x_t | u_{t-1}, x_{t-1}^{\langle i \rangle})$
        set $w_t^{\langle i \rangle} = p(o_t | x_t^{\langle i \rangle})$
        add $\langle x_t^{\langle i \rangle}, w_t^{\langle i \rangle} \rangle$ to $X_t$
    endfor

    normalize all weights $w_t^{\langle i \rangle}$ in $X_t$ so that they sum up to 1.
    return $X_t$
```

Table 1: The particle filter algorithm. This algorithm is an approximate version of Bayes filters that uses particles to represent the posterior belief. The input is a weighted particle set X_{t-1} representing $p(x_{t-1}|d^{t-1})$, along with a control U_{t-1} and a measurement z_t. The output is a new weighted set of n particles representing $p(x_t|d^t)$.

for robotics, since it is easy to implement, and both more efficient and more general than most alternatives [17].

In the context of mapping, the application of particle filters and MCL for posterior estimation is straightforward. At each point in time, the robot has access to the map built from past data. Another key advantage of the particle representation is that directly facilitates the optimization of (26) using gradient ascent. Our approach performs gradient ascent using each sample as a starting point, then computes the goodness of the result using the obvious likelihood function. If the samples are spaced reasonably densely (which is easily done with only a few dozen samples), one can guarantee that the global maximum of the likelihood function can be found. This differs from the simple-minded approach above, where only a single starting pose is used for hill-climbing search, and which hence might fail to produce the global maximum (and hence a usable map).

4.3 Backwards Correction

As argued above, when closing cycles it is imperative that maps are adjusted backwards in time. The amount of backwards correction is given by the difference Δ_{s_t}:

$$\Delta_{s_t} = \bar{s}_t - \hat{s}_t \tag{27}$$

where $\hat{s}_t$ is as defined in (21) and $\bar{s}_t$ as in (26), respectively. This expression is the difference between the *incremental* best guess (c.f., our base-line approach) and the best guess using the full posterior.

If $\Delta_{s_t} = 0$, which is typically the case when *not* closing a loop, no backwards correction has to take place. When $\Delta_{s_t} \neq 0$, however, a shift occurred due to reconnection with a previously mapped area, and poses have to be revised backwards in time.

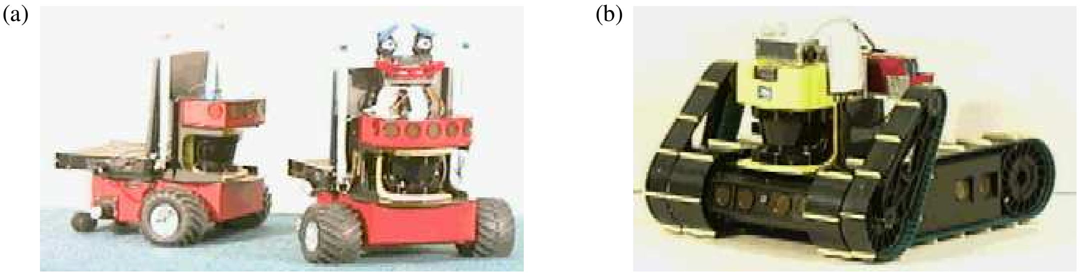


Figure 6: Robots: (a) Two of the pioneer robots used for multi-robot mapping. (b) Urban robot for indoor and outdoor exploration. The urban robot's odometry is extremely poor. All robots have been manufactured by RWI/ISR.

Our approach does this in three steps:

1. First, the size of the loop is determined by determining the scan in the map which led to the adjustment (this is a trivial side-result in the posterior computation).
2. Second, the error Δ_{s_t} is distributed *proportionally* among all poses in the loop. This computation does *not* yield a maximum likelihood match; however, it places the intermediate poses in a good starting position for subsequent gradient ascent search.
3. Finally, gradient ascent search is applied iteratively for all poses inside the loop, until the map is maximally consistent (maximizes likelihood) under this new constraint arising from the cycle.

These three steps implement an efficient approximation to the maximum likelihood estimator for the entire loop. Strictly speaking, this approach is not an online algorithm any longer, since the time required for backwards correction is not constant. A remedy might be found by amortizing the costs of the loop over the time used to acquire the next one. However, we found that in practice the entire correction can well be performed between subsequent range measurements, for all environments discussed in the experimental result section of this paper.

5 Results for Single-Robot Mapping

We validated the basic algorithm in various different environments. The mapping algorithm was at the core of several demonstrations to Government officials in DARPA's Tactical Mobile Robot (TMR) program. The role of CMU's Mercator project was to demonstrate the feasibility of rapidly acquiring accurate maps of building interiors with mobile robots. Our experiments are aimed at illustrating the robustness and accuracy of our approach in environments that traditionally are difficult to map.

The majority of the results discussed below were obtained with the robots shown in Figure 6. Figure 6a shows two RWI Pioneer AT robots, which were modified by replacing two of the wheels with passive casters (the original skid-steering mechanism was unable to turn on carpet). Figure 6b shows a tracked robot known as *urban robot*, also developed by RWI/ISR. This highly versatile robot has been developed for the DARPA TMR program, for indoor and outdoor missions in rugged terrain.

Our first experiment is concerned with closing the cycle in cyclic environments. Figure 7 shows different stages of mapping. using the very same data as in Figure 4. Here, however, the posterior estimator is used to detect and correct inconsistencies when closing the cycle. The particles are also shown in Figure 7, along with the most likely robot pose. As is easily seen, the resulting map is now consistent. Moments before the cycle is closed, the accumulated error is approximately 50 centimeters, as shown in Figure 7b. When closing the cycle, the most likely pose is adjusted in accordance to the previously mapped area, thereby practically eliminating this error. The backwards correction mechanisms then distributes the adjustment along the cycle, and the new, improved map is generated in less than a second of processing time. Figure 7c shows the final map.

To test the robustness of the mapping algorithm in the extreme, we manually deprived the data of its odometry information. Hence, only the laser range data can be used to estimate the robot's pose. Odometry-free mapping tests the robustness of the algorithm. If an algorithm can generate a consistent map without odometry data, it can accommodate failure of a robot's wheel encoders or INU (inertial navigation unit). Clearly, odometry-free mapping can only succeed if the environment possesses sufficiently distinguishing features. For example, without odometry it would be impossible to map a wide open area where all obstacles are beyond the reach of the sensors, or very long, featureless hallways.

Figure 8 shows the result of mapping the cyclic environment *without odometry*. Figure 8a depicts the raw data which, without odometry, is difficult to interpret. The map in Figure 8b has been generated using the statistical technique described in this paper. In particular, the same set of parameters were used to construct this map, as if the odometry data was available. This sheds light on the robustness of our approach.

Mapping with poor odometry has practical significance, as not all mobile robots have good odometry. A particularly inaccurate robot is the urban robot shown in Figure 6b. Figure 9a depicts a raw data set gathered while the robot was autonomously exploring and mapping a military facility in Fort Sam Houston in Texas. While the odometry is quite accurate when the robot moves straight, its tracks introduce up to 100% rotational error when the robot rotates. Thus, odometry alone is unusable even over very short time intervals. The map of this non-cyclic environment is shown in Figure 9b, along with the robot's exploration path. This map is not perfect, as attested by the fact that the walls are not perfectly aligned. However, is is sufficiently accurate for navigation.

A final result was obtained in a public place, the Tech Museum in San Jose, California. A key feature of this environment is the availability of an accurate blueprint, which enables us to evaluate the accuracy of the map acquired by the robot. The specific map was acquired with a RWI B21 robot. Figure 10a shows an occupancy grid map [15, 44] of a large center area in the museum, extracted from the aligned scans. A comparison with the architectural blueprint of the museum in Figure 10b illustrates that the learned map is quite

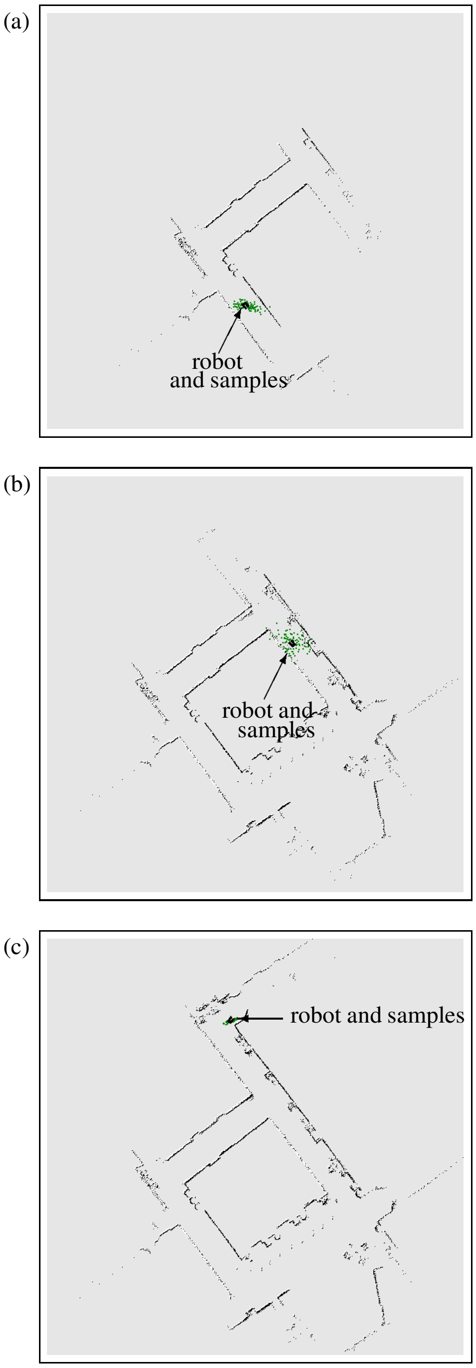


Figure 7: Incremental algorithm for concurrent mapping and localization. The dots, centered around the robot, indicate the posterior belief which grows over time (a and b). When the cycle is closed as in (c), the posterior becomes small again.

Figure 8: Mapping without odometry. Left: Raw data, right: map, generated online.

accurate. Several differences between these maps stem from the fact that the blueprint does not accurately reflect the nature and location of the exhibits in the museum. In this regard, the map learned by the robot is more accurate than the museum's blueprint.

6 Multi-Robot Mapping

The extension of our algorithm to multi-robot mapping is quite straightforward. In the multi-robot mapping problem, multiple robots gather data concurrently in an attempt to build a single, unified map of the environment. The use of multiple robots is advantageous if the speed at which a map is being constructed is important: In the best case, the speed-up by using N robots instead of one is super-unitary: In theory, environments exist where N robots consume approximately $\frac{1}{2N}$ the time it would take a single robot to map the environment. The factor 2 arises from the fact that in some environments, a single robot might waste time traversing known territory to get back to places not yet visited. In practice, however, the speedup is usually sub-linear [61].

Our current implementation makes two important restrictive assumptions:

- First, the robots must begin their operation in nearby locations, so that their range scans show substantial overlap.
- Second, the software must be told the approximate relative initial pose of the robots. The accuracy of the initial estimate is bound by the basin of attraction of the gradient ascent algorithm. A safe error is usually up to 50 cm and 20 degrees in orientation.

Under these conditions, the extension to multi-robot mapping is straightforward. In particular, each robot maintains a local, robot-specific estimate of its pose, while sharing the same

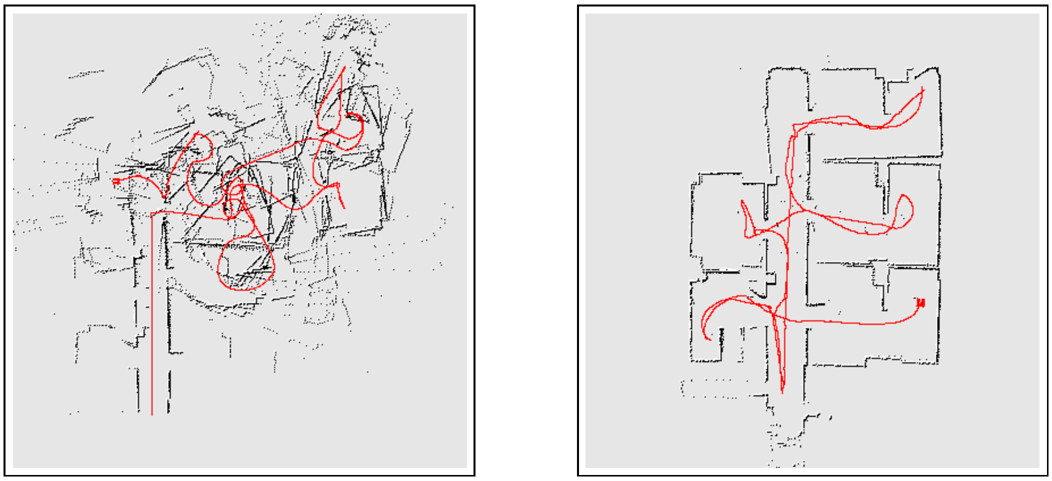


Figure 9: Autonomous exploration and mapping using the urban robot: Raw data and final map, generated in teal-time during exploration.

map. The estimation of the most likely pose using the incremental maximum likelihood estimator is based on a single, global map that fuses data collected by all robots.

Put mathematically, if we use k to index each particular robot, the posterior over the k-th robot pose is given by

$$p(s_{t,k}|d_k^t, m) \quad = \quad p(z_{t,k}|s_{t,k}, m) \int p(s_{t,k}|u_{t-1,k}, s_{t-1,k}) \; p(s_{t-1,k}|d_k^{t-1}, m) \; ds_{t-1,k} \tag{28}$$

This equation generalizes Equation (24) to the multi-robot case with a single global map m. The most likely pose of the k-th robot is then determined by generalizing Equation (26):

$$\bar{s}_{t,k} \quad = \quad \underset{s_{t,k}}{\operatorname{argmax}} \, p(s_{t,k}|d_k^t, \hat{m}(s_k^{t-1}, z_k^{t-1})) \tag{29}$$

where the function $\hat{m}$ incrementally grows the global map m. This extensions exploit the fact that none of the underlying statistics crucially depends on the number of robots. It also exploits an interesting conditional independence that enables us to factorize the posterior distribution of the joint pose space. In particular, the commitment to maximum likelihood maps eliminates possible dependencies between the pose estimates of different robots, which otherwise exist for the full Bayesian solution. Thus, in the general context, the posterior over robot poses would have to be estimated jointly, but here we can estimate it locally on each robot (see [18] for a more detailed discussion). We notice that the pose estimation can be carried out locally, on each robot. This leads to a computationally elegant decomposition of the problem, in which each robot basically performs the same basic algorithm as in single-robot mapping, but map updates are broadcast to all robots.

Figure 11 shows a map built by two autonomously exploring robots. This experiment was carried out in a TMR demonstration that took place in a training facility for firefighters

Figure 10: (a) Occupancy grid map and (b) architectural blueprint of a recently constructed museum in San Jose. Notice that the blueprint is inaccurate in certain places.

in Rockville, Maryland. The robots coordinate their exploration using a technique described in [4, 61]. In essence, this technique assigns robots to unexplored areas (frontiers) based on information-theoretic considerations. The map shown in Figure 11 is constructed online, as the robots explore, using the distributed software system outlined above. As is easily seen in Figure 11a through e, the robot quickly map the confined operating space. Finally, they return to locations that were not mapped perfectly in the first traversal of the building. In this example, the entire operation took a little less than two minutes, and the map was generated online as the robot moved. Figure 12 shows the result of a similar experiment, carried out at Fort Sam Houston. This experiment involved three robots, marked A, B, and C, in an environment more than three times the size. The resulting map was generated in less than three minutes. The paths in Figure 12 show how the robots systematically explore the environment and succeed in building a single, consistent map—despite the poor accuracy of the urban robot (robot B in the figure).

It is possible—though currently not part of our implemented software system—to relax the restrictive assumption of known relative initial poses. If the initial pose of robots relative to each other is unknown, the robots face a global localization problem, which is the problem of determining one a pose relative a given map. One natural way to solve the problem of determining one robot's pose relative to another is to globally localize one of the robots in the map built by the other robot. It is well-known [10, 18] that Monte Carlo localization (MCL)—which our approache uses for posterior pose estimation—can solve the global localization problem. It does so under one restrictive assumption: That the robot actually be inside the map in which it is being localized. While this assumption is trivially the case for the type problems studied in the localization literature, it is problematic in the context of

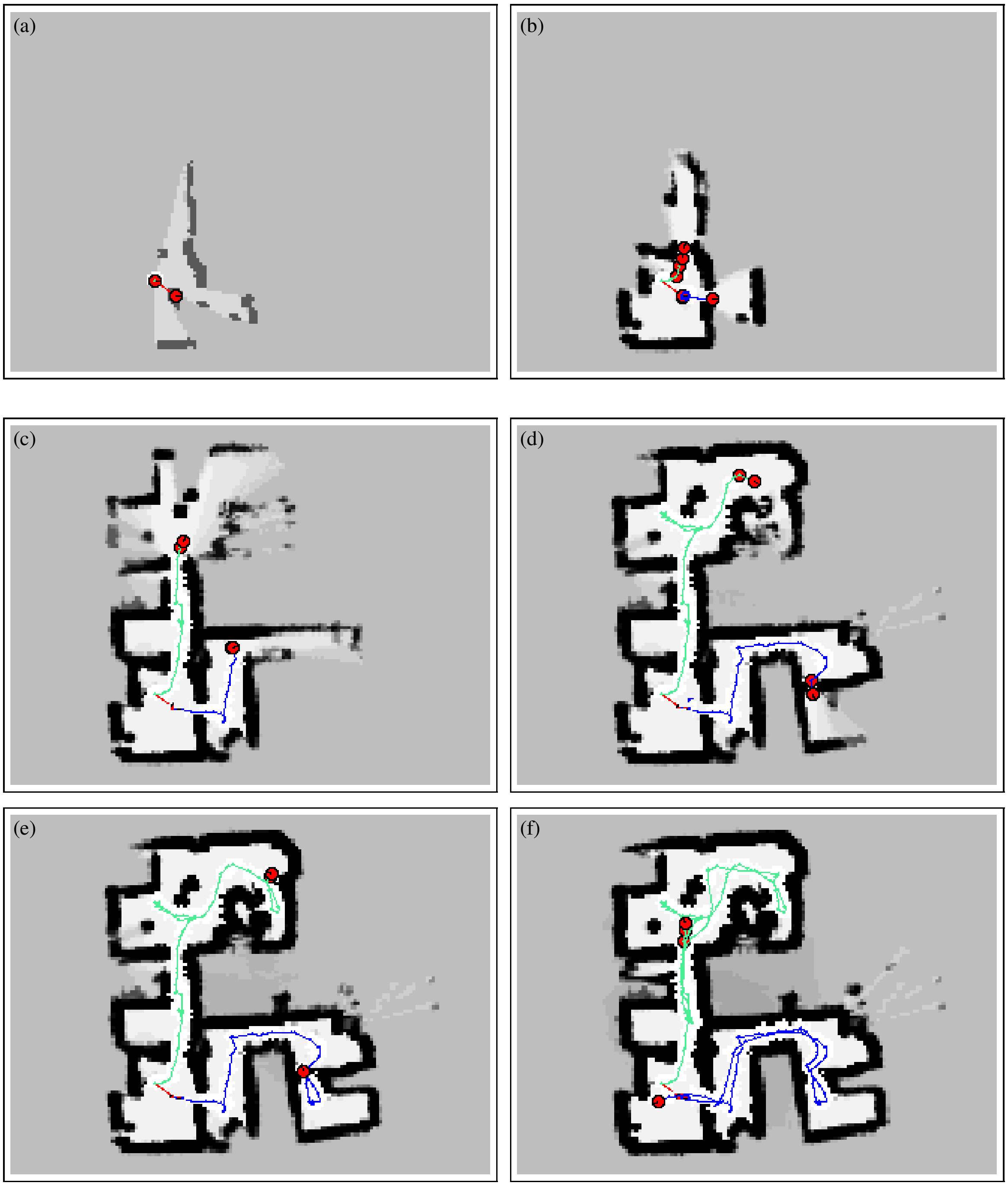


Figure 11: Online mapping during exploration with two robots.

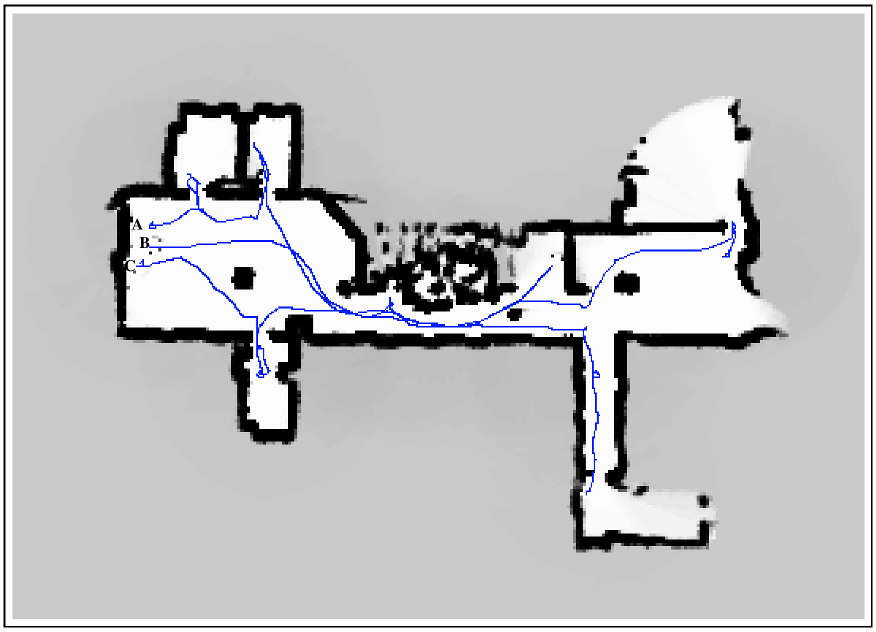


Figure 12: Map built by three autonomously exploring robots. The initial robot poses are on the left as marked by the letters A, B, and C. The robots A and C are pioneers, and the robot B is an urban robot.

multi-robot mapping. In particular, two robots might start out in different, non-overlapping parts of a building, and hence none of the robots operates inside the map built by any of the other robots. Thus, to estimate the poses of robot relative to each other, the robots must be able to estimate the degree of overlap between the areas traversed thus far. Clearly, MCL provides no answer to this hard and important problem.

However, MCL can be applied if it is known that one robot starts in the map built by another. Such a situation is shown in Figure 13. Here one robot is declared the *team leader*, which builds an initial, partial map. The second robot is then placed somewhere *inside* this map, but its pose inside this map is unknown. The particle filter of the second robot is then initialized by a set of particles drawn uniformly across the map of the first robot. The exact same posterior estimator depicted in Table 1 is then applied to globally localize the second robot in the map built by the team leader. Figure 13a shows the posterior after integrating one measurement. In this specific example, the samples have been generated in proximity to the first robot's path, assuming the second robot starts out near a location where the team leader has been before. A few moments later, the relative pose has been determined (Figure 13b) and the robot begins contributing data to the same, global map (Figure 13c).

7 3D Structural and Texture Mapping

A second extension of the basic mapping algorithm is the idea of three-dimensional maps. Such maps can be informative to people interested in a virtual view of a building interior.

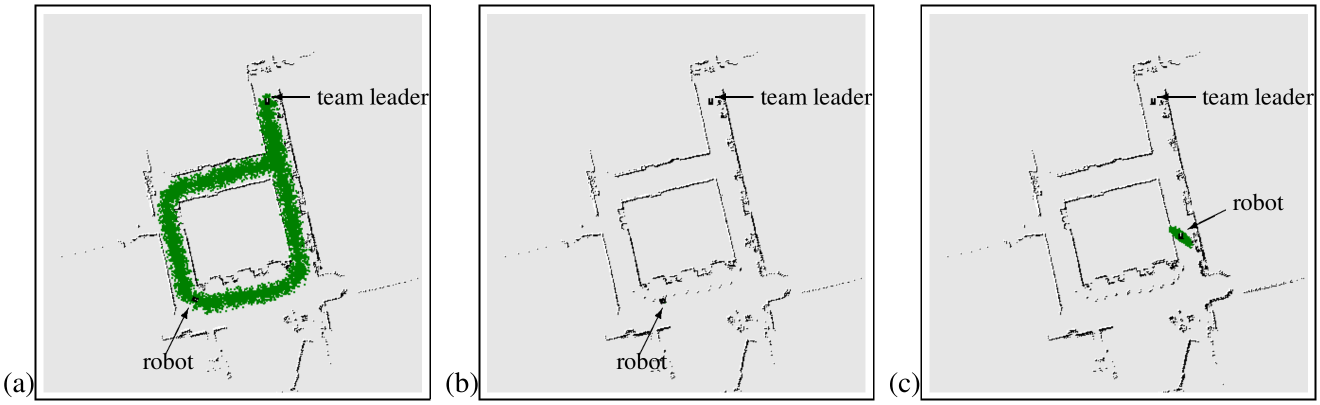


Figure 13: A second robot localizes itself in the map of the first, then contributes to building a single unified map. In (a), the initial uncertainty of the relative pose is expressed by a uniform sample in the existing map. The robot on the left found its pose in (b), and then maintains a sense of location in (c).

For example, it enables to pay a virtual visit using VR interfaces without actually physically entering the building. This might be useful for architects, real estate agents, and also for people preparing to operate in hazardous environments such as nuclear waste sites.

Figure 14a shows a robot equipped with two laser range finders. One of the laser range finders is pointed forward, to generate a 2D map using the techniques described above. As a result, it provides the robot with accurate pose estimates. The second laser range finder is mounted vertically, so that the light plane is perpendicular to both the first range scanner, and the motion direction of the robot. As a result, this range scanner measures cross-sections of the building as the robot moves. If the robot traverses the environment only once—which is typically the case in exploration—the measurements of the upward-pointed laser do not overlap; hence, there is little hope of using it for localization. However, the data provides information about the 3D structure of the building.

The robot shown in Figure 14a is also equipped with a panoramic camera. The panoramic camera utilizes a small mirror, which provides an image that contains the area covered by the upward-pointed laser range scanner. The mirror is mounted only a few centimeters away from the optical axis of the laser range scanner. Careful calibration of both devices enable us to match range and color measurements along the measurement plane of the vertical range finder, basically ignoring the small disparity between the optical axes of the laser and the camera. Figure 14 shows a panoramic image. The curved line highlights a vertical cross-section of the building, which corresponds to the perceptual field of the upward-pointed laser range finder.

Our approach represents 3D maps as collections of polygons with texture superimposed. Our 3D mapping algorithm proceeds in five steps, all of which can be carried out online.

1. **Pose estimation:** Pose estimation is carried out in 2D, using the techniques described above. The vertical range data is then projected into global coordinates using the maximum likelihood pose $\bar{s}$.

2. **Raw data smoothing:** Local noise in the range measurements is reduced by local smooth-

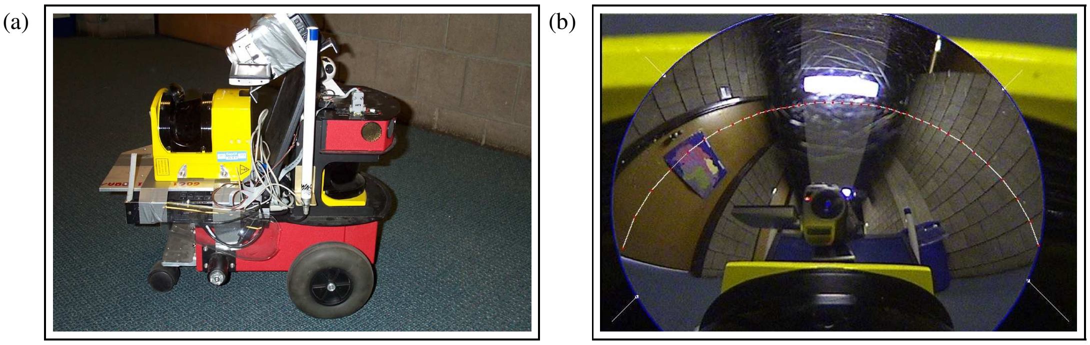


Figure 14: (a) Pioneer robot equipped with 2 laser range finders and panoramic camera used for 3D mapping. (b) Panoramic image acquired by the robot. Marked here is the region in the image that corresponds to the vertical slice measured by the laser range finder.

ing. Neighboring sensor scans are smoothed by replacing them with a locally weighted average, using a smoothing kernel of size 3 by 3. This step yields significantly smoother surfaces in the final map.

3. **Structural Mapping:** The map is then built by searching nearby range scans for nearby measurements. More specifically, for each time τ and index i, for which the coordinates of the four measurements

$$z_{\tau}^{[i]},\ z_{\tau}^{[i+1]},\ z_{\tau+}^{[i]},\ z_{\tau+}^{[i+1]} \tag{30}$$

   are less than δ apart in 3D coordinates, these points are assumed to correspond to the same objects. Hence, a polygon is generated with the endpoints corresponding to those four measurements and added to the map. Notice that polygon's are only constructed for measurements recorded right after another, and for neighboring sensor beams. The test of adjacency in 3D space is essential, since large discontinuities often correspond to open space (e.g., an open door).

4. **Texture Mapping:** If a polygon is generated, the image sequence recorded between the times τ and $\tau + 1$ are used as texture for the polygon. In particular, our algorithm extracts from the camera image a stripe that corresponds to the perceptual range of the laser. Stripes are then concatenated, which generates a texture image for the entire range scan, and the region corresponding to the area between the i-th and the $(i + 1)$-th range scan are extracted using a straightforward geometrical argument. This texture is then mapped onto the polygon.

5. **Map simplification:** To simplify the maps, the polygons are finally fused by a software package developed by computer graphics researchers [20, 21]. This approach, which our software uses without modification, fuses polygon's based on a quadratic error measures on their appearance in the rendered image. We found that the number of polygons can be

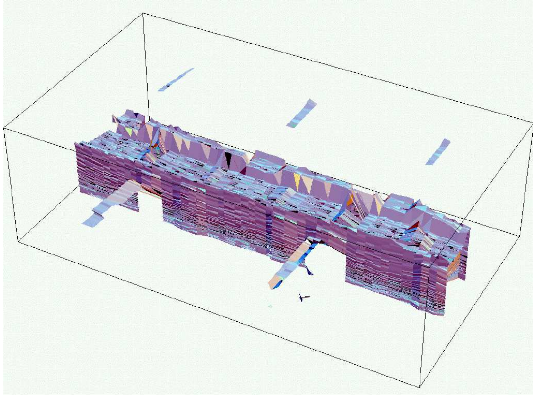


Figure 15: 3D Map.

reduced by a factor of ten without significantly altering the shape of the model. The result of this simplification step is, thus, a reduced complexity model that is also smoother.

The result of this procedure is a collection of textured polygons that can be rendered using a VRML viewer. While in principle, these steps can be carried out online, we remark that our current implementation is offline, due to the lack of a fast framegrabber on our mobile robot.

Figures 15 and 16 show views of a 3D structural map (without texture). The map in Figure 15 has been created from approximately 1 minute or raw data obtained in a straight corridor segment, whereas the ones in Figure 16 include a cycle and correspond to approximately 5 minutes of robot operation. All of these maps are in VRML format, enabling a user to navigate interactively through the building. Figure 16 illustrates the effect of the map simplification step above. The top row shows an un-simplified map, which contains 82,899 polygons. The bottom row is the simplified polygonal model, which contains only 8,289 polygons (10%). The appearance of both is similar; however, rendering the more compact one is an order of magnitude faster.

Figure 17 shows synthesized views of a structural map with texture superimposed, of a hallway segment in an office building. All views in this figure are rendered from the 3D model, using a VRML viewer. Figure 17a shows a view from the outside with the texture of the inside walls. Such a view is unattainable in the physical building. The other three renderings show views from the inside. One of the most notable errors are the lids of the two trashbins, which appear to be sloped. The robot is smaller than those trashbins and therefore unable to see the top. The sloped lid, thus, is the result of the polygonal interpolation. The map in Figure 17 did *not* undergo the map simplification step above and possesses 80,470 polygons.