Machine Learning 33, 41–76 (1998)


# Bayesian Landmark Learning for Mobile Robot Localization

SEBASTIAN THRUN http://www.cs.cmu.edu/∼thrun
*Computer Science Department and Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213-3891*

**Editor:** Pat Langley

**Abstract.** To operate successfully in indoor environments, mobile robots must be able to localize themselves. Most current localization algorithms lack flexibility, autonomy, and often optimality, since they rely on a human to determine what aspects of the sensor data to use in localization (e.g., what landmarks to use). This paper describes a learning algorithm, called BaLL, that enables mobile robots to learn what features/landmarks are best suited for localization, and also to train artificial neural networks for extracting them from the sensor data. A rigorous Bayesian analysis of probabilistic localization is presented, which produces a rational argument for evaluating features, for selecting them optimally, and for training the networks that approximate the optimal solution. In a systematic experimental study, BaLL outperforms two other recent approaches to mobile robot localization.

**Keywords:** artificial neural networks, Bayesian analysis, feature extraction, landmarks, localization, mobile robots, positioning

## 1. Introduction

To operate autonomously, mobile robots must know where they are. *Mobile robot localization*, that is the process of determining and tracking the position (location) of a mobile robot relative to its environment, has received considerable attention over the past few years. Accurate localization is a key prerequisite for successful navigation in large-scale environments, particularly when global models are used, such as maps, drawings, topological descriptions, and CAD models (Kortenkamp, Bonassi, & Murphy, in press). As demonstrated by a recent survey of localization methods by Borenstein, Everett, & Feng (1996), the number of existing approaches is diverse. Cox (1991) noted that "Using sensory information to locate the robot in its environment is the most fundamental problem to providing a mobile robot with autonomous capabilities."

Virtually all existing localization algorithms extract a small set of *features* from the robot's sensor measurements. *Landmark-based approaches*, which have become very popular in recent years, scan sensor readings for the presence or absence of landmarks to infer a robot's position. Other techniques, such as most *model matching approaches*, extract certain geometric features such as walls or obstacle configurations from the sensor readings, which are then matched to models of the robot's environment. The range of features used by different approaches to mobile robot localization is quite broad. They range from artificial markers such as barcodes and more natural objects such as ceiling lights and doors to geometric features such as straight wall segments and corners. This raises the question

as to what features might be the best ones to extract, in the sense that they produce the best localization results. Assuming that features correspond to landmarks[1] in the robot's environment, the questions addressed in this paper are: *What landmarks are best suited for mobile robot localization? Can a robot learn its own sets of features, can it define its own landmarks for localization, and can it learn optimal features?* The problem of learning the right landmarks has been recognized as a significant scientific problem in robotics (Borenstein, Everett, & Feng, 1996), artificial intelligence (Greiner & Isukapalli, 1994), and in cognitive science (Chown, Kaplan, & Kortenkamp, 1995).

Few localization algorithms enable a robot to learn features or to define its own landmarks. Instead, they rely on static, hand-coded sets of features for localization, which has three principle disadvantages:

1. **Lack of flexibility.** The usefulness of a specific feature depends on the particular environment the robot operates in and also often hinges on the availability of a particular type of sensors. For example, the landmark "ceiling light"—which has been used successfully in several mobile robot applications—is useless when the environment does not possess ceiling lights, or when the robot is not equipped with the appropriate sensor (such as a camera). If the features are static and predetermined, the robot can localize itself only in environments where those features are meaningful, and with sensors that carry enough information for extracting them.
2. **Lack of optimality.** Even if a feature is generally applicable, it is usually unclear how good it is or what the *optimal* landmark would be. Of course, the goodness of features depends, among other things, on the environment the robot operates in and the type of uncertainty it faces. Existing approaches usually do not strive for optimality, which can lead to brittle behavior.
3. **Lack of autonomy.** For a human expert to select appropriate features, he/she has to be knowledgeable about the characteristics of the robot's sensors and its environment. Consequently, it is often not straightforward to adjust an existing localization approach to new sensors or to new environments. Additionally, humans might be fooled by introspection. Since the human sensory apparatus differs from that of mobile robots, features that appear appropriate for human orientation are not necessarily appropriate for robots.

These principal deficiencies are shared by most existing localization approaches (Borenstein, Everett, & Feng, 1996).

This paper presents an algorithm, called BaLL (short for *Bayesian landmark learning*), that lets a robot *learn* such features, along with routines for extracting them from sensory data. Features are computed by artificial neural networks that map sensor data to a lower-dimensional feature space. A rigorous Bayesian analysis of probabilistic mobile robot localization quantifies the average posterior error a robot is expected to make, which depends on the features extracted from the sensor data. By training the networks so as to minimize this error, the robot learns features that directly minimize the quantity of interest in mobile robot localization (see also Greiner & Isukapalli, 1994).

We conjecture that the learning approach proposed here is more *flexible* than static approaches to mobile robot localization, since BaLL can automatically adapt to the particular environment, the robot, and its sensors. We also conjecture that BaLL will often yield *better*

*results* than static approaches, since it directly chooses features by optimizing their utility for localization. Finally, BaLL increases the *autonomy* of a robot, since it requires no human to choose the appropriate features; instead, the robot does this by itself. The first and the third conjecture follow from the generality of the learning approach. The second conjecture is backed with experimental results which illustrate that BaLL yields significantly better results than two other approaches to localization.

Section 2 introduces the basic probabilistic localization algorithm, which has in large parts been adopted from various successful mobile robot control systems. Section 3 formally derives the posterior error in localization and Section 4 derives a neural network learning algorithm for minimizing it. An empirical evaluation and comparison with two other approaches is described in Section 5, followed by a more general discussion of related work in Section 6. Section 7 discusses the implications of this work and points out interesting directions for future research.

## 2. A probabilistic model of mobile robot localization

This section lays the groundwork for the learning approach presented in Section 3, providing a rigorous probabilistic account on mobile robot localization. In a nutshell, probabilistic localization alternates two steps:

1. **Sensing.** At regular intervals, the robot queries its sensors. The results of these queries are used to refine the robot's internal belief as to where in the world it is located. Sensing usually decreases the robot's uncertainty.
2. **Acting.** When the robot executes an action command, its internal belief is updated accordingly. Since robot motion is inaccurate due to slippage and drift, it increases the robot's uncertainty.

The derivation of the probabilistic model relies on the assumption that the robot operates in a partially observable Markov environment (Chung, 1960) in which the only "state" is the location of the robot. In other words, the Markov assumption states that noise in perception and control is independent of noise at previous points in time. Various other researchers, however, have demonstrated empirically that the probabilistic approach works well even in dynamic and populated environments, due to the robustness of the underlying probabilistic representation (Burgard et al., 1996a; Kaelbling, Cassandra, & Kurien, 1996; Leonard, Durrant-Whyte, & Cox, 1992; Koenig & Simmons, 1996; Kortenkamp & Weymouth, 1994; Nourbakhsh, Powers, & Birchfield, 1995; Simmons & Koenig, 1995; Smith & Cheeseman, 1985; Smith, Self, & Cheeseman, 1990; Thrun, 1996).

### *2.1. Robot motion*

BaLL employs a probabilistic model of robot motion. Let $\xi$ denote the location of the robot within a global reference frame. Throughout this paper, the term *location* will be used to refer to three variables: the robot's $x$ and $y$ coordinates and its heading direction $\theta$. Although physically a robot always has a unique location $\xi$ at any point in time, internally

it only has a belief as to where it is located. BaLL describes this belief by a probability density over all locations $\xi \in \Xi$, denoted by

$$Bel(\xi), \tag{1}$$

where $\Xi$ denotes the space of all locations. Occasionally we will distinguish the belief *before* taking a sensor snapshot, denoted by $Bel_{\text{prior}}(\xi)$, and the belief after incorporating sensor information, denoted by $Bel_{\text{posterior}}(\xi)$. The problem of localization is to approximate as closely as possible the "true" distribution of the robot location, which has a single peak at the robot's location and is zero elsewhere.

Each motion command (e.g., translation, rotation) changes the location of the robot. Expressed in probabilistic terms, the effect of a motion command $a \in A$, where $A$ is the space of all motion commands, is described by a transition density

$$P(\xi \mid \tilde{\xi}, a), \tag{2}$$

which specifies the probability that the robot's location is $\xi$, given that it was previously at $\tilde{\xi}$ and that it just executed action $a$. In practice it usually suffices to know a pessimistic approximation of $P(\xi \mid \tilde{\xi}, a)$, which can easily be derived from the robot's kinematics/dynamics.

If the robot would *not* use its sensors, it would gradually lose information as to where it is due to slippage and drift (i.e., the entropy of $Bel(\xi)$ would increase). Incorporating sensor readings counteracts this effect, since sensor measurements convey information about the robot's location.

### *2.2. Sensing*

Let $S$ denote the space of all sensor measurements (sensations) and let $s \in S$ denote a single sensation, where sensations depend on the location $\xi$ of the robot. Let

$$P(s \mid \xi) \tag{3}$$

denote the probability that $s$ is observed at location $\xi$. In practice, computing meaningful estimates of $P(s \mid \xi)$ is difficult in most robotic applications. For example, if one robot's sensors include a camera, $P(s \mid \xi)$ would be a high-dimensional density capable of determining the probability of every possible camera image that could potentially be taken at any location $\xi$. Even if a full-blown model of the environment is available, computing $P(s \mid \xi)$ will be a complex, real-time problem in computer graphics. Moreover, the current work does not assume that a model of the environment is given to the robot; hence, $P(s \mid \xi)$ must be estimated from data.

To overcome this problem, it is common practice to extract (filter) a lower-dimensional feature vector from the sensor measurements. For example, landmark-based approaches scan the sensor input for the presence or absence of landmarks, neglecting all other information contained therein. Model-matching approaches extract partial models such as geometric maps from the sensor measurements, which are then compared to an existing

model of the environment. Only the result of this comparison (typically a single value) is then considered further.

To formally model the extraction of features from sensor data, let us assume sensor data are projected into a smaller space $F$, and the robot is given a function

$$\sigma : S \longrightarrow F, \tag{4}$$

which maps sensations $s \in S$ into features $f \in F$. Borrowing terms from the signal processing literature, $\sigma$ will be called a *filter*, and the result of filtering a sensor reading $f = \sigma(s)$ will be called a *feature vector*. Instead of having to know $P(s \mid \xi)$, it now suffices to know

$$P(f \mid \xi), \tag{5}$$

where $P(f \mid \xi)$ relates the sensory features $f = \sigma(s)$ to different locations of the environment, for wnich reason it is often called a *map of the environment.* The majority of localization approaches described in the literature assumes that the map is given (Borenstein, Everett, & Feng, 1996). The probability $P(f \mid \xi)$ can also be learned from examples. $P(f \mid \xi)$ is often represented by a piecewise constant function (Buhmann et al., 1995; Burgard et al., 1996a; Burgard et al., 1996b; Kaelbling, Cassandra, & Kurien, 1996; Koenig & Simmons, 1996; Moravec & Martin, 1994; Nourbakhsh, Powers, & Birchfield, 1995; Simmons & Koenig, 1995), or a parameterized density such as a Gaussian or a mixture of Gaussians (Gelb, 1974; Rencken, 1995; Smith & Cheeseman, 1985; Smith, Self, & Cheeseman, 1990). Below, in our experimental comparison, a $k$-nearest neighbor algorithm will be used to represent $P(f \mid \xi)$.

In landmark-based localization, for example, $\sigma$ filters out information by recording only the presence and absence of individual landmarks, and $P(f \mid \xi)$ models the likelihood of observing a landmark at the various locations $\xi$. $P(f \mid \xi)$ can be estimated from data. The mathematically inclined reader may notice that the use of $\sigma(s)$ instead of $s$ is mathematically justified only if $\sigma$ is a *sufficient statistic* (Vapnik, 1982) for estimating location—otherwise, all approaches that filter sensor data may yield sub-optimal results (by ignoring important sensor information). In practice, the suboptimality is tolerated, since $P(f \mid \xi)$, or an approximate version of $P(f \mid \xi)$, is usually much easier to obtain than $P(s \mid \xi)$, and often is a good approximation to this probability.

### *2.3. Robot localization*

For reasons of simplicity, let us assume that at any point in time $t$, the robot queries its sensors and then executes an action command that terminates at time $t + 1$. In response to the sensor query, the robot receives a sensor reading $s^{(t)}$, from which it extracts a feature vector $f^{(t)}$. Let $f^{(1)}, f^{(2)}, \ldots = \sigma(s^{(1)}), \sigma(s^{(2)}), \ldots$ denote the sequence of feature vectors, and let $a^{(1)}, a^{(2)}, \ldots$ denote the sequence of actions. Furthermore, let $\xi^{(0)}, \xi^{(1)}, \ldots$ denote the sequence of robot locations. Occasionally, locations will annotated by a $^*$ to distinguish them from variables used for integration.

Initially, at time $t = 0$, the robot has a *prior belief* as to what its location might be; this prior belief is denoted $Bel_{\text{prior}}(\xi^{(0)})$ and reflects the robot's initial uncertainty. If the robot knows its initial location and the goal of localization is to compensate slippage and drift, $Bel_{\text{pri}}(\xi^{(0)})$ is a point-centered distribution that has a peak at the correct location. The corresponding localization problem is called *position tracking*. Conversely, if the robot has no initial knowledge about its position, $Bel_{\text{prior}}(\xi^{(0)})$ is a uniform distribution. Here the corresponding localization problem is called *self localization*, *global localization*, or the "kidnapped robot problem" (Engelson, 1994), a task that is significantly more difficult than position tracking.

Sensor queries and actions change the robot's internal belief. Expressed probabilistically, the robot's belief after executing the $t-1$th action is

$$Bel_{\text{prior}}\big(\xi^{(t)}\big) = P\big(\xi^{(t)} \mid f^{(1)}, a^{(1)}, f^{(2)}, a^{(2)}, \ldots, f^{(t-1)}, a^{(t-1)}\big) \tag{6}$$

and after taking the $t$-th sensor measurement it is

$$Bel_{\text{posterior}}\big(\xi^{(t)}\big) = P\big(\xi^{(t)} \mid f^{(1)}, a^{(1)}, f^{(2)}, a^{(2)}, \ldots, a^{(t-1)}, f^{(t)}\big). \tag{7}$$

We will treat these two cases separately, starting with the second one.

***2.3.1. Sensing.*** According to Bayes' rule,

$$\begin{aligned} Bel_{\text{posterior}}\big(\xi^{(t)}\big) &= P\big(\xi^{(t)} \mid f^{(1)}, \ldots, a^{(t-1)}, f^{(t)}\big) \\ &= \frac{P\big(f^{(t)} \mid \xi^{(t)}, f^{(1)}, \ldots, a^{(t-1)}\big) P\big(\xi^{(t)} \mid f^{(1)}, \ldots, a^{(t-1)}\big)}{P\big(f^{(t)} \mid f^{(1)}, \ldots, a^{(t-1)}\big)}. \end{aligned} \tag{8}$$

The Markov assumption states that sensor readings are conditionally independent of previous sensor readings and actions given knowledge of the exact location:

$$P\big(s^{(t)} \mid \xi^{(t)}\big) = P\big(s^{(t)} \mid \xi^{(t)}, s^{(1)}, a^{(1)}, \ldots, a^{(t-1)}\big). \tag{9}$$

Since $f^{(t)} = \sigma(s^{(t)})$, it follows that

$$P\big(s^{(t)} \mid \xi^{(t)}\big) = P\big(f^{(t)} \mid \xi^{(t)}, f^{(1)}, a^{(1)}, \ldots, a^{(t-1)}\big). \tag{10}$$

It is important to notice that the Markov assumption does not specify the independence of different sensor readings if the robot's location is unknown; neither does it make assumptions on the extent to which $\xi^{(t)}$ is known during localization. In mobile robot localization, the location is usually unknown—otherwise there would not be a localization problem—and subsequent sensor readings and actions usually depend on each other. See Chung (1960), Howard (1960), Mine & Osaki (1970), and Pearl (1988) for more thorough treatments of conditional independence and Markov chains.

The Markov assumption simplifies (8), which leads to the important formula (Moravec, 1988; Pearl, 1988):

$$Bel_{\text{posterior}}\big(\xi^{(t)}\big) = \frac{P\big(f^{(t)} \mid \xi^{(t)}\big) P\big(\xi^{(t)} \mid f^{(1)}, \ldots, a^{(t-1)}\big)}{P\big(f^{(t)} \mid f^{(1)}, \ldots, a^{(t-1)}\big)} = \frac{P\big(f^{(t)} \mid \xi^{(t)}\big) Bel_{\text{prior}}\big(\xi^{(t)}\big)}{P\big(f^{(t)} \mid f^{(1)}, \ldots, a^{(t-1)}\big)}. \tag{11}$$

The denominator on the right hand side of (11) is a normalizer which ensures that the belief $Bel_{\text{posterior}}(\xi^{(t)})$ integrates to 1. It is calculated as

$$P\big(f^{(t)} \mid f^{(1)}, \ldots, a^{(t-1)}\big) = \int_{\Xi} P\big(f^{(t)} \mid \xi^{(t)}\big) P\big(\xi^{(t)} \mid f^{(1)}, \ldots, a^{(t-1)}\big)\, d\xi^{(t)} = \int_{\Xi} P\big(f^{(t)} \mid \xi^{(t)}\big) Bel_{\text{prior}}\big(\xi^{(t)}\big) d\xi^{(t)}. \tag{12}$$

To summarize, the posterior belief $Bel_{\text{posterior}}(\xi^{(t)})$ after observing the $t$th feature vector $f^{(t)}$ is proportional to the prior belief $Bel_{\text{prior}}(\xi^{(t)})$ multiplied by the likelihood $P(f^{(t)} \mid \xi^{(t)})$ of observing $f^{(t)}$ at $\xi^{(t)}$.

***2.3.2. Acting.*** Actions change the location of the robot and thus its belief. Recall that the belief after executing the $t$th action is given by

$$Bel_{\text{prior}}\big(\xi^{(t+1)}\big) = P\big(\xi^{(t+1)} \mid f^{(1)}, \ldots, f^{(t)}, a^{(t)}\big), \tag{13}$$

which can be rewritten using the theorem of total probability as

$$\int_{\Xi} P\big(\xi^{(t+1)} \mid \xi^{(t)}, f^{(1)}, \ldots, f^{(t)}, a^{(t)}\big) P\big(\xi^{(t)} \mid f^{(1)}, \ldots, f^{(t)}, a^{(t)}\big)\, d\xi^{(t)}. \tag{14}$$

Since $\xi^{(t)}$ does not depend on the action $a^{(t)}$ executed there, (14) is equivalent to

$$\int_{\Xi} P\big(\xi^{(t+1)} \mid \xi^{(t)}, f^{(1)}, \ldots, f^{(t)}, a^{(t)}\big)\; P\big(\xi^{(t)} \mid f^{(1)}, \ldots, f^{(t)}\big)\, d\xi^{(t)}. \tag{15}$$

By virtue of the Markov assumption, which if $\xi^{(t)}$ is known renders conditional independence of $\xi^{(t+1)}$ from $f^{(1)}, a^{(1)}, \ldots, f^{(t)}$ (but not from $a^{(t)}$), $Bel_{\text{pri}}(\xi^{(t+1)})$ can be expressed as

$$\int_{\Xi} P\big(\xi^{(t+1)} \mid \xi^{(t)}, a^{(t)}\big)\; P\big(\xi^{(t)} \mid f^{(1)}, \ldots, f^{(t)}\big)\, d\xi^{(t)}$$

or

$$\int_{\Xi} P\big(\xi^{(t+1)} \mid \xi^{(t)}, a^{(t)}\big) Bel_{\text{posterior}}\big(\xi^{(t)}\big)\, d\xi^{(t)}. \tag{16}$$

*Table 1.* The incremental localization algorithm.

1. Initialization: $Bel(\xi) \longleftarrow Bel_{\text{prior}}(\xi^{(0)})$
2. For each observed feature vector $f = \sigma(s)$ do:

$$Bel(\xi) \longleftarrow P(f \mid \xi)\ Bel(\xi) \tag{17}$$

$$Bel(\xi) \longleftarrow Bel(\xi) \left[ \int_{\Xi} Bel(\tilde{\xi})\, d\tilde{\xi} \right]^{-1} \qquad \text{(normalization)} \tag{18}$$

3. For each action command $a$ do:

$$Bel(\xi) \longleftarrow \int_{\Xi} P(\xi \mid \tilde{\xi}, a)\ Bel(\tilde{\xi})\, d\tilde{\xi} \tag{19}$$

Put verbally, the probability of being at $\xi^{(t+1)}$ at time $t+1$ is the result of multiplying the probability of previously having been at $\xi^{(t)}$ with the probability that action $a^{(t)}$ carries the robot to location $\xi^{(t+1)}$, integrated over all potential locations $\xi^{(t)}$. The transition probability $P(\xi^{(t+1)} \mid \xi^{(t)}, a^{(t)})$ has been defined in (2) in Section 2.1.

### *2.4. The incremental localization algorithm*

Beliefs can be updated incrementally. This follows from the fact that the belief $Bel_{\text{posterior}}(\xi^{(t)})$ is obtained from the belief $Bel_{\text{prior}}(\xi^{(t)})$ just before sensing, using (11), and the belief $Bel_{\text{prior}}(\xi^{(t+1)})$ is computed from the belief $Bel_{\text{posterior}}(\xi^{(t)})$ just before executing an action command, using (16). The incremental nature of (11) and (16) lets us state the compact algorithm for probabilistic localization shown in Table 1. As can be seen in the table, to update $Bel(\xi)$ three probabilities must be known: $Bel_{\text{prior}}(\xi^{(0)})$, the initial estimate (uncertainty); $P(\xi \mid \tilde{\xi}, a)$, the transition probability that describes the effect of the robot's actions; and $P(f \mid \xi)$, the map of the environment.

Figure 1 provides a graphical example that illustrates the localization algorithm. Initially, the location of the robot is unknown except for its orientation. Thus, $Bel(\xi)$ is uniformly distributed over all locations shown in figure 1(a). The robot queries its sensors and finds out that it is next to a door. This information alone does not suffice to determine its position uniquely—partially because of the existence of multiple doors in the environment and partially because the feature extractor might err. As a result, $Bel(\xi)$ is large for door locations and small everywhere else, as shown in figure 1(b). Next, the robot moves forward, in response to which its density $Bel(\xi)$ is shifted and slightly flattened out, reflecting the uncertainty $P(\xi \mid \tilde{\xi}, a)$ introduced by robot motion, as in figure 1(c). The robot now queries its sensors once more and finds out that again it is next to a door. The resulting density, in figure 1(d) now has a single peak and is fairly accurate. The robot "knows" with high accuracy where it is.

Notice that the algorithm derived in this paper is a general instance of an updating algorithm for a partially observable Markov chain. For example, it subsumes Kalman filters (Kalman, 1960) when applied mobile robot localization (Smith, Self, & Cheeseman, 1990; Leonard, Durrant-Whyte, & Cox, 1992). It also subsumes hidden Markov models (Rabiner, 1989) if robot location is the only state in the environment, as assumed here and elsewhere. Due to its generality, our algorithm subsumes various probabilistic algorithms published in the recent literature on mobile robot localization and navigation (see Burgard et al.,

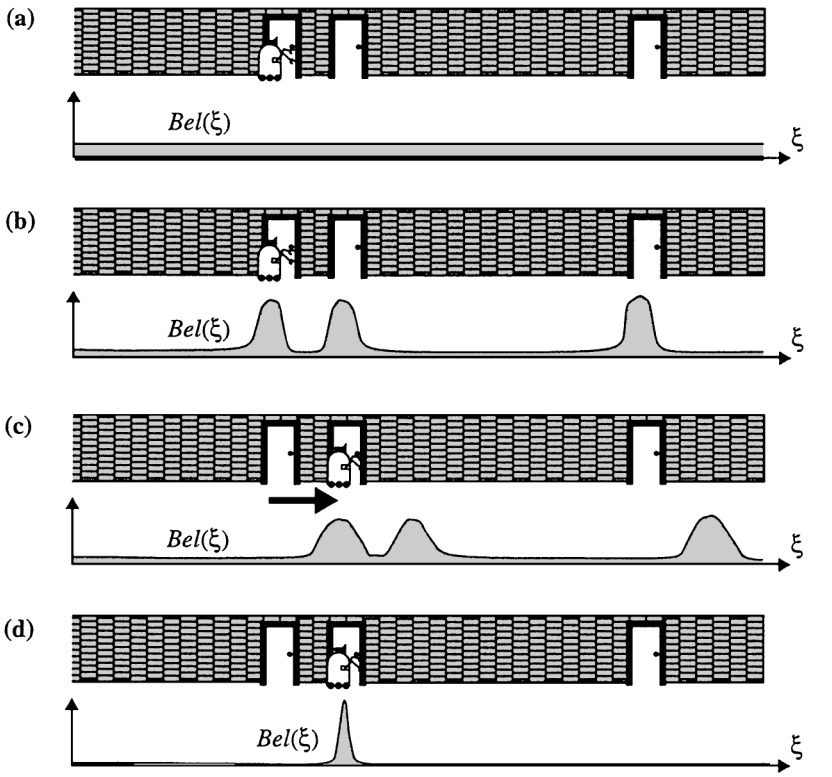


*Figure 1.* Probabilistic localization—an illustrative example. (a) Initially, the robot does not know where it is, hence $Bel(\xi)$ is uniformly distributed. (b) The robot observes a door next to it, and changes its belief accordingly. (c) The robot moves a meter forward; as a result, the belief is shifted and flattened. (d) The repeated observation of a door prompts the robot to modify its belief, which now approximates the "true" location well.

1996a; Kaelbling, Cassandra, & Kurien, 1996; Koenig & Simmons, 1996; Kortenkamp & Weymouth, 1994; Nourbakhsh, Powers, & Birchfield, 1995; Simmons, & Koenig, 1995; Smith, Self, & Cheeseman, 1990).

## 3. The Bayesian localization error

This section and the following one present BaLL, a method for learning $\sigma$. The input to the BaLL algorithm is a set of sensor snapshots labeled by the location at which they were taken:

$$X = \{\langle s_k, \xi_k \rangle \mid k = 1, \ldots, K\}, \tag{20}$$

where $K$ denotes the number of training examples.

Localization is a specific form of state estimation. As it is common practice in the statistical literature on state estimation (Vapnik, 1982; Casella & Berger, 1990), the effectiveness

of an estimator will be judged by measuring the expected deviation between estimated and true locations. BaLL learns $\sigma$ by minimizing this deviation.[2]

### 3.1. *The posterior error* $E_{\text{posterior}}$

The key to learning $\sigma$ is to minimize the localization error. To analyze this error, let us examine the update rule (17) in Table 1. This update rule transforms a prior belief to a refined, posterior belief, which is usually more accurate. Obviously, the posterior belief and thus the error depend on $\sigma$, which determines the information extracted from sensor data $s$.

Let $\xi^*$ denote the *true* location of the robot (throughout the derivation, we will omit the time index to simplify the notation), and let $e(\xi^*, \xi)$ denote an error function for measuring the error between the true position $\xi^*$ and an arbitrary other position $\xi$. The concrete nature of $e$ is inessential to the basic algorithm; for example, $e$ might be the Kullback-Leibler divergence or a metric distance.

The *Bayesian localization error at* $\xi^*$, denoted by $E(\xi^*)$, is obtained by integrating the error $e$ over all belief positions $\xi$, weighted by the likelihood $Bel(\xi)$ that the robot assigns to $\xi$, giving

$$E(\xi^*) = \int_{\Xi} e(\xi^*, \xi) Bel(\xi)\, d\xi. \tag{21}$$

If this error is computed prior to taking a sensor snapshot, that is, if $Bel(\xi) = Bel_{\text{prior}}(\xi)$, it is called the *prior Bayesian error at* $\xi^*$ with respect to the next sensor reading and will be denoted $E_{\text{prior}}$. The prior localization error is a function of $Bel_{\text{pri}}(\xi)$.

We are now ready to derive the Bayesian error *after* taking a sensor snapshot. Recall that $\xi^*$ denotes the true location of the robot. By definition, the robot will sense a feature vector $f$ with probability $P(f \mid \xi^*)$. In response, it will update its belief according to Eq. (17). The *posterior Bayesian error at* $\xi^*$, which is the error the robot is expected to make at $\xi^*$ after sensing, is obtained by applying the update rule (17) to the error (21), giving

$$\begin{aligned} E_{\text{posterior}}(\xi^*) &= \int_{\Xi} e(\xi^*, \xi) Bel_{\text{posterior}}(\xi)\, d\xi \\ &= \int_{\Xi} e(\xi^*, \xi) \int_{F} \frac{P(f \mid \xi) Bel_{\text{prior}}(\xi)}{P(f)} P(f \mid \xi^*)\, df\, d\xi, \end{aligned} \tag{22}$$

where $E_{\text{posterior}}$ is averaged over all possible sensor feature vectors $f$ weighted by their likelihood $P(f \mid \xi^*)$. The normalize $P(f)$ is computed just as in equations (12) or (18).

Thus far, the posterior error $E_{\text{posterior}}$ corresponds to a single position $\xi^*$ only. By averaging over all possible positions $\xi^*$, weighted by their likelihood of occurrence $P(\xi^*)$, we obtain the *average posterior error*

$$\begin{aligned} E_{\text{posterior}} &= \int_{\Xi} E_{\text{posterior}}(\xi^*) P(\xi^*)\, d\xi^* \\ &= \int_{\Xi} \int_{\Xi} e(\xi^*, \xi) \int_{F} \frac{P(f \mid \xi) Bel_{\text{prior}}(\xi)}{P(f)} P(f \mid \xi^*) P(\xi^*)\, df\, d\xi\, d\xi^*. \end{aligned} \tag{23}$$

Since $f = \sigma(s)$, expression (23) can be rewritten as

$$E_{\text{posterior}} = \int_{\Xi}\int_{\Xi} e(\xi^*, \xi) \int_{S} \frac{P(\sigma(s) \mid \xi) Bel_{\text{prior}}(\xi)}{P(\sigma(s))} P(\sigma(s) \mid \xi^*) P(\xi^*)\, ds\, d\xi\, d\xi^*, \tag{24}$$

where

$$P(\sigma(s)) = \int_{\Xi} P(\sigma(s) \mid \tilde{\xi}) Bel_{\text{prior}}(\tilde{\xi})\, d\tilde{\xi}. \tag{25}$$

The error $E_{\text{posterior}}$ is the exact localization error after sensing.

### 3.2. *Approximating* $E_{\text{posterior}}$

While $E_{\text{posterior}}$ measures the "true" Bayesian localization error, it cannot be computed in any but the most trivial situations (since solving the various integrals in (24) is usually mathematically impossible). However, $E_{\text{posterior}}$ can be approximated using the data. Recall that to learn $\sigma$, the robot is given a set of $K$ examples

$$X = \{\langle s_k, \xi_k \rangle \mid k = 1, \ldots, K\}, \tag{26}$$

where $X$ consists of $K$ sensor measurements $s_k$ that are labeled by the location $\xi_k$ at which they were taken. $X$ is used to approximate $E_{\text{posterior}}$ with the expression

$$\tilde{E}_{\text{posterior}} = \sum_{\langle \xi^*, s^* \rangle \in X} \sum_{\langle \xi, s \rangle \in X} e(\xi^*, \xi) \frac{P(\sigma(s) \mid \xi) Bel_{\text{prior}}(\xi)}{P(\sigma(s))} P(\sigma(s) \mid \xi^*) P(\xi^*), \tag{27}$$

where

$$P(\sigma(s)) = \sum_{\langle \tilde{\xi}, \tilde{s} \rangle \in X} P(\sigma(s) \mid \tilde{\xi}) Bel_{\text{prior}}(\tilde{\xi}). \tag{28}$$

Equation (27) follows directly from Eq. (24). The integration variables $\xi \in \Xi$ and $s \in S$, which are independent in (24), are collapsed into a single summation over all training patterns $\langle \xi, s \rangle \in X$ in (27). $\tilde{E}_{\text{posterior}}$ is a stochastic approximation of $E_{\text{posterior}}$, based on data, that converges uniformly to $E_{\text{posterior}}$ as the size of the data set $X$ goes to infinity.

Leaving problems of small sample sizes aside, $\tilde{E}_{\text{posterior}}$ lets the robot compare different $\sigma$ with each other: the smaller $\tilde{E}_{\text{posterior}}$, the better $\sigma$ for the purpose of localization. This alone is an important result, as it lets one *compare* two filters to each other.

The error $\tilde{E}_{\text{posterior}}$ is a function of the prior uncertainty $Bel_{\text{pri}}(\xi)$ as well. As a result, a specific $\sigma$ that is optimal under one prior uncertainty can perform poorly under another. This observation matches our intuition: when the robot is globally uncertain, it is usually advantageous to consider different features than when it knows its location within a small margin of uncertainty.

## 4. The BaLL algorithm

BaLL learns the filter $\sigma$ by minimizing $\tilde{E}_{\text{posterior}}$ through search in the space of filters $\sigma$, that is, by computing

$$\sigma = \operatorname*{argmin}_{\hat{\sigma} \in \Sigma} \tilde{E}_{\text{posterior}}(\hat{\sigma}), \tag{29}$$

where $\Sigma$ is a class of functions from which $\sigma$ is chosen. This section presents a specific search space $\Sigma$, for wnich it derives a gradient descent algorithm.

### 4.1. *Neural network filters*

BaLL realizes $\sigma$ by a collection of $n$ backpropagation-style feedforward artificial neural networks (Rumelhart, Hinton, & Williams, 1986). Each network, denoted by $g_i$ with $i = 1, \ldots, n$, maps the sensor data $s$ to a feature value in $(0, 1)$. More formally, we have

$$\sigma = (g_1, g_2, \ldots, g_n), \tag{30}$$

where for all $i = 1, \ldots, n$,

$$g_i : S \longrightarrow (0, 1) \tag{31}$$

is realized by an artificial neural network. The $i$th network corresponds to the $i$th feature, where $n$ is the dimension of the feature vector $f$.

Neural networks can approximate a large class of functions (Hornik, Stinchcombe, & White, 1989). Thus, there are many features that a neural network can potentially extract. To the extent that neural networks are capable of recognizing landmarks, our approach lets a robot automatically select its own and learn routines for their recognition.

### 4.2. *Stochastic filters*

At first glance, it might seem appropriate to define $f = (g_1(s), g_2, (s), \ldots, g_n(s))$, making the feature vector $f$ be the concatenated $n$-dimensional output of the $n$ neural networks. Unfortunately, such a definition would imply $F = (0, 1)^n$, which contains an infinite number of feature vectors $f$ (since neural networks produce real-valued outputs). If the sensor readings are noisy and distributed continuously, as is the case for most sensors used in today's robots, the chance is zero that two different sensations taken at the same location will generate the same feature vector $f$. In other words, if $f = (g_1(s), g_2, (s), \ldots, g_n(s))$, $F$ would be too large for the robot to ever recognize a previous location—a problem that specifically occurs when using real-valued function approximators as feature detectors.

Fortunately, there exists an alternative representation that has several nice properties. In the BaLL algorithm $F = \{0, 1\}^n$ and $|F| = 2^n$ (which is finite). Each neural network is interpreted as a *stochastic* feature extractor, which generates the value $f_i = 1$ with

probability $g_i(s)$ and the value $f_i = 0$ with probability $1 - g_i(s)$, giving

$$\begin{aligned} P(f_i = 1 \mid s) &= g_i(s) \\ P(f_i = 0 \mid s) &= 1 - g_i(s). \end{aligned} \tag{32}$$

We assume that the joint probability $P(f \mid s)$ is given by the product of the marginal probabilities $P(f_i \mid s)$:

$$P(f \mid s) = \prod_{i=1}^{n} P(f_i \mid s). \tag{33}$$

The stochastic setting lets $\sigma$ express confidence in its result by assigning probabilities to the different $f \in F$—a generally desirable property for a filter.

The stochastic representation has another advantage, which is important for the efficiency of the learning algorithm. As we show below, $\tilde{E}_{\text{posterior}}$ is differentiable in the output of the function approximator and hence in the weights and biases of the neural networks. Differentiability is a necessary property for training neural networks with gradient descent.

### *4.3. The neural network learning algorithm*

The new, stochastic interpretation of $\sigma$ requires that $E_{\text{post}}$ and its approximation $\tilde{E}_{\text{posterior}}$ be modified to reflect the fact that $\sigma$ generates a probability distribution over $F$ instead of a single $f \in F$. Following the theorem of total probability and using (23) as a starting point, $E_{\text{post}}$ is given by

$$\begin{aligned} E_{\text{posterior}} = \int_{\Xi}\int_{\Xi} e(\xi^*, \xi) \sum_{f=(0,\ldots,0)}^{f=(1,\ldots,1)} \frac{\int_S P(f \mid s) P(s \mid \xi) Bel_{\text{prior}}(\xi)\, ds}{P(f)} \\ \times \int_S P(f \mid s) P(s \mid \xi^*) P(\xi^*)\, ds\, d\xi\, d\xi^*, \end{aligned} \tag{34}$$

where

$$P(f) = \int_{\Xi}\int_{S} P(f \mid s) P(s \mid \tilde{\xi}) Bel_{\text{prior}}(\tilde{\xi})\, ds\, d\tilde{\xi}. \tag{35}$$

The approximation of this term is governed by

$$\begin{aligned} \tilde{E}_{\text{posterior}} &= \sum_{\langle \xi^*, s^* \rangle \in X} \sum_{\langle \xi, s \rangle \in X} e(\xi^*, \xi) \sum_{f=(0,\ldots,0)}^{f=(1,\ldots,1)} \frac{P(f \mid s) Bel_{\text{prior}}(\xi)}{P(f)} P(f \mid s^*) P(\xi^*) \\ &= \sum_{\langle \xi^*, s^* \rangle \in X} \sum_{\langle \xi, s \rangle \in X} e(\xi^*, \xi) P(\xi^*)\, Bel_{\text{prior}}(\xi) \sum_{f=(0,\ldots,0)}^{f=(1,\ldots,1)} \frac{P(f \mid s)}{P(f)} P(f \mid s^*), \end{aligned} \tag{36}$$

where

$$P(f) = \sum_{\langle \tilde{\xi}, \tilde{s} \rangle \in X} P(f \mid \tilde{s}) Bel_{\text{prior}}(\tilde{\xi}). \tag{37}$$

The mathematically inclined reader should notice that (24) and (27) are special cases of (34) and (36). They are equivalent if one assumes that $P(f \mid s)$ is deterministic, that is, if $P(f \mid s)$ is centered on a single $f$ for each $s$.

Armed with an appropriate definition of $\tilde{E}_{\text{posterior}}$, we are now ready to derive the gradient descent learning algorithm for training the neural network feature recognizers to minimize $\tilde{E}_{\text{posterior}}$. This is done by iteratively adjusting the *weights* and *biases* of the $i$th neural network, denoted by $w_{i\mu\nu}$, in the direction of the negative gradients of $\tilde{E}_{\text{posterior}}$:

$$w_{i\mu\nu} \leftarrow w_{i\mu\nu} - \eta \frac{\partial \tilde{E}_{\text{posterior}}}{\partial w_{i\mu\nu}}. \tag{38}$$

Here $\eta > 0$ is a learning rate, which is commonly used in gradient descent to control the magnitude of the updates. Computing the gradient in the right-hand side of (38) is a technical matter, as both $\tilde{E}_{\text{posterior}}$ and neural networks are differentiable:

$$\frac{\partial \tilde{E}_{\text{posterior}}}{\partial w_{i\mu\nu}} = \sum_{\langle \bar{\xi}, \bar{s} \rangle \in X} \frac{\partial \tilde{E}_{\text{posterior}}}{\partial g_i(\bar{s})} \frac{\partial g_i(\bar{s})}{\partial w_{i\mu\nu}}. \tag{39}$$

The second gradient on the right-hand side of (39) is the regular output-weight gradient used in the backpropagation algorithm, whose derivation we omit (see Hertz, Krogh, & Palmer, 1991; Rumelhart, Hinton, & Williams, 1986; Wasserman, 1989). The first gradient in (39) can be computed as

$$\begin{aligned}
\frac{\partial \tilde{E}_{\text{posterior}}}{\partial g_i(\bar{s})} &\overset{(36)}{=} \sum_{\langle \xi^*, s^* \rangle \in X} \sum_{\langle \xi, s \rangle \in X} e(\xi^*, \xi)\ P(\xi^*) Bel_{\text{prior}}(\xi) \\
&\quad \times \sum_{f_1=0}^{1} \sum_{f_2=0}^{1} \cdots \sum_{f_n=0}^{1} \frac{\partial}{\partial g_i(\bar{s})} \left[ \left( \prod_{i=1}^{n} P(f_i \mid s^*)\ P(f_i \mid s) \right) P(f)^{-1} \right] \qquad (40) \\
&= \sum_{\langle \xi^*, s^* \rangle \in X} \sum_{\langle \xi, s \rangle \in X} e(\xi^*, \xi) P(\xi^*) Bel_{\text{prior}}(\xi) \\
&\quad \times \sum_{f_1=0}^{1} \sum_{f_2=0}^{1} \cdots \sum_{f_n=0}^{1} \prod_{j \neq i} P(f_j \mid s^*) P(f_j \mid s) \\
&\quad \cdot \left[ \frac{\delta_{\xi^*, \bar{\xi}} P(f_i \mid s) + \delta_{\xi, \bar{\xi}} P(f_i \mid s^*)}{\sum_{\langle \tilde{\xi}, \tilde{s} \rangle \in X} \prod_{j=1}^{n} P(f_j \mid \tilde{s})} \right. \\
&\qquad \left. - \frac{P(f_i \mid s^*) P(f_i \mid s) \prod_{j \neq i} P(f_j \mid \bar{s}) Bel_{\text{prior}}(\bar{\xi})}{\left( \sum_{\langle \tilde{\xi}, \tilde{s} \rangle \in X} \prod_{j=1}^{n} P(f_j \mid \tilde{s}) \right)^2} \right] (2\delta_{f_i,1} - 1).
\end{aligned}$$

*Table 2.* BaLL, the algorithm for learning neural network filters $\sigma$.

**Input:** Data set $X = \{\langle s_k, \xi_k\rangle \mid k = 1, \ldots, K\}$, prior belief $Bel_{\text{prior}}(\xi)$.

**Output:** Optimized parameters (weights and biases) $w_{i\mu\nu}$ for the $n$ networks $g_1, \ldots, g_n$.

**Algorithm:**

1. Initialize the parameters $w_{i\mu\nu}$ of every network with small random values.
2. Iterate until convergence criterion is fulfilled:

2.1 For all $\langle \xi, s\rangle \in X$, compute the conditional probabilities

$$P(f_i \mid s) = \begin{cases} g_i(s) & \text{if } f_i = 1 \\ 1 - g_i(s) & \text{if } f_i = 0 \end{cases} \tag{41}$$

where $g_i(s)$ is the output of the $i$th network for input $s$ (cf. (32)).

2.2 Compute the error $\tilde{E}_{\text{posterior}}$ (cf. (36))

$$\tilde{E}_{\text{posterior}} = \sum_{\langle \xi^*, s^*\rangle \in X} \sum_{\langle \xi, s\rangle \in X} e(\xi^*, \xi) P(\xi^*)\, Bel_{\text{prior}}(\xi) \cdot \sum_{f_1=0}^{1} \sum_{f_2=0}^{1} \cdots \sum_{f_n=0}^{1} \left(\prod_{i=1}^{n} P(f_i \mid s^*) P(f_i \mid s)\right) \left[\sum_{\langle \tilde{\xi}, \tilde{s}\rangle \in X} \left(\prod_{i=1}^{n} P(f_i \mid \tilde{s})\right) Bel_{\text{prior}}(\tilde{\xi})\right]^{-1} \tag{42}$$

2.3 For all network parameters $w_{i,\mu,\nu}$, compute

$$\frac{\partial \tilde{E}_{\text{posterior}}}{\partial w_{i\mu\nu}} = \sum_{\langle \tilde{\xi}, \tilde{s}\rangle \in X} \frac{\partial g_i(\tilde{s})}{\partial w_{i\mu\nu}} \sum_{\langle \xi^*, s^*\rangle \in X} \sum_{\langle \xi, s\rangle \in X} e(\xi^*, \xi) P(\xi^*)\, Bel_{\text{prior}}(\xi) \cdot \sum_{f_1=0}^{1} \sum_{f_2=0}^{1} \cdots \sum_{f_n=0}^{1} \prod_{j \neq i} P(f_j \mid s^*) P(f_j \mid s)(2\delta_{f_i,1} - 1) \cdot \left[\frac{\delta_{\xi^*, \tilde{\xi}} P(f_i \mid s) + \delta_{\xi, \tilde{\xi}} P(f_i \mid s^*)}{\sum_{\langle \tilde{\xi}, \tilde{s}\rangle \in X} \prod_{j=1}^{n} P(f_j \mid \tilde{s})} - \frac{P(f_i \mid s^*) P(f_i \mid s) \prod_{j \neq i} P(f_j \mid \tilde{s})\, Bel_{\text{prior}}(\tilde{\xi})}{\left(\sum_{\langle \tilde{\xi}, \tilde{s}\rangle \in X} \prod_{j=1}^{n} P(f_j \mid \tilde{s})\right)^2}\right]. \tag{43}$$

The gradients $\frac{\partial g_i(\tilde{s})}{\partial w_{i\mu\nu}}$ are obtained with backpropagation (cf. (39) and (40)).

2.4 For all network parameters $w_{i,\mu,\nu}$, update (cf. (38))

$$w_{i\mu\nu} \leftarrow w_{i\mu\nu} - \eta \frac{\partial \tilde{E}_{\text{posterior}}}{\partial w_{i\mu\nu}}. \tag{44}$$

Here $\delta_{x,y}$ denotes the Kronecker symbol, which is 1 if $x = y$ and 0 if $x \neq y$. $P(f_j \mid s^*)$ is computed according to Eq. (32).

Table 2 describes the BaLL algorithm and summarizes the main formulas derived in this and the previous section. BaLL's input is the data set $X$ and a specific prior belief $Bel_{\text{pri}}(\xi)$. Below, we will train networks for different prior beliefs characterized by different entropies (i.e., degrees of uncertainty). The gradient descent update is repeated until one reaches a termination criterion (e.g., early stopping using a cross-validation set or pseudoconvergence of $E_{\text{posterior}}$), as in regular backpropagation (Hertz, Krogh, & Palmer, 1991).[3] BaLL differs from conventional backpropagation (supervised learning) in that no target values are generated for the outputs of the neural networks. Instead, the quantity of interest, $E_{\text{posterior}}$, is minimized directly. The output characteristics of the individual networks and, hence, the features they extract, emerge as a side effect of minimizing $E_{\text{post}}$.

The output of the BaLL algorithm is a set of filters specified by a set of weights and biases for the different networks. As noted above, $E_{\text{posterior}}$ and the resulting filter $\sigma$ depend on the uncertainty $Bel_{\text{prior}}(\xi)$. Below, when presenting experimental results, we will show that, in cases in which the uncertainty is small (the entropy of $Bel_{\text{prior}}(\xi)$ is low), quite different features are extracted than when the uncertainty is large. However, although the networks must be *trained* for a particular $Bel_{\text{prior}}(\xi)$, they can be *used* to estimate the location for arbitrary uncertainties $Bel_{\text{pri}}(\xi)$, but with degraded performance. It is therefore helpful, but not necessary, to train different networks for different prior uncertainties.

### *4.4. Algorithmic complexity*

The complexity of the learning and the performance methods must be analyzed separately. The localization algorithm described in Table 1 must be executed in real time, while the robot is in operation, whereas the learning algorithm described in Table 2 can be run offline. Our primary concern in the analysis is time complexity.

***4.4.1. Localization.*** The complexity of probabilistic localization (Table 1) depends on the representation of $P(f \mid \xi)$ and $Bel(\xi)$. In the worst case, processing a single sensor reading requires $O(Kn + nW)$ time, where $K$ is the training set size, $n$ is the number of networks and $W$ is the number of weights and biases in each neural network. Processing an action requires $O(K^2 n)$ time. Various researchers have implemented versions of the probabilistic localization algorithm that work in real time (Burgard et al., 1996a; Burgard, Fox, & Thrun, 1997; Kaelbling, Cassandra, & Kurien, 1996; Koenig & Simmons, 1996; Nourbakhsh, Powers, & Birchfield, 1995; Simmons & Koenig, 1995; Thrun et al., 1996; Thrun, 1996). Given the relatively small computational overhead of the existing implementations, scaling to larger environments is not problematic.

***4.4.2. Learning.*** BaLL requires $O(N2^n K^3 + NKnW)$ time, where $n$, $K$, and $W$ are the same as above, and where $N$ is the number of gradient descent iterations. If the number of training patterns is greater than both the number of inputs and the number of hidden units in each network, which is a reasonable assumption since otherwise the number of free parameters exceeds the number of training patterns by a huge margin, then $O(N2^n K^3)$ dominates $O(NKnW)$. Thus, under normal conditions, the training the networks requires $O(N2^n K^3)$ time. The constant factor is small (cf. Table 2). Most existing localization algorithms use only one or two features (e.g., one or two landmarks), indicating that even small values for $n$ work well in practice.

There are several ways to reduce the complexity of learning:

1. Instead of training all networks in parallel, they can also be trained one after another, similar to the way units are trained one after another in the cascade correlation algorithm (Fahlman & Lebiere, 1989). Sequential training would reduce worst-case exponential to linear complexity, since networks are trained one after another, which requires $O(NnK^3)$ time.
2. Compact representations for $P(f \mid \xi)$ and $Bel(\xi)$ can reduce the complexity significantly. For example, in Burgard et al. (1996a), Koenig & Simmons (1996), and Simmons

& Koenig (1995), the number of grid cells used to represent $P(f \mid \xi)$ and $Bel(\xi)$ is independent of the training set size. Using their representations, our learning algorithm would scale quadratically in the size of the environment and linearly in the size of the training set. In addition, coarse-grained representations such as the one reported by Koenig & Simmons (1996) and Simmons & Koenig (1995) can reduce the constant factor even further.

3. The learning algorithm in Table 2 interleaves one computation of $\tilde{E}_{\text{posterior}}$ and its derivatives with one update of the weights and biases. Since the bulk of processing time is spent computing $\tilde{E}_{\text{posterior}}$ and its derivatives, the overall complexity can be reduced by modifying the training algorithm so that multiple updates of the networks' parameters are interleaved with a single computation of $\tilde{E}_{\text{posterior}}$ and its derivatives. The necessary steps include:

   3.1. The network outputs $g_i(s)$ are computed for each training example $\langle s, \xi \rangle \in X$.
   3.2. The gradients of $\tilde{E}_{\text{posterior}}$ with respect to the network outputs $g_i(s)$ are computed (cf. (40)).
   3.3. For each training example $\langle s, \xi \rangle \in X$, "pseudo-patterns" are generated using the current network output in conjunction with the corresponding gradients, giving

$$\left\langle s \ , \ \ g_i(s) - \frac{\tilde{E}_{\text{posterior}}}{g_i(s)} \right\rangle . \tag{45}$$

   3.4. These patterns are fitted using multiple epochs of regular backpropagation.

   This algorithm approximates gradient descent, but it reduces the complexity by a constant factor.

In addition, modifications such as online learning, stochastic gradient descent, or higher-order methods such as momentum or conjugate gradient methods (Hertz, Krogh, & Palmer, 1991) yield further speedup. Little is currently known about principal complexity bounds that would apply here.

As noted above, learning $\sigma$ can be done offline and is only done once. With the modifications proposed here, one complexity of training is low-order polynomial (mostly linear) in $K$, $n$, $N$, and $W$. In the light of the modifications discussed here, scaling up our approach to larger environments, larger training sets, and more neural networks does not appear to be problematic.

In our implementation (see below), training the networks required between 30 min and 12 h on a 200 MHz Pentium Pro.

## 5. Empirical evaluation and comparison

This section presents some empirical results obtained with BaLL, using data obtained from a mobile robot equipped with a color camera and an array of sonar sensors, as shown in figure 2(a). To compare our approach with other state-of-the-art methods, we reimplemented two previously published approaches.

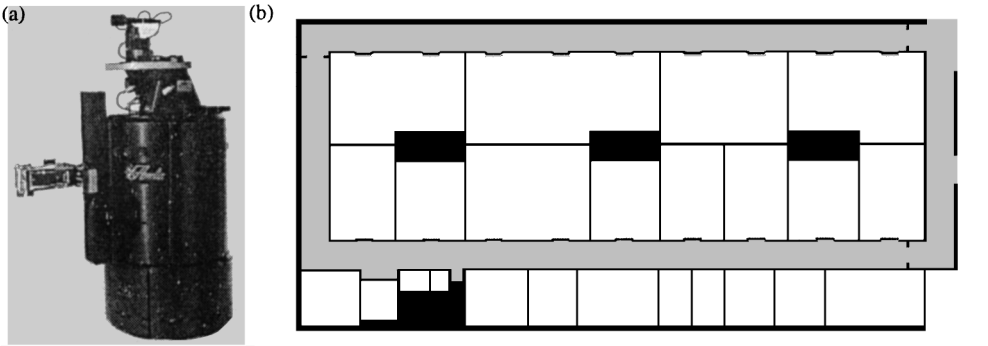


*Figure 2.* (a) The Real World Interface B21 robot used in our research. (b) The testing environment.

1. **Localization using doors.** A team of researchers at our university has recently developed a similar probabilistic localization method that uses doors as its primary landmark (see Koenig & Simmons, 1996; Simmons & Koenig, 1995). This group is interested in reliable long-term mobile robot operation, for which reason it has operated an autonomous mobile robot almost on a daily basis over the last two years, in which the robot moved more than 110 km in more than 130 hours. Since we are located in the same building as this group, we had the unique opportunity to conduct comparisons in the same environment using the same sensor configuration.
2. **Localization with ceiling lights.** Various research teams have successfully used ceiling lights as landmarks, including HelpMate Robotics, which has built a landmark commercial service robot application that has been deployed in hospitals world wide (King & Weiman, 1990). HelpMate's navigation system is extremely reliable. In our building, ceiling lights are easy to recognize, stationary, and rarely blocked by obstacles, making them prime candidate landmarks for mobile robot localization.

Our previously best localization algorithm (Thrun, in press; Thrun et al., 1996), which is based on *model matching*[4] and which is now distributed commercially by a mobile robot manufacturer (Real World Interface, Inc.), was not included in the comparison, because this approach is incapable of localizing the robot under global uncertainty. In fact, most approaches in the literature are restricted to *position tracking*, i.e., localization under the assumption that the initial position is known. Of the few approaches to global localization, most require that a single sensor snapshot suffices to disambiguate the position—an assumption which rarely holds true in practice.

### *5.1. Testbed and implementation*

This section describes the robot, its environment, the data, and the specific implementation used throughout our experiments.

***5.1.1. Environment.*** Figure 2(b) shows a hand-drawn map of our testing environment, of which we used an 89 meter-long corridor segment. The environment contains two windows

(at both corners), various doors, an elevator, three to four trash bins, and a hallway. The environment was also dynamic. While the data was recorded, the corridors were populated, the status of some of the doors changed, and the natural daylight had a strong effect on camera images taken close to the windows. Strictly speaking, such dynamics violate the Markov assumption (cf. Section 2.3), but as documented here and elsewhere (Burgard et al., 1996a; Burgard et al., 1996b; Kaelbling, Cassandra, & Kurien, 1996; Leonard, Durrant-Whyte, & Cox, 1992; Koenig & Simmons, 1996; Nourbakhsh, Powers, & Birchfield, 1995; Smith, Self, & Cheeseman, 1990), the probabilistic approach is fairly robust to such dynamics.

***5.1.2. Data collection.*** During data collection, the robot moved autonomously at approximately 15 cm/s, controlled by our local obstacle avoidance and navigation routines (Fox, Burgard, & Thrun, 1996). In 12 separate runs, a total of 9,815 sensor snapshots were collected (228 MB raw data). The dats was recorded using three different pointing directions for the robot's camera:

1. **Data set D-1:** In 3,232 snapshots, the camera was pointed towards the outer side of the corridor, so that doors were clearly visible when the robot passed by them.
2. **Data set D-2:** In 3,110 snapshots, the camera was pointed towards the interior of the building. Here the total number of doors is much smaller, and doors are wider.
3. **Data set D-3:** Finally, in 3,473 data points, the camera was pointed towards the ceiling. This data set was used to compare with landmark-based localization using ceiling lights.

The illumination between the different runs varied slightly, as the data was recorded at different times of day. In each individual run, approximately three quarters of the data was used for training, and one quarter for testing (with different partitionings of the data in different runs). When partitioning the data, items collected in the same run were always part of the same partition.

Unless otherwise noted, the robot started at a specific location in each run, from where it moved autonomously through the 89 meter-long segment of corridor. Thus, the principal heading directions in all data are the same; however, to avoid collisions with humans, the obstacle avoidance routines sometimes substantially changed the heading of the robot. Most of our data was collected close to the center of the corridor. Consequently, the networks $\sigma$ and the map $P(f \mid \xi)$ are specialized to our navigation algorithms (Thrun et al., 1996). This is similar to work by others (Kuipers & Byun, 1988; Kuipers & Byun, 1991; Matarić, 1990), whose definition of a landmark also requires that the robot use a particular navigation algorithm that makes it stay at a certain proximity to obstacles. Although the robot travels the corridor in both directions in everyday operation, we felt that for the purpose of scientific evaluation, using data obtained for a single travel direction was sufficient.[5]

Location $\xi$ was modeled by a three-dimensional variable $\langle x, y, \theta \rangle$. Instead of measuring the exact locations of the robot by hand, which would not have been feasible given the large number of positions, we used the robot's odometry and the position tracking algorithm described by Thrun (in press) to derive the position labels. The error of these automatically derived position labels was significantly lower than the tolerance threshold of our existing navigation software (Thrun et al., 1996).

***5.1.3. Preprocessing.*** In all our runs, images were preprocessed to eliminate some of the daytime- and view-dependent variations and to reduce the dimensionality of the data. First, the pixel mean and variance were normalized in each image. Subsequently, each image was subdivided into ten equally sized rows and independently into ten equally sized columns. For each row and column, seven characteristic image features were computed:

- **average brightness**,
- **average color** (one for each of the three color channels), and
- **texture information:** the average absolute difference of the RGB values of any two adjacent pixels (in a subsampled image of size 60 by 64, computed separately for each color channel).

In addition, 24 sonar measurements were collected, resulting in a total of $7 \times 20 + 24 = 164$ sensory values per sensor snapshot. During the course of this research, we tried a variety of different image encodings, none of which appeared to have a significant impact on the quality of the results. The features are somewhat specific to domains that possess brightness, color, or texture cues, which we believe to be applicable to a wide range of environments. The basic learning algorithm, however, does not depend on the specific choice of the features, and it does not require any preprocessing for reasons other than computational efficiency.

***5.1.4. Neural networks.*** In all our experiments, multilayer perceptrons with sigmoidal activation functions (Rumelhart, Hinton, & Williams, 1986) were used to filter the (preprocessed) sensor measurements. These networks contained 164 input units, six hidden units, and one output unit. Runs using different network structures (e.g., two hidden layers) gave similar results as long as the number of hidden units per layer was not smaller than four. To decrease the training time, we used the pseudo-pattern training method described in item 3 of Section 4.4.2, interleaving 100 steps of backpropagation training with one computation of $\tilde{E}_{\text{posterior}}$ and its derivatives. Networks were trained using a learning rate of 0.0001, a momentum of 0.9, and a version of conjugate gradient descent (Hertz, Krogh, & Palmer, 1991). These modifications of the basic algorithm were exclusively adopted to reduce the overall training time, but an initial comparison using the unmodified algorithm (see Table 2) gave statistically indistinguishable results. As noted above, learning required between 30 min and 12 h on a 200 MHz Pentium Pro.

***5.1.5. Error function.*** In our implementation, the error $e(\xi^*, \xi)$ measures the distance the robot must travel to move from $\xi^*$ to $\xi$. Thus, the further the robot must travel when erroneously believing to be at $\xi$, the larger its error.

***5.1.6. Map.*** *Nearest neighbor* (Franke, 1982; Stanfill & Waltz, 1986) was used to compute $P(f_i \mid \xi)$. More specifically, in our experiments the entire training set $X$ was memorized. For each query location $\xi$, a set of $k$ values $g(s_1), g(s_2), \ldots, g(s_k)$ was computed for the $k$ data points in $X$ nearest to $\xi$. Nearness was calculated using Euclidean distance. The desired probability $P(f_i \mid \xi)$ was assumed to be the average $k^{-1} \sum_{i=1}^{k} g(s_i)$. This approach was found to work reasonably well in practice. The issue of how to best approximate $P(f \mid \xi)$ from finite sample sizes is orthogonal to the research described here and was

therefore not investigated in depth; for example, see Burgard et al. (1996a), Gelb (1974), Nourbakhsh, Powers, & Birchfield (1995), Simmons & Koenig (1995), and Smith, Self, & Cheeseman (1990) for further literature on this topic.

***5.1.7. Testing conditions.*** We were particularly interested in measuring performance under different uncertainties—from local to global. Such comparisons are motivated by the observation that the utility of a feature depends crucially on uncertainty (cf. Section 3.2). In most of our runs, the uncertainty $Bel_{\text{prior}}(\xi)$ was uniformly distributed and centered around the true (unknown) position. In particular, the distributions used here had the widths: [−1 m, 1 m], [−2 m, 2 m], [−5 m, 5 m], [−10 m, 10 m], [−50 m, 50 m], and [−89 m, 89 m]. The range of uncertainties captures situations with global uncertainty as well as situations where the robot knows its position within a small margin. An uncertainty of [−89 m, 89 m] corresponded to *global* uncertainty, since the environment was 89 m long. We will refer to uncertainties at the other end of the spectrum ([−1 m, 1 m], [−2 m, 2 m]) as *local*. An uncertainty of [−1 m, 1 m] is generally sufficient for our autonomous navigation routines, so no smaller uncertainty was used. If $Bel_{\text{prior}}$ is used in training, we refer to it as *training uncertainty*. If it is used in testing, we call it *testing uncertainty*. When evaluating BaLL, sometimes different prior uncertainties are used in training and testing to investigate the robustness of the approach.

***5.1.8. Dependent measures.*** The absolute error $\tilde{E}_{\text{posterior}}$ depends on the prior uncertainty $Bel_{\text{prior}}(\xi)$, hence it is difficult to compare for different prior uncertainties. We therefore chose to measure instead the *error reduction*, defined as

$$1 - \frac{\tilde{E}_{\text{posterior}}}{\tilde{E}_{\text{prior}}}. \tag{46}$$

Like the posterior error $\tilde{E}_{\text{posterior}}$, the prior error $\tilde{E}_{\text{prior}}$ denotes the approximation of $E_{\text{prior}}$ based on the data $X$. For example, if the prior error $\tilde{E}_{\text{prior}}$ is four times as large as the posterior error $\tilde{E}_{\text{posterior}}$ after taking a sensor snapshot, the error reduction is 75%. The larger the error reduction, the more useful the information extracted from the sensor measurement for localization. In our experiments, the initial error of a globally uncertain robot is 42.2 m; thus, to lower the error to 1 m, it must reduce its error by 97.6%. The advantage of plotting the error reduction instead of the absolute error is that all results are in the same scale regardless of the prior uncertainty, which facilitates their comparison.

### *5.2. Results*

The central hypothesis underlying our research is that filters learned by BaLL can outperform human-selected landmarks. Thus, the primary purpose of our experiments was to compare the performance of BaLL to that the other two approaches. Performance was measured in terms of localization error. The secondary purpose of our experiments was to understand what features BaLL uses for localization. Would the features used by BaLL be similar to those chosen by humans, or would they be radically different?

In a first set of experiments, we evaluated the error reduction for BaLL and compared it to the two other approaches, under different experimental conditions. The error reduction directly measures the usefulness of a filter $\sigma$ or a particular type of landmark for localization. Thus, it lets us judge empirically how efficient each approach is in estimating a robot's location. Different experiments were conducted for different uncertainties (from local to global), and different numbers of networks $n$.

***5.2.1. One neural network and same uncertainty in training and testing.*** In the first experiment, BaLL was used to train a single neural network, which was then compared to the other two approaches. While the different approaches were evaluated under the different uncertainties, in each experiment the testing uncertainty was the same as the training uncertainty. Consequently, the results obtained here represent the *best case* for BaLL, since $\sigma$ was trained for the specific uncertainty that was also used in testing.

Figure 3 shows the error reduction obtained for the three different data sets D-1, D-2, and D-3. In each of these diagrams, the solid line indicates the error reduction obtained for BaLL, whereas the dashed line depicts the error reduction for the other two approaches (landmark-based localization using doors in figures 3(a) and (b), and ceiling lights in figure 3(c)). In all graphs, 95% confidence intervals are also shown.

As can be seen in all three diagrams, BaLL significantly outperforms the other approaches. For example, as the results obtained with data set D-1 indicate (figure 3(a)), doors appear to be best suited for $\pm 2$ m uncertainty. If the robot knows its location within $\pm 2$ m, doors, when used as landmarks, reduce the uncertainty by an average of 8.31%. BaLL identifies a filter that reduces the error by an average of 14.9%. This comparison demonstrates that BaLL extracts more useful features from the sensor data. The advantage of our approach is even larger for increasing uncertainties. For example, if the robot's uncertainty is $\pm 50$ m, BaLL reduces the error by 36.9% (data set D-1), whereas doors reduce the error by as little as 5.02%.

Similar results occur for the other two data sets. For example, in data set D-2, where the camera is pointed towards the inside wall, the door-based approach reduces the error by a maximum average of 9.78% (figure 3(b)). In this environment, doors appear to be best suited for $\pm 5$ m uncertainty (and not for $\pm 2$ m), basically because doors on the interior side of the testing corridor are wider and there are fewer of them. Here, too, BaLL outperforms the door-based approach. It successfully identifies a feature that reduces the uncertainty by 15.6% under otherwise equal conditions. In figure 3(b), the largest relative advantage of our approach over the door-based approach occurs when the robot is globally uncertain about its position. Here BaLL reduces the error by 27.5%, whereas the door-based approach yields no noticeable reduction (0.00%).

The results obtained for data set D-3, where the camera is pointed upward, are generally similar. If the prior uncertainty is $\pm 2$ m, ceiling lights manage to reduce the error by as much as 15.6%, indicating that they are significantly better suited for this type of uncertainty than doors. We attribute this finding to the fact that most of our ceiling lights are spaced in regular intervals of about 5 m. BaLL outperforms localization based on ceiling lights in all cases, as can be seen in figure 3(c). For example, it reduces the error by 21.0% for $\pm 2$ m prior uncertainty and by 39.5% for $\pm 50$ m uncertainty. All these results are significant at the 95% confidence level.

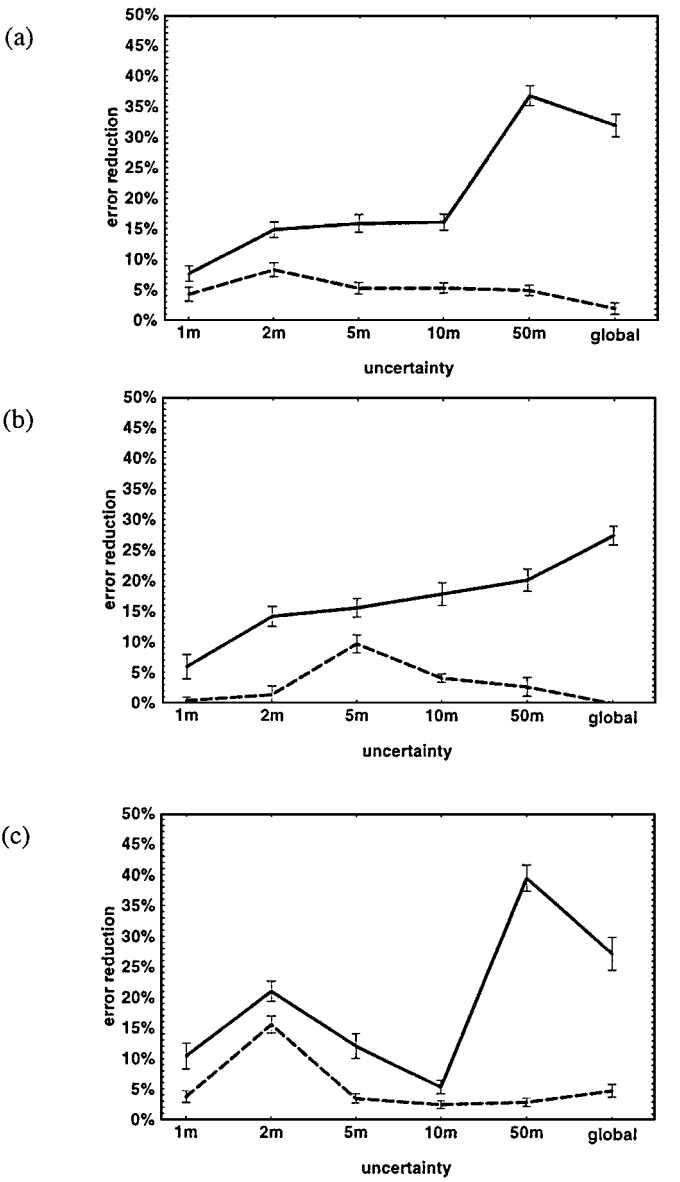


*Figure 3*. Average results. The dashed line indicates the error reduction obtained for doors, and the solid line indicates the error reduction if the filters are learned using BaLL, for (a) D-1, (b) D-2, and (c) D-3.

*Table 3.* Comparison of BaLL with the other two approaches ($\pm 2$ m uncertainty). Here $n$ specifies the dimension of the features vector $f$ (i.e., the number of networks), and the numbers in the table give the average error reduction and their 95% confidence interval. The results in the first two rows can also be found in figure 3 at $\pm 2$ m.

| Data set | D-1 | D-2 | D-3 |
|---|---|---|---|
| Doors | $8.3\% \pm 1.1\%$ | $1.4\% \pm 1.5\%$ | — |
| Ceiling lights | — | — | $15.6\% \pm 1.4\%$ |
| BaLL, $n = 1$ | $14.9\% \pm 1.3\%$ | $14.2\% \pm 1.6\%$ | $21.0\% \pm 1.7\%$ |
| BaLL, $n = 2$ | $33.5\% \pm 1.9\%$ | $36.7\% \pm 1.9\%$ | $34.5\% \pm 2.0\%$ |
| BaLL, $n = 3$ | $39.7\% \pm 2.0\%$ | $36.3\% \pm 2.3\%$ | $41.5\% \pm 1.8\%$ |
| BaLL, $n = 4$ | $41.7\% \pm 2.5\%$ | $36.7\% \pm 1.9\%$ | $42.9\% \pm 1.6\%$ |

***5.2.2. Multiple neural networks and same uncertainty in training and testing.*** In a second experiment, we held uncertainty constant but varied the number of networks and hence the dimensionality of the feature vector (from one to four). As described in Section 4, BaLL can simultaneously train multiple networks.

The primary result of this study was that, as the number of networks increases, BaLL's advantage over the alternative approaches increases. Table 3 shows the average error reduction (and 95% confidence interval) obtained for $\pm 2$ m uncertainty and for the three different data sets. For $n = 4$, the difference between BaLL and both other approaches is huge. Our approach finds features that reduce the error on average by 41.7% (data set D-1), 36.7% (D-2), and 42.9% (D-3), whereas the other approaches reduce the error only by 8.3%, 1.4%, and 15.6%, respectively. According to these numbers, BaLL is between 2.75 and 26.2 times as data efficient as the alternatives.

To understand the performance improvement over the single network case, it is important to notice that multiple networks tend to extract different features. If all networks recognized the same features, their output would be redundant and the result would be the same as if $n = 1$. If their outputs differ, however, the networks will generally extract *more* information from the sensors, which will usually lead to improved results, as demonstrated by the performance results. Thus, the observation that the different networks tend to select different features is a result of minimizing the posterior error $E_{\text{posterior}}$. As we have shown in an earlier version of this article (Thrun, 1996), the output of the networks is largely uncorrelated, demonstrating that different, nonredundant features are extracted by the different networks.

***5.2.3. Multiple neural networks and different uncertainty in training and testing.*** In a third experiment, we investigated BaLL's performance when trained for one particular uncertainty but tested under another. These experiments have practical importance, since in our current implementation the slowness of the learning procedure prohibits training new networks every time the uncertainty changes. We conducted a series of runs in which networks were trained for $\pm 2$ m uncertainty and tested under the various different uncertainties. In the extreme case, the testing uncertainty was $\pm 89$ m, whereas the training uncertainty was $\pm 2$ m.

Figure 4 shows the results obtained for $n = 4$ networks using the three different data sets. As expected, the networks perform best when the training uncertainty equals the

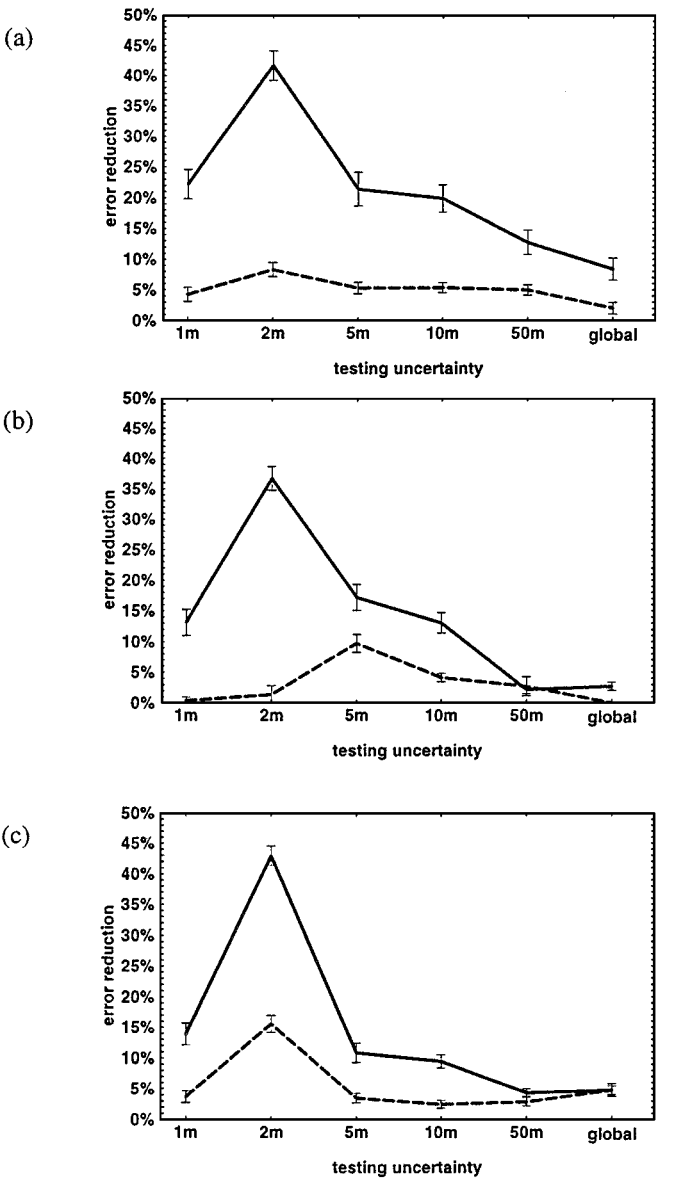


*Figure 4.* Even under different uncertainties, the learned filters reduce the error significantly more than those that recognize doors. This figure shows results obtained when $n = 4$ networks are trained for $\pm 2$ m uncertainty. BaLL's error reduction is plotted by the solid line, whereas the dashed line depicts the error reduction when doors are used as landmarks. The figure shows results for data set (a) D-1, (b) D-2, and (c) D-3.

testing uncertainty. The primary results are that the performance degrades gracefully, in that even if the training uncertainty differs drastically from the testing uncertainty, the networks extract useful information for localization. BaLL still outperforms both alternative approaches or produces statistically indistinguishable results. These results suggest that, in our environment, a single set of filters might still be sufficient for localization, although the results might not be as good as they would be for multiple sets of filters.

***5.2.4. Global localization.*** In a final experiment, BaLL was applied to the problem of global localization, in which the robot does not know its initial location. We model this by assuming its initial uncertainty is uniformly distributed. As the robot moves, the internal belief is refined based on sensor readings. In our environment, a single sensor snapshot is usually insufficient to determine the position of the robot uniquely. Thus, multiple sensor readings must be integrated over time.

We conducted 35 global localization runs comparing the door-based localization approach with BaLL. In each run, the robot started at a random position in the corridor. Sensor snapshots were taken every 0.5 m and incorporated into the internal belief using the probabilistic algorithm described in Table 1. Both approaches—the door-based approach and BaLL, used the same data; thus, any performance, difference was exclusively due to the different information extracted from the sensor readings. BaLL used $n = 4$ networks, which were trained for $\pm 2$ m uncertainty prior to robot operation and held constant while the robot localized itself.

Figure 5 depicts an example run with BaLL. This figure shows the belief $Bel(\xi)$ and the error at different stages of the localization. Each row in figure 5 gives the belief $Bel(\xi)$ at a different time, with time progressing from the top to the bottom. In each row, the true position of the robot is marked by the square. As can be seen in figure 5, the features

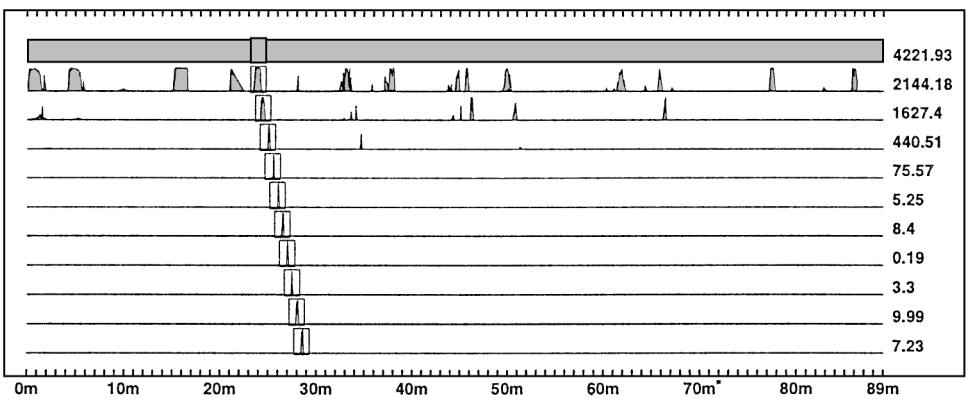


*Figure 5.* An example of global localization with BaLL. Each curve represents the belief $Bel(\xi)$ of the robot at a different time. Initially (top curve), the robot's position is in the center of the rectangle. Its initial belief $Bel(\xi)$ is uniformly distributed. As the robot senses and moves forward, $Bel(\xi)$ is refined. After four sensor measurements, $Bel(\xi)$ is centered on the correct position. The numbers on the right side depict the error $E_{\text{posterior}}$ at the different stages of the experiment.

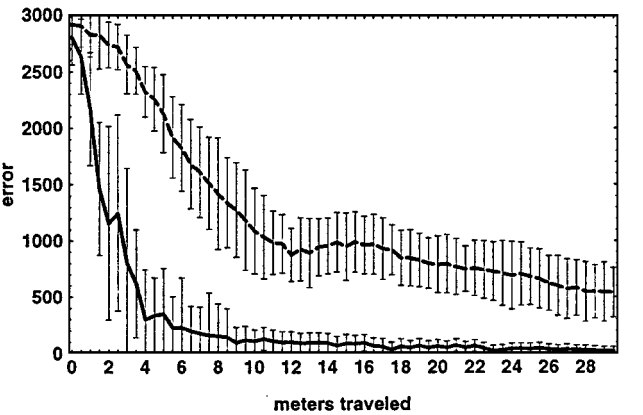


*Figure 6.* Absolute error in cm as a function of meters traveled, averaged over 35 different runs with different starting points. Every 0.5 m, the robot takes a sensor snapshot. The dashed line indicates the error when doors are used as landmarks and the solid line corresponds to BaLL, which gives superior results.

extracted by BaLL reduce the overall uncertainty fairly effectively, and after four sensor readings (2 meters of robot motion) the robot "knows where it is."

Figure 6 summarizes the result of the comparison between BaLL and the door-based approach (dashed line). This shows the average error (in cm) as a function of the distance traveled, averaged over 35 different runs with randomly chosen starting positions, along with 95% confidence intervals. The results demonstrate the relative advantage of learning the features. After 30 m of robot travel, the average error of the door-based approach is 7.46 m. In contrast, BaLL attains the same accuracy after only 4 m, making it approximately 7.5 times as data efficient. After 9.5 m, BaLL yields an average error that is smaller than 1 m. The differences are all statistically significant at the 95% level.

As noted above, when applied in larger environments or in the bidirectional case, the number of sensor readings required for global localization increases. To quantify this increase, it is useful to consider the problem of localization from an information-theoretic viewpoint. Reducing the uncertainty from 89 m to 1 m $\log_2(89\text{ m}/1\text{ m}) = 6.48$ independent bits of information. This is because it takes 6.48 bits to code a symbol using an alphabet of 89 symbols. Of course, consecutive sensor readings are not independent, reducing the amount of information they convey. Empirically, BaLL requires on average 19 sensor readings (9.5 m) to reduce the error to 1 m. These numbers suggest that, on average, it needs $3.5 = 19/5.4$ sensor readings for one bit of independent position information. Consequently, in a corridor twice the size of the one considered here (or in the bidirectional case), 23 sensor readings should be sufficient to obtain an average localization error of 1 m (since $23 \geq 19 + 3.5$).

### *5.3. What features do the neural networks extract?*

Analyzing the trained neural networks led to some interesting findings. In general, the networks use a mixture of different features for localization, such as doors, dark spots, wall

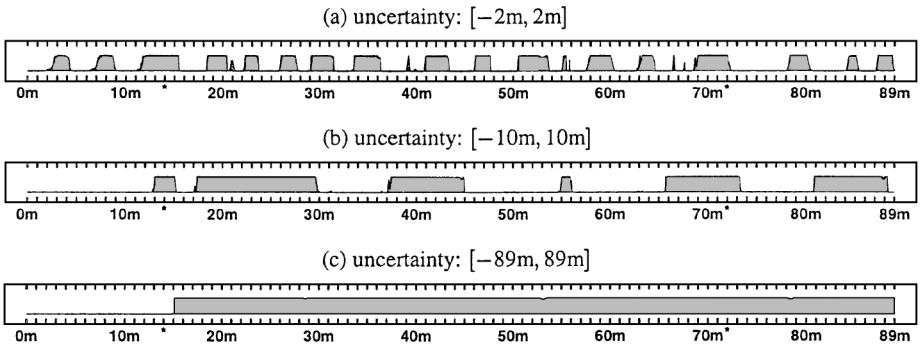


*Figure 7.* Example output characteristics of a filter, plotted over data obtained in one of the runs. The filters were trained for (a) ±2 m, (b) ±10 m, and (c) ±89 m (global) uncertainty.

color, hallways, and blackboards. In several runs, the networks became sensitive to a spot in our corridor whose physical appearance did not, to us, differ from other places. Closer investigation of this spot revealed that, due to an irregular pattern of the ceiling lights, the wall is slightly darker than the rest of the corridor. This illumination difference is barely visible to human eyes, since they compensate for the total level of illumination. However, our camera is very sensitive to the total level of illumination, which explains why the robot repeatedly selected this spot for use in localization.

We also investigated the effect of different training uncertainties on the filter $\sigma$. As the above results suggest, different filters are learned for different uncertainties, so the question arises as to what type of features are used under these different conditions.

Using data set D-1 as an example, figure 7 depicts example outputs of trained networks for ±2 m, ±10 m, and ±89 m training uncertainty. Each curve plots the output value for the corresponding network, which were evaluated in the 89 meter-long corridor. Obviously, the features extracted by the different networks differ substantially. Whereas the network trained for small uncertainty is sensitive to local features such as doors, hallways, and dark spots, the network trained for large uncertainty is exclusively sensitive to the color of the walls. Roughly a quarter of our corridor is orange and three quarters are light brown. If the robot is globally ignorant about its position, wall color is vastly superior to any other feature, as illustrated by the performance results described in the previous section. If the robot is only slightly uncertain, however, wall color is a poor feature.

Figure 8 depicts the output of $n = 4$ networks, simultaneously trained for local uncertainty (±2 m). As discussed above, the different networks specialize to different perceptual

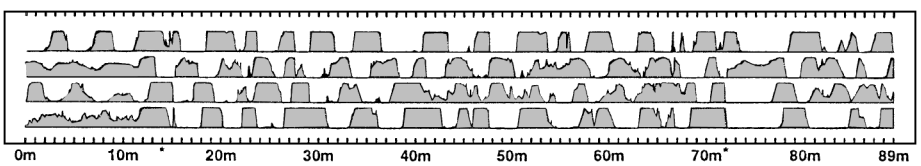


*Figure 8.* Example output characteristics of $n = 4$ filters, optimized for ±2 m uncertainty.

features. More examples and a more detailed discussion of the different features learned under the various experimental conditions can be found in thrun (1996).

## 6. Related work

Mobile robot localization has frequently been recognized as a key problem in robotics with significant practical importance. Cox (1991) considers localization to be a fundamental problem to providing a mobile robot with autonomous capabilities. A recent book by Borenstein, Everett, & Feng (1996) provides an excellent overview of the state-of-the-art in localization. Localization—and in particular localization based on landmarks—plays a key role in various successful mobile robot architectures.[6] While some localization approaches, such as Horswill (1994), Koenig & Simmons (1996), Kortenkamp & Weymouth (1994), Matarić (1990), and Simmons & Koenig (1995), localize the robot relative to some landmarks in a topological map, BaLL localizes the robot in a metric space, as in the approach by Burgard et al. (1996a, 1996b).

However, few localization approaches can localize a robot globally; they are used mainly to track the position of a robot. Recently, several authors have proposed probabilistic representations for localization. Kalman filters, which are used by Gelb (1974), Rencken (1995), Smith & Cheeseman (1985), and Smith, Self, & Cheeseman (1990), represent the location of a robot by a Gaussian distribution; however, they only can represent unimodal distributions, and so they are usually unable to localize a robot globally. It is feasible to represent densities using mixtures of Kalman filters, as in the approach to map building reported by Cox (1994), which would remedy the limitation of conventional Kalman filters. The probabilistic approaches described by Burgard et al. (1996a, 1996b), Kaelbling, Cassandra, & Kurien (1996), Koenig & Simmons (1996), Nourbakhsh, Powers, & Birchfield (1995), and Simmons & Koenig (1995) employ mixture models or discrete approximations of densities that can represent multimodal distributions. Some of these approaches are capable of localizing a robot globally. The probabilistic localization algorithm described in Section 2 borrows from this literature but generalizes all these approaches. It smoothly blends both position tracking and global optimization, using only a single update equation.

Most existing approaches to mobile robot localization extract static features form the sensor readings, usually using hand-crafted filter routines. The most popular class of approaches to mobile robot localization, *localization based on landmarks*, scan sensor readings for the presence or absence of landmarks. A diverse variety of objects and spatial configurations have been used as landmarks. For example, many of the landmark-based approaches reviewed in Borenstein, Everett, & Feng (1996) require artificial landmarks such as bar-code reflectors (Everett et al., 1994), reflecting tape, ultrasonic beacons, or visual patterns that are easy to recognize, such as black rectangles with white dots (Borenstein, 1987). some recent approaches use more natural landmarks that do not require modifications of the environment. For example, the approaches of Kortenkamp & Weymouth (1994) and Matarić (1990) use certain gateways, doors, walls, and other vertical objects to determine the robot's position, and the Helpmate robot uses ceiling lights to position itself (King & Weiman, 1990). The approaches reported in Collet & Cartwright (1985) and Wolfart,

Fisher, & Walker (1995) use dark/bright regions and vertical edges as landmarks. These are just a few representative examples of many different landmarks used for localization.

*Map matching*, which comprises a second, not quite as popular family of approaches to localization, converts sensor data to metric maps of the surrounding environment, such as occupancy grids (Moravec, 1988) or maps of geometric features such as lines. The sensor map is then matched to a global map, which might have been learned or provided by hand (Cox, 1991; Rencken, 1993, Schiele & Crowley, 1994; Thrun, 1993; Thrun, in press; Yamauchi & Beer, 1996; Yamauchi & Langley, 1997). Map-matching approaches filter sensor data to obtain a map, and then use only the map in localization.

These approaches have in common the fact that the features extracted from the sensor readings are predetermined. By hand selecting a particular set of landmarks, the robot ignores all other information in the sensor readings which, as our experiments demonstrate, might carry some additional information. By mapping sensor readings to metric maps in a fixed, precoded way, the robot ignores other potentially relevant information in the sensor readings. The current work lets the robot determine by itself what features to extract for localization, based on their utility for localization. As shown by our empirical comparison, enabling a robot to extract its own features (and learning its own landmarks) has a noticeable impact on the quality of the results.

Probably the most related research is that by Greiner and Isukapalli (1994). Their approach can select a set of landmarks from a larger, predefined set of landmarks, using an error measure that bears close resemblance to the one used in BaLL. The selection is driven by the localization error after sensing, which is determined empirically from training data. In that regard, both approaches exploit the same basic objective function in learning filters. Greiner and Isukapalli's approach differs from BaLL primarily in three respects. First, it assumes that the exact location of each landmark is known before learning. The robot does not define its own landmarks; rather, it starts with a set of human-defined landmarks and rules out those that are not suited for localization. Second, it is tailored towards correcting small localization errors and cannot perform global localization. Third, the class of landmarks that BaLL can learn is much broader. The filters used by Greiner and Isukapalli are functions of three parameters (distance, angle, and type of landmark); in contrast, BaLL employs neural networks that have many more degrees of freedom.

## 7. Conclusion

This paper has presented a Bayesian approach to mobile robot localization. The approach relies on a probabilistic representation and uses Bayes' rule to incorporate sensor data into internal beliefs and to model robot motion. The key novelty is a method, called BaLL, for training neural networks to extract a low-dimensional feature representation from high-dimensional sensor data. A rigorous Bayesian analysis of probabilistic localization provided a rational objective for training the neural networks so as to directly minimize the quantity of interest in mobile robot localization: the localization error. As a result, the features extracted from the sensor data emerge as a side effect of the optimization.

An empirical comparison with two other localization algorithms under various conditions demonstrated the advantages of BaLL. We compared BaLL with an existing method by

Koenig & Simmons (1996), which use doors as landmark in localization, and with King & Weiman's (1990) method, which uses ceiling lights. In this experiment, our approach identified features that led to superior localization results. In some cases, the performance differences were large, particularly for localization under global uncertainty.

BaLL's ability to learn its own features and thus to discover its own landmarks has important consequences for mobile robot navigation. In particular, the probabilistic paradigm has four principal advantages when compared to conventional approaches to this problem:

1. **Autonomy.** BaLL obviates the need for manually determining the features to extract from the sensor data. In most previous approaches to localization, a human designer had to manually specify the features to extract. For example, most landmark-based approaches rely on predetermined landmarks, chosen by human expert, which requires expertise about the robot, its sensors, and the environment. BaLL replaces the need for selecting the right features by automated learning.
2. **Optimality.** In BaLL, filters are learned by attempting to optimize the accuracy of the localization routines that employs them. Asymptotically, minimizing the Bayesian error should yield optimal filters. Of course, BaLL might fail to find an optimal set of filters for three reasons: BaLL is trained using finite sample sets; backpropagation networks might not be sufficient to represent optimal filters; and the gradient descent training procedure might converge to a local minimum.
3. **Environmental flexibility.** Our method can also customize itself to different environments. Any routine that relies on static, built-in landmarks should fail in environments that do not possess such landmarks. For example, although ceiling lights might be appropriate landmarks in some environments, they are inappropriate in environments that do not possess them. By providing a method that supports the automatic customization of robots to their environments, we hope to achieve a level of flexibility that facilitates the design of future service robot applications (e.g., service robots operated in private homes, whose design varies greatly from home to home).
4. **Sensor flexibility.** The current approach does not hinge on a specific sensor technology. In contrast, most existing localization approaches are closely tied to a particular type of sensor. For example, routines that rely on visual cues require that the robot be equipped with a camera. BaLL is more flexible in that it automatically adapts to the particular type sensor. We conjecture that it will also scale better to high-dimensional sensor spaces that arise if a large number of sensors are used simultaneously. Exploiting the information in high-dimensional sensor spaces has proven to be extremely difficult for human engineers; for this reason we believe that data-driven learning approaches such as the one proposed in this paper will ultimately make a lasting contribution to the field of robotics.

A key limitation of the current approach is that it does not learn the location of landmarks (in $x$-$y$ coordinates). Instead, it learns to associate sensor readings with robot locations. If the robot knew the location of the landmarks, it could apply projective geometry to predict a landmark's appearance from different, nearby locations. In the current approach, this is not possible and we suspect that this limitation causes an increased need for training data. A second limitation arises from the fact that, after the initial training phase, learning is

discontinued. If the environment changes, it is desirable that the localization routines adapt to the changes, and it appears feasible to extend BaLL accordingly. As the robot knows roughly where it is, BaLL can use its position estimates to label sensor data automatically, and use this self-labeled data for further training. The effectiveness of such an extension in practice remains to be seen.

Although we have applied BaLL only to one specific problem in mobile robot localization, the mathematical framework presented here is more general and can be applied to a whole range of decision problems arising in the context of mobile robotics. These include:

1. **Active sensing.** As shown in Thrun (1996), the mathematical framework yields a rational incentive for pointing sensors so as to best localize a robot. Empirical results have demonstrated that by actively controlling the pointing direction of the robot's camera so as to minimize the future expected localization error, the efficiency and robustness of localization can be improved further.
2. **Sensor selection.** the approach can also be used to determine what sensor to include on mobile robots. Previously, robot designers lacked a formal method for determining the sensors best suited for localization. By comparing the Bayesian localization error for different type sensors, our analysis provides a rational criterion for determining which sensors work best and where to mount them.
3. **Navigation for localization.** The approach can be used to determine where to move so as to best localize the robot, a problem that has previously been studied by Kaelbling, Cassandra, & Kurien (1996). The mathematical details are discussed in Thrun (1996), while a similar approach with empirical results can be found in Burgard, Fox, & Thrun (1997).

The Bayesian method presented here is an instance of a general approach for the estimation of hidden state and the integration of high-dimensional sensor data over time. Put in this light, a limiting assumption of the current approach is the requirement that, during training, the hidden state must be accessible. This is obviously a reasonable assumption to make in some situations (such as the one studied here), but it is unreasonable in many other situations. Thus, an open research issue is the extension of the current methods for the estimation of hidden state when it is not accessible. Preliminary results carried out in our lab in the context of intelligent building control have led to an extension to situations where only a low-dimensional projection of the hidden state is accessible during training. We suspect that the general paradigm of Bayesian analysis has the potential for new class of more capable learning algorithms, with the current work being just an initial example.

## Acknowledgments

The author wishes to thank the RHINO and the XAVIER mobile robot groups, particularly Wolfram Burgard, Dieter Fox, Tom Mitchell, and Reid Simmons, for stimulating discussions and feedback during the course of this research. He also thanks the editor Pat Langley and three anonymous reviewers for suggestions that improved the quality of this manuscript, and in particular Pat Langley for five iterations of improvements.

This research is sponsored in part by Daimler-Benz Research (via Frieder Lohnert) and the Defense Advanced Research Projects Agency (DARPA) via the Airforce Missile System Command under contract number F04701-97-C-0022. 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 Daimler-Benz Research, the Airforce Missile System Command, or the United States Government.

## Notes

1. As noted by Matarić (1990), "Although intuitively clear, the concept [of a landmark] is difficult to define." We adopt Matarić's and Presson & Montello's (1988) definition, who define a landmark as "any element (object or feature) which can serve as a point reference." The reader may notice that some authors, such as Chown, Kaplan, & Kortenkamp (1995) or Kuipers & Levitt (1988), propose more specific definitions, such as that landmarks must be unique, they must correspond to physical objects, or they must be visible from everywhere.
2. Other error measures, such as measures related to the efficiency of robot motion, are not considered, since our work focuses on the problem of state estimation, and our current approach does not address robot control.
3. In our experiments, we did not observe an overfitting effect, as experiments reported in Thrun (1996) demonstrate. Thus, we simply trained the networks for a large number of iterations (such as 10000).
4. See also Chatila & Laumond (1985), Rencken (1993), Schiele & Crowley (1994), and Yamauchi & Beer (1996).
5. As far as tracking the position of the robot is concerned, the results obtained here should directly transfer to the bidirectional case, since the robot never turns an unnoticed 180°. Global localization in the bidirectional case is generally more difficult for any approach, due to the increased number of possible locations. The main point of this paper is to provide ways for learning features (landmarks) for localization, and we have no reason to believe that the qualitative results obtained here do not transfer to the bidirectional case (as they should not change when one goes from a 89 m to a 178 m corridor). Section 5.2.4 quantifies the effect of larger environments, or the general bidirectional case.
6. Examples can be found in Betke & Gurvits (1993), Cox (1991), Cox (1994), Horswill (1994), Fukuda et al. (1993), Hinkel & Knieriemen (1988), Koenig & Simmons (1996), Kortenkamp & Weymouth (1994), Leonard & Durrant-Whyte (1992), Leonard, Durrant-Whyte, & Cox (1992), Matarić (1990), Neven & Schöner (1995), Nourbakhsh, Powers, & Birchfield (1995), Peters et al. (1994), Rencken (1993), Schiele & Crowley (1994), Simmons & Koenig (1995), Thrun et al. (1996), Weiß, Wetzler, & von Puttkamer (1994), and various chapters in Kortenkamp, Bonassi, & Murphy (in press).

## References

Betke, M., & Gurvits, L. (1993). *Mobile robot localization using landmarks* (Technical Report SCR-94-TR-474). Princeton: Siemens Corporate Research.

Borenstein, J. (1987). *The nursing robot system*. Doctoral dissertation, Technion, Haifa, Israel.

Borenstein, J., Everett, B., & Feng, L. (1996). *Navigating mobile robots: systems and techniques*. Wellesley, MA: A. K. Peters, Ltd.

Buhmann, J. (1995). Data clustering and learning. In M. Arbib (Ed.), *Handbook of brain theory and neural networks* (pp. 278–282). Cambridge, MA: Bradfort Books/MIT Press.

Buhmann, J., Burgard, W., Cremers, A.B., Fox, D., Hofmann, T., Schneider, F., Strikos, J., & Thrun, S. (1995). The mobile robot Rhino. *AI Magazine, 16*, 31–38.

Burgard, W., Fox, D., Hennig, D., & Schmidt, T. (1996a). Estimating the absolute position of a mobile robot using position probability grids. *Proceedings of the Thirteenth National Conference on Artificial Intelligence* (pp. 896–901). Menlo Park, CA: AAAI Press/MIT Press.

Burgard, W., Fox, D., Hennig, D., & Schmidt, T. (1996b). Position tracking with position probability grids. *Proceedings of the First Euromicro Workshop on Advanced Mobile Robots* (pp. 2–9). IEEE Computer Society Press.

Burgard, W., Fox, D., & Thrun, S. (1997). Active mobile robot localization. *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence* (pp. 1346–1352). San Francisco, CA: Morgan Kaufmann.

Casella, G., & Berger, R. (1990). *Statistical Inference*. Pacific Grove, CA: Wadsworth & Brooks.

Chatila, R., & Laumond, J.-P. (1985). Position referencing and consistent world modeling for mobile robots. *Proceedings of the 1985 IEEE International Conference on Robotics and Automation* (pp. 138–145).

Chown, E., Kaplan, S., & Kortenkamp, D. (1995). Prototypes, location, and associative networks (PLAN): Towards a unified theory of cognitive mapping. *Cognitive Science*, *19*, 1–51.

Chung, K. (1960). *Markov chains with stationary transition probabilities*. Berlin, Germany: Springer Publisher.

Collet, T., & Cartwright, B. (1985). Landmark learning in bees. *Journal of Comparative Physiology*.

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

Cox, I, (1994). Modeling a dynamic environment using a Bayesian multiple hypothesis approach. *Artificial Intelligence*, *66*, 311–344.

Engelson, S. (1994). *Passive map larning and visual place recognition*. Doctoral dissertation, Department of Computer Science, Yale University.

Everett, H., Gage, D., Gilbreth, G., Laird, R., & Smurlo, R. (1994). Real-world issues in warehouse navigation. *Proceedings of the SPIE Conference on Mobile Robots IX*. Volume 2352. Boston, MA.

Fahlman, S.E., & Lebiere, C. (1989). *Cascade-correlation learning architecture* (Technical Report CMU-CS-90-100). Pittsburgh, PA: Carnegie Mellon University, Computer Science Department.

Fox, D., Burgard, W., & Thrun, S. (1997). The dynamic window approach to collision avoidance. *IEEE Robotics and Automation Magazine*, *4*(1), 23–33.

Franke, R. (1982). Scattered data interpolation: Tests of some methods. *Mathematics of Computation*, *38*(157), 181–200.

Fukuda, T., Ito, S., Oota, N., Arai, F., Abe, Y., Tanake, K., & Tanaka, Y. (1993). Navigation system based on ceiling landmark recognition for autonomous mobile robot. *Proceedings of the International Conference on Industrial Electronics Control and Instrumentation*, (Vol. 1, pp. 1466–1471).

Gelb, A. (1974). *Applied optimal estimation*. Cambridge, MA: MIT Press.

Greiner, R., & Isukapalli, R. (1994). Learning to select useful landmarks. *Proceedings of National Conference on Artificial Intelligence* (pp. 1251–1256). Menlo Park, CA: AAAI Press/The MIT Press.

Hertz, J., Krogh, A., & Palmer, R.G. (1991). *Introduction to the theory of neural computation*. Redwood City, CA: Addison-Wesley.

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

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feed-forward networks are universal approximators. *Neural Networks*, *2*, 359–366.

Horswill, I. (1994). *Specialization of perceptual processes* (Technical Report AI TR-1511). Cambridge, MA: MIT, Artificial Intelligence Laboratory.

Howard, R.A. (1960). *Dynamic Programming and Markov Processes.* Cambridge, MA: MIT Press and Wiley.

Kaelbling, L., Cassandra, A., & Kurien, J. (1996). Acting under uncertainty: Discrete bayesian models for mobile-robot navigation. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 963–972).

Kalman, R.E. (1960). A new approach to linear filtering and prediction problems. *Transactions ASME, Journal of Basic Engineering*, *82*, 35–45.

King, S., & Weiman, C. (1990). Helpmate autonomous mobile robot navigation system. *Proceedings of the SPIE Conference on Mobile Robots* (Vol. 2352, pp. 190–198). Boston, MA.

Koenig, S., & Simmons, R. (1996). Passive distance learning for robot navigation. *Proceedings of the Thirteenth International Conference on Machine Learning* (pp. 266–274). San Mateo, CA: Morgan Kaufmann.

Kortenkamp, D., Bonassi, D.P., & Murphy, R. (Eds.) (in press). *AI-based mobile robots: Case studies of successful robot systems.* Cambridge, MA: MIT Press.

Kortenkamp, D., & Weymouth, T. (1994). Topological mapping for mobile robots using a combination of sonar and vision sensing. *Proceedings of the Twelfth National Conference on Artificial Intelligence* (pp. 979–984). Menlo Park, CA: AAAI Press/MIT Press.

Kuipers, B., & Byun, Y.-T. (1988). A robust qualitative method for spatial learning in unknown environments. *Proceeding of Eighth National Conference on Artificial Intelligence* (pp. 774–779). Menlo Park, CA: AAAI Press/The MIT Press.

Kuipers, B., & Byun, Y.-T. (1991). A robot exploration and mapping strategy based on a semantic hierachy of spatial representations. *Journal of Robotics and Autonomous Systems, 8*, 47–63.

Kuipers, B., & Levitt, T. (1988). Navigation and mapping in large-scale space. *AI Magazine, 9*, 25–43.

Leonard, J.J., & Durrant-Whyte, H.F. (1992). *Directed Sonar Sensing for Mobile Robot Navigation.* Boston, MA: Kluwer Academic Publishers.

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

Matarić, M.J. (1990). A distributed model for mobile robot environment-learning and navigation. Master's thesis, MIT, Artificial Intelligence Laboratory, Cambridge, MA.

Mine, H., & Osaki, S. (1970). *Markovian Decision Processes.* New York, NY: American Elsevier.

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

Moravec, H., & Martin, M. (1994). Robot navigation by 3D spatial evidence grids (Internal Report). Pittsburgh, PA: Carnegie Mellon University, The Robotics Institute.

Neven, H., & Schöner, G. (1996). Dynamics parametrically controlled by image correlations organize robot navigation. *Biological Cybernetics, 75*, 293–307.

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

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: Networks of plausible inference.* San Mateo, CA: Morgan Kaufmann Publishers.

Peters, L., Surmann, H., Guo, S., Beck, K., & Huser, J. (1994). Moria fuzzy logik gesteuertes, autonomes fahrzeug (Internal Report). Sankt Augustin, Germany: German National Research Center for Information Technology (GMD).

Presson, C., & Montello, D. (1988). Points of reference in spatial cognition: Stalking the elusive landmark. *British Journal of Developmental Psychology, 6*, 378–381.

Rabiner, L.R. (1989). A tutorial on hidden Markov models and selected application in speech recognition. *Proceedings of the IEEE.* IEEE Log Number 8825949.

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

Rencken, W. (1995). Autonomous sonar navigation in indoor, unknown and unstructired environments. In V. Graefe (Ed.), *Intelligent robots and systems.* Amsterdam: Elsevier.

Rumelhart, D.E., Hinton, G.E., & Williams, R.J. (1986). Learning internal representations by error propagation. In D.E. rumelhart & J.L. McClelland (Eds.), *Parallel distributed processing.* Cambridge, MA: MIT Press.

Schiele, B., & Crowley, J. (1994). A comparison of position estimation techniques using occupancy grids. *Proceedings of the 1994 IEEE International Conference on Robotics and Automation* (pp. 1628–1634). San Diego, CA.

Simmons, R., & Koenig, S. (1995). Probabilistic robot navigation in partially observable environments. *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence* (pp. 1080–1087). San Mateo, CA: Morgan Kaufmann.

Smith, R.C., & Cheeseman, P. (1985). *On the representation and estimation of spatial uncertainty* (Technical Report TR 4760 & 7239). Menlo Park, CA: SRI International.

Smith, R., Self, M., & Cheeseman, P. (1990). Estimating uncertain spatial relationships in robotics. In I. Cox & G. Wilfong (Eds.), *Autonomous Robot Vehnicles.* Berlin, Germany: Springer-Verlag.

Stanfill, C., & Waltz, D. (1986). Towards memory-based reasoning. *Communications of the ACM, 29*, 1213–1228.

Thrun, S. (1993). Exploration and model building in mobile robot domains. *Proceedings of the International Conference on Neural Networks* (pp. 175–180). San Francisco, CA: IEEE Neural Network Council.

Thrun, S. (1996). *A Bayesian approach to landmark discovery and active perception for mobile robot navigation* (Technical Report CMU-CS-96-122). Pittsburgh, PA: Carnegie Mellon University, Department of Computer Science.

Thrun, S. (1997). To know or not to know: On the utility of models in mobile robotics. *AI Magazine, 18*, 47–54.

Thrun, S. (in press). Learning maps for indoor mobile robot navigation. *Artificial Intelligence.*

Thrun, S., Bücken, A., Burgard, W., Fox, D., Fröhlinghaus, T., Hennig, D., Hofmann, T., Krell, M. & Schimdt, T. (in press). Map learning and high-speed navigation in RHINO. In D. Kortenkamp, R. Bonasso, & R. Murphy (Eds.), *AI-based mobile robots: Case studies of successful robot systems.* Cambridge, MA: MIT Press.

Thrun, S., Fox, D., & Burgard, W. (in press). A Probabilistic Approach to Concurrent Mapping and Localization for Mobile Robots. *Machine Learning and Autonomous Robots* (joint issue).

Vapnik, V. (1982). *Estimations of dependences based on statistical data.* Berlin, Germany: Springer Publisher.

Wasserman, P.D. (1989). *Neural computing: Theory and practice.* Hew York, NY: Von Nostrand Reinhold.

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

Wolfart, E., Fisher, R., & Walker, A. (1995). Position refinement for a navigating robot using motion information based on honey bee strategies. *Proceedings of the International Symposium on Robotic Systems* (*SIR 95*). Pisa, Italy.

Yamauchi, B., &., Beer, R. (1996). Spatial learning for navigation in dynamic environments. *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics, Special Issue on Learning Autonomous Robots.* Also available at http://www.aic.nrl.navy.mil/∼yamauchi/.

Yamauchi, B., & Langley, P. (1997). Place recognition in dynamic environments. *Journal of Robotic Systems, Special Issue on Mobile Robots*, *14*, 107–120.

Received April 16, 1997
Accepted February 4, 1998
Final Manuscript February 4, 1998