Communicated by Christopher Bishop

A Mobile Robot That Learns Its Place

Sageev Oore
Geoffrey E. Hinton
Department of Computer Science, University of Toronto, Toronto, Canada M5S 1A4

Gregory Dudek
School of Computer Science, McGill University, Montreal, Canada H3A 2A7

We show how a neural network can be used to allow a mobile robot to derive an accurate estimate of its location from noisy sonar sensors and noisy motion information. The robot's model of its location is in the form of a probability distribution across a grid of possible locations. This distribution is updated using both the motion information and the predictions of a neural network that maps locations into likelihood distributions across possible sonar readings. By predicting sonar readings from locations, rather than vice versa, the robot can handle the very nongaussian noise in the sonar sensors. By using the constraint provided by the noisy motion information, the robot can use previous readings to improve its estimate of its current location. By treating the resulting estimates as if they were correct, the robot can learn the relationship between location and sonar readings without requiring an external supervision signal that specifies the actual location of the robot. It can learn to locate itself in a new environment with almost no supervision, and it can maintain its location ability even when the environment is nonstationary.

1 Introduction

We describe a method that allows a robot equipped with a ring of sonar range finders to learn to compute its location within an initially unfamiliar environment. The obvious way to solve this problem is to build an explicit map of the environment. An interesting alternative, however, is for the knowledge of the environment to be implicit in the weights of a neural network that relates sonar readings to locations.

The robot, of small trash-can proportions, has internal sensors that detect the movement in the wheels. These motion sensors may not be entirely accurate (they are blind to skidding) and therefore are not sufficient for accurate navigation over long periods of time. Each of the 12 uniformly spaced Polaroid sonar transducers crowning the robot acts as both transmitter and receiver, measuring the return time of the first echo from an emitted chirp. The time is converted into a distance r, and the vector $\mathbf{r}$ of all 12 readings obtained at a point is called a *sonar signature*. The sonar sensings, too, are affected by various forms of noise, both reproducible and irreproducible. For example, the sound emitted by each sensor spreads out in a cone, and rays within the cone may make different numbers of bounces before returning. This can lead to phantom walls and phantom holes in corners (Dudek, Jenkin, Milios, & Wilkes, 1993; Wilkes, Dudek, Jenkin, & Milios, 1990), depending on the configuration and reflective properties of objects in the room. In addition, faulty sensors, cross-talk between transducers, energy dissipation of the sonar signal, unreturned echos, and very nearby objects can lead to irreproducible random effects that have no apparent structure. Moreover, even the accurate sensings can be assumed to have approximately gaussian additive noise.

In this work, we have used simulator-based data, as will be described in section 3.1, and have assumed that the robot's orientation is known.

2 Learning a Model That Relates Locations to Sonar Readings

2.1 Predicting the Location from the Sonar Readings. There is an obvious way of using a feedforward neural network to derive a location from a set of sonar readings. The inputs to the net are the sonar readings, and the outputs are the location, which may be represented as two real-valued coordinates or may be a probability distribution over a grid of possible locations. Although this type of net works moderately well (Dudek & Hinton 1992), it has some major drawbacks. During the initial training, the network requires supervision in the form of accurate information about the true location of the robot, and further supervision is required whenever the environment changes. The network is also poor at handling situations in which a sonar sensor is unreliable or a small change in location causes the sensor to return the maximum value rather than the distance to a small, close surface. When this happens, the input to the network changes a lot, but the outputs should change only slightly. Such slight changes are hard to achieve in a standard feedforward network because the outputs must be sensitive to at least some of the inputs.

2.2 Integrating Sonar Information over Time. Instead of using a neural network to predict locations from sonar readings, we could predict in the opposite direction and then use Bayes' theorem. For each sonar signature, $\mathbf{r}$, the same feedforward neural network is applied once at each possible grid cell, g_i, to estimate the likelihood, $P(\mathbf{r} \mid g_i)$, of obtaining that signature from that cell. Instead of integrating over all possible locations within each cell, the neural net takes as input the location, $\mathbf{x}_i$, of the center of the cell and produces as output a conditional probability distribution under which the density of $\mathbf{r}$ is computed. The likelihood of $\mathbf{r}$ at each cell is then multiplied by the robot's prior probability of being at the cell, and the products are renormalized[1] to obtain a posterior probability distribution that incorporates the information in the current sonar readings. In the case that we know the environment is stationary (a common assumption), we can improve the efficiency of the tracking algorithm by a large factor, once the neural network has been trained, by keeping the density distribution functions $P(\mathbf{r} \mid g_i)$ in a lookup table, and thus eliminating the need for any more forward passes through the network:

$$P^t_{\text{posterior}}(g_i) = \frac{P(\mathbf{r}^t \mid g_i) P^t_{\text{prior}}(g_i)}{\sum_j P(\mathbf{r}^t \mid g_j) P^t_{\text{prior}}(g_j)}. \tag{2.1}$$

To make the updating straightforward, we need to assume that the prior probabilities of cells are independent of the likelihoods obtained for the current sonar readings. This independence assumption is dubious because the prior probability distribution was derived using the same neural network and the same sensors on earlier nearby locations. To see just how bad the independence assumption can be, imagine what would happen if the robot stayed at exactly the same location and kept repeating its sonar readings. It would eventually be certain that it was in the cell that had the highest likelihood of producing the observed distribution of sonar signatures, even if many other cells had a likelihood almost as high. It is well known that maximum likelihood estimation will yield a biased prediction of the variance of a distribution, which could further increase the chances of confidence in an incorrect position estimate. To check for this effect, we tried using Mackay's (1991) unbiased variance prediction method and found that the results did not differ significantly.

The prior probability distribution is obtained by taking the previous posterior distribution and updating it using the noisy movement information.[2] To eliminate the increase in entropy of the distribution caused by quantization effects when the movement in each direction is not an integral multiple of the grid spacing, we separate the expected movement in each dimension into an integral and a fractional part. The fractional part, $\mathbf{f}^{t-1,t}$, is accumulated by moving the location of every grid cell:

$$\mathbf{x}^t_i = \mathbf{x}^{t-1}_i + \mathbf{f}^{t-1,t}. \tag{2.2}$$

[1] By choosing a fine enough grid, we can assume that the density of the position distribution is locally linear within a cell, and therefore the probability mass contained in the cell is well approximated by using the probability density of a reading being obtained at the center of a cell.

[2] This approach resembles Kalman filtering, but the probability distribution over the underlying state is nongaussian, and the model that relates the underlying state to the observations is nonlinear.

We can imagine this procedure as manipulating a single vector offset associated with the entire grid. For example, suppose each grid cell is 10 cm wide, and a motion of 23 cm to the east is perceived. Then the probabilities in each cell are moved over by two cells, and furthermore the offset of the entire grid is increased by 3 cm (i.e., the centers of the individual cells are themselves displaced). If the offset was already 9 cm, then we would instead move the probabilities over by three cells and subtract 7 cm from the offset, to keep the offset between zero and one cell width.

After moving the probabilities over and adjusting the grid offset, we blur the distribution by the assumed noise, $\sigma^{t-1,t}$ in the movement estimate,

$$P^t_{\text{prior}}(g_i) = \sum_j w_{ji} P^{t-1}_{\text{posterior}}(g_j), \tag{2.3}$$

where the w_{ji} are the terms in a discrete approximation to a gaussian convolution kernel. In our simulation, the actual motion of the robot is a random sample from a gaussian with mean $\boldsymbol{\mu}^{t-1,t}$ and standard deviation $\sigma^{t-1,t}$. We have not added orientation noise to the motion model.

When the boundaries of the environment are known, any probability of being outside the grid can be accumulated in the nearest edge cell. In this way, continual motion in the same direction will eventually lead to near certainty of being somewhere along the corresponding edge.

2.3 Representing the Robot in the Environment. The most convenient and natural way of representing the environment given the nature of our localization method is in the form of a grid, where a value $P(g_i)$ is associated with each cell representing the probability of the robot being in that cell. This is reminiscent of Moravec's (1988) certainty grid but with a key difference: The typical certainty grid lends itself well to constructing a map of the environment, and has been used effectively for that purpose (e.g., see Thrun et al., in press). In our approach the sensor readings are used immediately to update the estimate of the robot's position; using other representations, there needs to be a way of making combined use of the map and the sensor readings to compute the location of the robot *within* that map (e.g., by modeling the inverse sonar process or optimizing a template match).

In our approach the structure of the environment remains implicit in the models that assign probability distributions to sonar readings at each grid location. The integration over time of our knowledge of the environment is handled implicitly in the learning, whereas the integration of motion over time is handled explicitly by moving the probability distribution.

2.4 Learning to Predict Sonar Readings from Location. We considered several different types of networks for estimating the likelihoods of the sonar readings. All of these nets had two input units to represent location coordinates, and the output units were divided into 12 groups, with each group specifying the parameters of a probability distribution for the readings from one sonar sensor. When the true location of the robot was known, the net was trained to maximize the log probability density of the actual sonar readings under the distribution represented by the output values of the net when the inputs were the true location.

Initially we used a single hidden layer of logistic units and modeled the probability distribution for each sonar sensor as a mixture of a gaussian and a uniform distribution between 0 and the maximum value of 4 m. This required three output units per sonar sensor: a logistic unit for the mixing proportion of the uniform distribution, a linear unit for the mean of the gaussian, and an exponential unit for its variance. The use of an exponential is appropriate for a scale parameter like the variance, and it prevents the variance from ever becoming negative or zero.

In our initial model, the predicted mean of the gaussian could be a nonlinear function of the two input coordinates because we used logistic hidden units. Later we found that it was better to use a different type of nonlinearity to predict the mean of the gaussian from the input coordinates. The mean is typically a linear function of the input coordinates over a small region but then switches to a different linear function when the sonar return comes from a different planar surface (see Fig. 1), so it is natural to use a mixture of linear models for predicting the mean. We used 12 separate networks, one for each sonar sensor; each such network, n, implemented a mixture of local models. Each local model was itself a mixture of a gaussian and a uniform. The mixing proportions of the local models were a function of the current input coordinates and were specified by the activity levels of a group of normalized radial basis units. The activity level of each basis unit, j, was determined by the relative proximity of its center, $\mathbf{c}_{nj}$, to the two input coordinates, $\mathbf{x}$:

$$\psi_{nj}(\mathbf{x}) = \frac{\exp\left[\frac{-\|\mathbf{x}-\mathbf{c}_{nj}\|^2}{2\sigma^2_{RBF_n}}\right]}{\sum_k \exp\left[\frac{-\|\mathbf{x}-\mathbf{c}_{nk}\|^2}{2\sigma^2_{RBF_n}}\right]}. \tag{2.4}$$

Both the centers of the units and their shared variance $\sigma^2_{RBF_n}$ were learned.[3]

Each basis unit has three associated output units that represent the parameters of a predicted probability distribution for one sonar reading. This distribution is a mixture of a gaussian and a uniform. The basis unit has one adaptive outgoing weight that determines the mixing proportion, z_j, of the gaussian, one adaptive weight for the log of the variance, and three more

[3] The planar walls of the room cause the piecewise linear regions in Figure 1 to have linear boundaries. This makes it appropriate for the radial basis functions to have the same variances because this ensures that, after normalization, the region dominated by an RBF has linear boundaries.

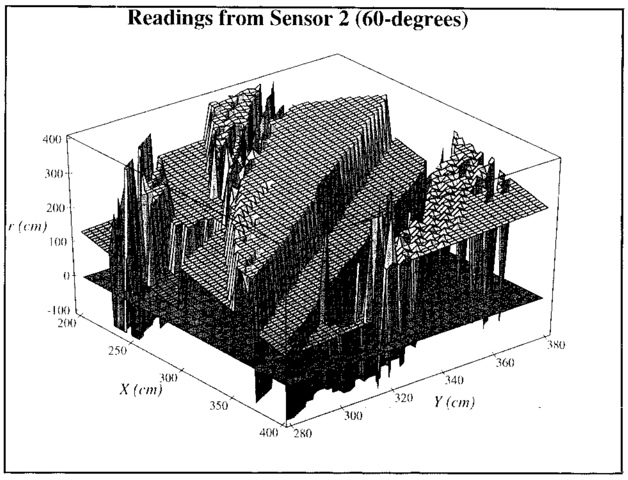

Figure 1: Readings from sensor 2. East-west and north-south positions are plotted along the x- and y-axes, respectively. Signatures were taken at every 2.5 cm inside a 2 m $\times$ 1 m rectangular region in the middle of a cluttered room. The measurements obtained by sensor 2 are plotted on the vertical axis. The measurements that drop below the 0 plane indicate locations where the sonar pulse did not return a detectable echo. Many of the vertical surfaces in the room are (nearly) orthogonal to the x- or y-axes, but sensor 2 is not parallel to either axis of the room. So sensor 2 tends to produce signals that have more multiple reflections and therefore a greater number of smaller discontinuous pieces.

outgoing weights, $\mathbf{a}$, b, that define a linear model, which predicts the mean of the gaussian from the input coordinates. A picture of the network for a single sensor is shown in Figure 2. The likelihood distribution produced by basis unit j is:

$$P_{nj}(r_n \,|\, \mathbf{x}) = z_{nj}\frac{1}{2\pi\sigma_{nj}^2}\exp\left[\frac{-\|r_n - \mathbf{a}_{nj}\mathbf{x} - b_{nj}\|^2}{2\sigma_{nj}^2}\right] + \frac{1 - z_{nj}}{r_{\max}}. \qquad (2.5)$$

We have allowed each region an adaptive uniform noise mixing component because it is possible that the noisy, or unpredictable, readings are a function not only of the sensor but of the location of the robot as well.

The normalized activity levels of the basis units are treated as mixing proportions for combining their likelihood distributions into a single likelihood

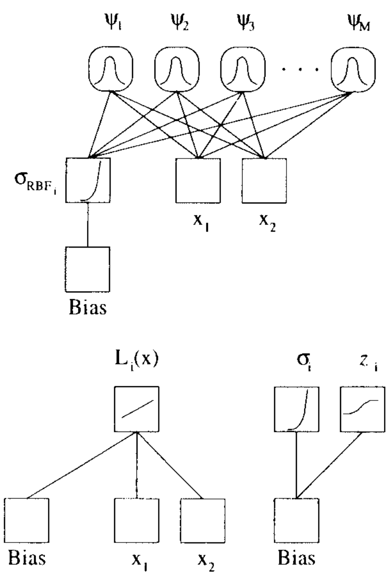

Figure 2: Final network model for one sonar sensor. The upper network shows the normalized radial basis function units. A single variance σ^2_{RBF} is inferred for all regions of each sensor model. For each of the M regions, a linear unit is used to predict the mean of the gaussian $L_i(\mathbf{x})$, exponential units are used to learn variance, and sigmoid units are used to learn mixing proportions for uniform noise.

distribution for each sensor n:

$$P_n(r_n \mid \mathbf{x}) = \sum_j P_{nj}(r_n \mid \mathbf{x})\psi_{nj}(\mathbf{x}). \tag{2.6}$$

The number of normalized radial basis functions (RBFs) needed to reach a certain positional accuracy is related to the size and complexity of the environment. When a local model is responsible for only a single data point, there is a danger that it will assign an extremely high-probability density to the observed reading by collapsing the width of the gaussian component of its output distribution toward zero. This form of overfitting is avoided by tying together the widths of the output gaussians for all the local models for one sonar sensor. Generally the more RBFs used, the better the performance tended to be, at the expense of learning time. We investigated some methods for pruning and adding RBF units during learning, with encouraging results, though beyond the scope of the current discussion (e.g., this could be very useful in an environment that changes drastically over time).

When $\mathbf{x}$ is known, all parameters—$\mathbf{a}$, b, and σ, as well as those associated with the RBF—can be learned by maximizing $\log P_n(r_n \mid \mathbf{x})$ (i.e., by gradient descent), but when the robot's true location is unknown and all that is available is the robot's current probability distribution over cells, it is less obvious what should be optimized. Our unsupervised algorithm, which is closely related to expectation maximization, maximizes the expected log likelihood of the sonar signatures, where the expectation is taken with respect to the posterior probabilities of being at each grid cell:

$$\langle \log P(\mathbf{r} \mid \mathbf{x}) \rangle_{P(\mathbf{x})} = \int_x \log P(\mathbf{r} \mid \mathbf{x}) P(\mathbf{x}) d\mathbf{x} \tag{2.7}$$

$$\approx \sum_i \log P(\mathbf{r} \mid g_i) P_{\text{posterior}}(g_i) \tag{2.8}$$

where $P_{\text{posterior}}(g_i)$ was calculated in equation 2.1. We used the conjugate gradient method to tune the parameters.

3 Experimental Results

3.1 The Simulated Environment and Noise. The environment and sonar data used in this work were generated synthetically by a sophisticated simulator provided by Wilkes et al. (1990). This simulator used a ray-tracing algorithm to capture many of the reproducible forms of nongaussian noise that can occur with sonar sensors, such as phantom holes, phantom walls, and occasional failures to detect nearby objects. Furthermore, to account for the noise that is irreproducible, we have made the worst-case assumption by replacing a certain fraction of the readings by random values chosen from a uniform distribution in the interval $(0, r_{\text{max}})$. This replacement was done by assigning each sensor i an "unreliability" quotient q_i, so that any reading taken by sensor i would have a probability q_i of being replaced by a random number. The unreliability quotients were $\{0.1, 0.5, 0.001, 0.01, 0.1, 0.001, 0.1, 0.2, 0.001, 0.001, 0.1, 0.05\}$. In addition, to make the task more difficult, all sensor readings that attained the maximum value r_{max} were also replaced by a random number. Even for sensor readings considered "reliable," gaussian noise was added (with standard deviations of $\{4, 4, 3, 3, 5, 5, 5, 5, 2, 1, 2, 10\}$ cm).

Note that the net does not attempt to model the relationship between the value of a sonar reading and the distance to the nearest object, only the relationship between the value of the sonar reading and the position of the robot. This means that any reproducible effects can potentially be utilized for better localization, including the "nearby objects appearing far away" effect, as long as there is some degree of consistency some of the time. The fraction of the time when there is no consistency can simply be modeled by the uniform noise-mixing proportion.

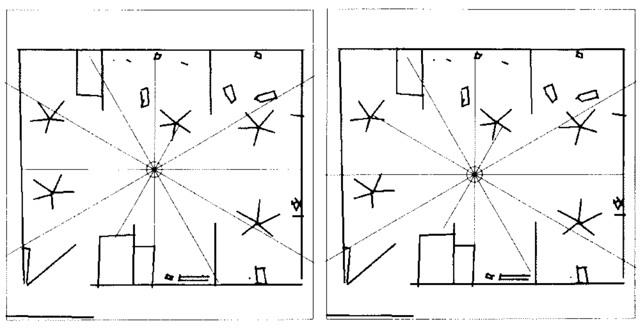

Figure 3: Top-down view of complete sonar signatures at two points 10 cm apart. The position of the robot is marked by the circle near the middle of the room (the one on the left is just slightly further north than the one on the right). The thicker lines are meant to be reflective surfaces: walls, dividers, chair legs, feet, a soda can under a desk, and a door. Each of the thin lines centered at the robot represents the distance that was measured by the sensor facing in that direction. For example, using a clock numbering of the sensors, the distance to the east wall was accurately measured in both cases by sensor 3, while some confusion occurred with sensor 9 as to the location of the western wall due to the interference of a chair base on the right. The lines that touch the outer boundaries—the measurements of sensor 2 and sensor 4—represent the directions from which no strong echo was returned.

The environment used in all of the following tests was a cluttered office, about 5 m by 5 m. Figure 3 illustrates the environment and signatures taken from two nearby positions. Note the difference in the two signatures. These signatures have not yet had any random noise added to them. We believe the clutter in the environment to be a realistic model of working environments, where numerous objects are left all over the place.

3.2 Tracking Algorithm. The unsupervised algorithm relies on the premise that combining the knowledge of the motion with the sonar-based network predictions will produce better position estimates than those that would be obtained by using either source of information alone. To demonstrate that this is typically true, we used supervised training of the network and then compared the errors of the dead-reckoned estimates (motion alone), the network estimates (sonar alone), and the filtered estimates (sonar and motion combined) as the robot travels along a path. The position is actually represented by a probability distribution over a grid, but for these comparisons, we used the mean of the distribution as the estimate.

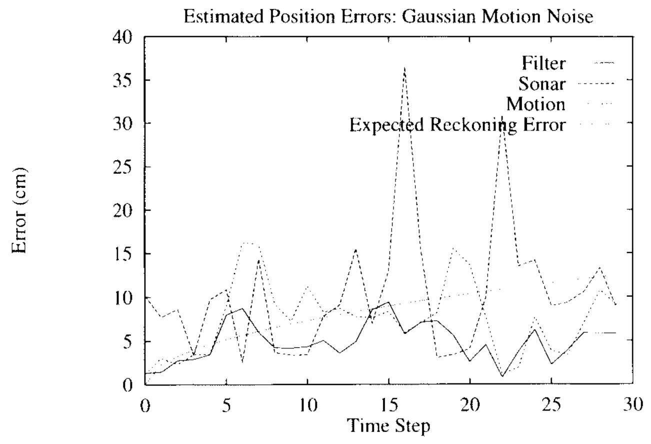

Figure 4: Absolute errors in the predicted position for a single path. The noise in the motion sensors is multiplicative gaussian with a mean of zero and standard deviation of $\sigma_{\text{motion}} = 5\ cm$ for every $20\ cm$ of motion.

In Figure 4, a network was first trained on a set of 80 labeled examples taken from a boxed region of the position space, and then a random path 30 steps long was traversed without giving position information. The tracking algorithm is successful; the errors of the filtered predictions in Figure 4 are roughly equal to the smaller of the sonar- and motion-based prediction errors. Also shown is the expected value of the absolute error that would be obtained over many runs of a single path using the same noise model for the motion sensors. This smooth curve would continue to increase as the square root of the number of time steps. The variance in the errors of the dead-reckoned predictions would increase as well.

At the early stages of the unsupervised learning, the network will typically make larger errors, but these will be more easily overwhelmed by the motion constraints, since the distributions predicted by the net will have a higher entropy. In Figure 5, a network was used that had been trained on only 20 examples. Although the variance in the sonar sensor predictions was considerably higher, the filtered predictions were still quite good. We also did runs in which we kept the robot's motion model as described above (i.e., zero-mean gaussian), but affected the motion sensings with a systematic 10% overestimation error in addition to the gaussian noise (i.e., biased noise). Still, the filtered errors generally remained below 10 cm (Oore, 1995).

A minimally trained network can be quite useful provided the net does not give too much confidence to its incorrect predictions. Occasional, accurate, high-certainty predictions made by the net cause a confident posterior

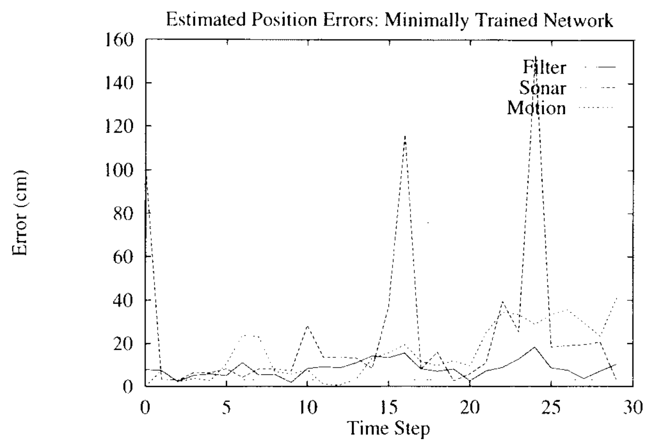

Figure 5: Errors in the predicted position for a single path. The motion sensings were affected by a systematic error (in which all traveled distances were overestimated by a factor of 10%) as well as a gaussian noise process with a larger variance, $\sigma_{\text{Motion}} = 10$ cm for every 20 cm traveled. The net was trained on 20 examples.

estimate. The motion model will then maintain this accuracy for a number of time steps, even if the subsequent predictions by the network are poor.

3.3 Unsupervised Learning. The motivation for an unsupervised method for robot learning begins with the dilemma that dead reckoning alone is insufficient but manual recalibration is too expensive to be practical. The aim of the unsupervised learning procedure is to discover a good generative model of the relationship between location and sonar readings. If the true location is known, this can be done by maximizing the likelihood of each observed sonar reading under the probability distribution produced by the model. Since the true location of the robot is unknown, the best we can do is to estimate the probability that it is at each grid location and then weight the learning by these estimated probabilities.

3.3.1 Training Procedure. The general training procedure used for the unsupervised learning consisted of a few steps:

0. **Initialization.** The 12 networks for predicting the likelihood distribution of a sonar reading from the coordinates of a grid point were initialized with small random weights, except that the normalized RBF centers were distributed uniformly over the environment; the

robot was placed in the room without any position information (i.e., a uniform distribution over the grid).

1. **Data collection.** The robot traversed a short path, consisting of a sequence of steps. An obstacle-avoidance mechanism was assumed in generating the path. A sonar signature was taken at each step, and the filtering algorithm given by equation 2.1 was used to update the probability distribution over grid cells by integrating the motion and sonar sensings. The updated distribution over cells was combined with the current sonar signature to yield weighted training examples of grid cell–sonar signature pairs. The examples were accumulated over all the steps in the path.

2. **Training.** Using the data just collected, the network was trained for a fixed number of iterations, after which the training set was discarded. The last filtered position distribution from the previous path was retained as the initial distribution when returning to step 1 to acquire more data. The weight vector of the trained net was retained as the starting point for the next minimization.

Although the paths were random, they were generated to drift slowly from left to right and back again (and again) while drifting up and down more quickly.

3.3.2 Performance of the Unsupervised Learning Algorithm. The cluttered office environment was used, with a motion noise of $\sigma_{\text{Motion}} = 2.5$ cm for every 20 cm traveled. The network was trained for 30 conjugate gradient iterations after every 30 steps. Figure 6 shows the actual paths traveled and the paths as predicted from the sonar readings alone by the network, after 3 and then after 35 paths. The first peculiarity to observe is that although the filtered estimates correctly capture the shape of the true path, they are a translation of it. This can be explained by noting that the motion sensings provide relative position information, while the only indications of absolute position are an initial distribution, if given, and the edges of the grid if they are reached by the exploration. Thus, since the paths we used systematically continued to touch the east-west walls but did not extend fully in the north-south direction, it is not surprising that the robot constructed a model that was self-consistent (that is, consistent with the motion constraints) but was a vertical translation of the actual environment. Examining the sonar-based errors, we can see that in the earlier predictions, the network seemed to get the right vicinity (modulo a translation) but made gross errors. After continuing to train over another 32 paths, the network learned a model that was a more accurate translation of the true environment. In both cases, the majority of the larger errors in the sonar predictions were corrected by the filtering algorithm.

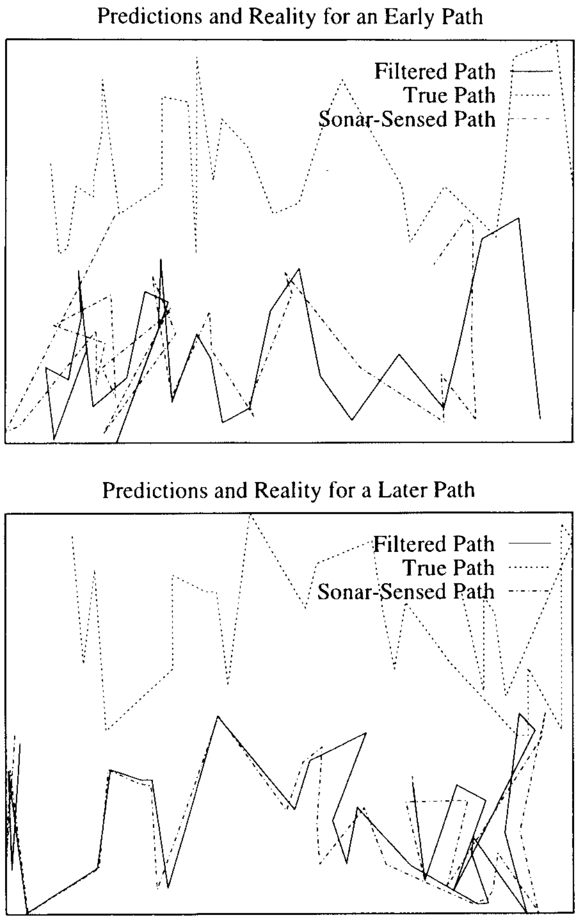

Figure 6: Real vs. predicted paths. The axes of the graphs can be seen as representing a top-down view of a portion of the environment, in which the robot took a zigzag path going from west to east. For visual clarity, the paths have not been adjusted to match up. Inspection will reveal that the sonar sensings in the lower left-hand region of the earlier path are very loosely related to the shape of the true path. Nevertheless, it is immediately clear that they correspond to the right general region. In contrast to this, looking at the right-hand side of the later path shows that the sonar-based estimates exhibit a much closer match to the filtered and true positions.

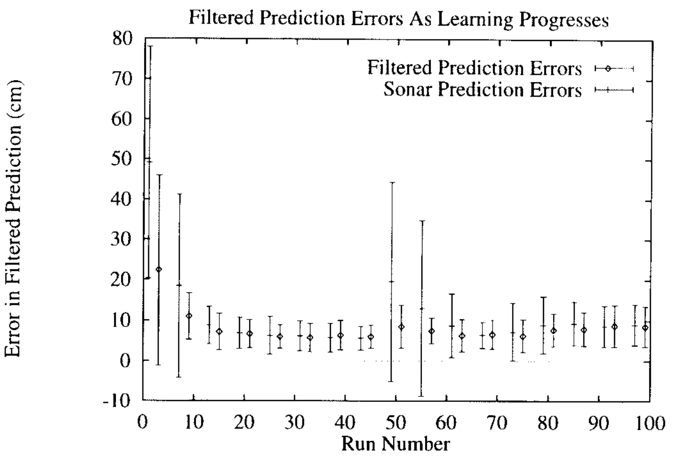

Figure 7: Learning while exploring. The means were calculated over sets of paths, and the standard deviations were estimated from the sample, assuming gaussian noise (which is incorrect for values that cannot be negative).

Figure 7 shows the filtered and sonar errors with error bars for paths as the learning progresses, after performing the appropriate translation.[4] We see that the filtered predictions were almost always better and had significantly smaller error bars than the sonar predictions. At around the fiftieth path, the variance in the sonar errors suddenly increased, for at this point, the environment was rearranged (chairs were added, moved, etc.). The average error did not increase by much because it was only in certain parts of the room that the network's predictions were affected. This allowed the motion constraints to maintain accurate localization for a few time steps and then, before the filtered predictions started to deviate by much, a few confident and accurate sonar predictions in a familiar part of the room were all that were needed to recalibrate the tracking effectively. Since the normalized RBFs respond locally, they can quickly and easily adapt to local changes in the room. Although the network acted desirably in this case, problems can arise if too many consecutive, overconfident, but inaccurate sonar predictions are made. The threshold that determines overconfidence is, in part, a function of the noise model in the motion sensings.

A possible danger is that if a certain region is avoided during initial training because of an obstacle, then when that obstacle is later removed, there may still be a hole in the posterior where the object used to be. However,

[4] Done by calculating the mean error over a path and subtracting this from each prediction in the path.

this will be prevented to a certain extent for two reasons. First, during the initial period of training, if the distribution of the robot's position is very entropic, the parameters for the distributions will be learned so that the entire position space is covered, and therefore those places that are not contained in the posterior distributions at later stages will still have been reasonably initialized early on. Second, for many of the sensor readings (in particular, those that are not pointing at the obstacle in question), the readings on either side of the obstacle are quite likely to belong to the same linear region, and therefore the corresponding linear predictors are already tuned to interpolate correctly through the occupied region.

Various modifications in the training procedure can significantly improve the tracking and learning. For example, the learning can be accelerated by starting with a net trained on a small number of labeled examples rather than using a purely random initial net. Alternatively, the initial net can be untrained, but accurate position information can be given at the beginning of each of the first few paths. If the robot is returned to the same part of the room to start each path, it learns to make confident and accurate predictions in that region, and with further exploration it gradually expands the region in which its model is accurate. The exploration strategy and length of paths between minimizations are also important. If too many of the training points in one path come from the same part of the room, then the centers of the RBF units need to be frozen to prevent an uneven allocation of the network's resources. Although some of these variations speeded up the learning, they were not crucial and were not used in the runs we have described.

4 Relationships to Other Approaches

4.1 Hidden Markov Models. The cells in the grid correspond to the states in a hidden Markov model, and the uncertainty in the movement information corresponds to the matrix of transition probabilities between states. The fact that this matrix changes at each time step is unusual but not incompatible with the standard hidden Markov model framework. Instead of having a separate output model at each state, all of the states share the same 12 neural networks, but the likelihood distributions produced by these nets are contingent on the real-valued location coordinates associated with a state. These coordinates change in a deterministic way as the robot moves, but this does not create any additional problems.

In estimating the parameters of the output models, we make one major simplification relative to the Baum-Welch method that is normally used for hidden Markov models (Baum & Eagon, 1967). The correct way to compute the posterior probability distribution over cells given a sequence of vectors of sonar readings is to use the forward-backward algorithm. Repeated application of equations 2.1, 2.2, and 2.3 corresponds to the forward pass, but we ignore the backward pass and so cannot make use of the information that sonar readings at time t provide about the true location of the robot at times earlier than t. The advantage of our suboptimal method of estimating posterior probabilities is that it can be used online. Also, Neal and Hinton (1993) show that the convergence of expectation maximization does not require the correct posterior distribution to be used. All that is required is that the E-step produce a distribution that has a smaller Kullback-Liebler distance from the true posterior than the previous estimate.[5] This is no longer guaranteed when the backward pass is ignored, but it is unlikely that a forward pass using the current parameters in the output models will give a worse estimate of the true posterior given those parameters than a forward pass using the previous parameters.

4.2 Kohonen-Style Self-Organizing Maps. Kröse and Eecen (1994) show that it is possible to compute the location of a robot by first building a topographic map from the vectors of sonar readings alone. The use of a self-organizing map (SOM) partially overcomes the need for labeled training data, but unfortunately, the function that is optimized is not appropriate for the location task. If we view the SOM in generative terms, it attempts to minimize the squared reconstruction error when the input vector is reconstructed from the winning hidden unit or from one of its neighbors. This would be the right thing to minimize if the constraint on sonar readings was that neighboring locations produced similar readings. But this is not true when the robot passes close to the edge of a surface. Provided we know that the movement is small, a much more realistic constraint is that temporally adjacent readings come from neighboring locations. This is precisely the constraint captured by our approach if we treat the movement as random gaussian noise. Obviously, if we know more about the precise movement, we can make this constraint stronger.

The relationship between our method and the SOM can be understood by viewing them as two quite different elaborations of a simple mixture model in which each hidden unit has a vector of 12 parameters that represent expected values for the 12 sonar readings. The SOM learning algorithm elaborates this model by forcing neighboring hidden units to have similar parameter vectors. It also uses a hard decision rule for deciding on a winner instead of using posterior probabilities to weight the learning. We elaborate the simple mixture model by using nongaussian output models for each cell in the grid and by tying all these models together to reduce greatly the number of free parameters. Different cells can still predict different probability distributions for the sonar readings because the predictions are conditional on the location coordinates of the cell. Finally, instead of just using the current sonar readings to compute the posterior probability of a cell, we take into account the earlier readings and the motion information, thus allow-

[5] Surprisingly, the relevant Kullback-Liebler distance is $\sum_\alpha Q_\alpha \log(Q_\alpha/P_\alpha)$ where Q is the estimated probability distribution and P is the true posterior.

ing the temporal constraints that are ignored by the SOM to influence the learning.

Acknowledgments

This research was funded by the Institute for Robotics and Intelligent Systems and by the Natural Science and Engineering Research Council. Hinton is the Nesbitt-Burns fellow of the Canadian Institute for Advanced Research. We thank Radford Neal, Allan Jepson, and Brendan Frey for helpful comments.

References

Baum, L. and Eagon, J. (1967). An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology. *Bulletin of the American Mathematicians Society*, 73.

Dudek, G., & Hinton, G. (1992). *Navigating without a map by directly transforming sensory input to location*. Unpublished manuscript.

Dudek, G., Jenkin, M., Milios, E., & Wilkes, D. (1993). *Reflections on modelling a sonar range sensor* (Tech. Rep. No. CIM-92-9). Montreal: McGill Research Centre for Intelligent Machines, McGill University.

Kröse, B., & Eecen, M. (1994). A self-organizing representation of sensor space for mobile robot navigation. In *Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems* (pp. 9–14). New York: IEEE.

Mackay, D. (1991). *Bayesian modelling and neural networks.* Unpublished doctoral dissertation, California Institute of Technology, Pasadena, CA.

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

Neal, R., & Hinton, G. (1993). *A new view of the EM algorithm that justifies incremental and other variants*. Unpublished manuscript.

Oore, S. (1995). A mobile robot that learns to estimate its position from a stream of sonar measurements. Unpublished master's thesis, University of Toronto.

Thrun, S., Buecken, A., Burgard, W., Fox, D., Froehlinghaus, D., Henning, D., Hofmann, T., Krell, M., & Schmidt, T. (In press.) Map learning and high-speed navigation in RHINO. In D. Kortenkamp, R. Bonasso, & R. Murphy (Eds.), *Case Studies of Successful Robot Systems.* Cambridge, MA: MIT Press.

Wilkes, D., Dudek, G., Jenkin, M., & Milios, E. (1990). A ray following model of sonar range sensing. In *Proc. Mobile Robots V*, pp. 536–542. Bellingham, WA: International Society for Optical Engineering.

Received December 5, 1995; accepted August 2, 1996.