Dynamic Sensory Probabilistic Maps for Mobile Robot Localization

N. A. Vlassis* G. Papakonstantinou P. Tsanakas

Dept. of Electrical and Computer Engineering
National Technical University of Athens
Zographou Campus, 15773 Athens, Greece
http://www.dsclab.ece.ntua.gr

Abstract

In order to localize itself, a mobile robot tries to match its sensory information at any instant against a prior environment model, the map. A probabilistic map can be regarded as a model that stores at each robot configuration q the probability density function of the sensor readings at q. By combining the knowledge of its current position, the new-coming sensory information, and the probabilistic map the robot is capable of improving its prior position estimate. In this paper we propose a novel sensor model and a method for maintaining a probabilistic map in cases of dynamic environments. When the environment structure changes, the map must adapt to this change by modifying the sensor densities at the respective configurations. We propose a combined algorithm for map update and robot localization.

1 Introduction

Recently, there has been an increasing interest in the mobile robots community in probabilistic models for robot localization and navigation in metric maps [7, 1, 4, 11, 8]. The key issue in most of these approaches is a *probabilistic map*, i.e., an assignment to each robot's configuration $\mathbf{q}$ of a probability density function that models the uncertainty of the sensor readings when observing an environment landmark from $\mathbf{q}$.

At any instant, the robot combines the current knowledge of its position together with the external sensory information and updates its position estimate by reasoning on the probabilistic map. The probabilistic approach becomes a convenient framework for mobile robot perception and navigation because it combines neatly the inherent uncertainty in both the robot motion and the sensor devices.

In most applications, the probabilistic map is built off-line from a CAD model of the environment [7, 1, 8]. A probabilistic model of the sensor device is applied at each possible robot configuration and it is simulated the density of the sensor readings at that particular point. This is the approach followed by [7], while in [1] the sensor uncertainties are additionally mapped onto an occupancy grid [2].

In [8] the density estimation is performed only at specific landmarks of the environment and an on-line procedure is proposed which combines map generation with on-line robot localization based on the Baum-Welch method for Maximum Likelihood estimation of the parameters of a hidden Markov model. Finally, in [4, 11] neural-network models based on probabilistic maps are proposed for position estimation and path planning.

In this paper we propose a method for building and maintaining sensory probabilistic maps based on proximity sensor information that can be changing with time. As in [1], we assume a proximity sensor s that measures at each robot configuration $\mathbf{q}$ a distance value r modeled as a random variable with density $p(r|s,\mathbf{q})$, conditioned on s and $\mathbf{q}$. To allow for non-Gaussian noise in the sensor devices due to, e.g., unreturned echoes, cross-talk, etc., we choose for the density $p(r|s,\mathbf{q})$ a particular parametric model, namely, the *finite Gaussian mixture* model [9]. In order to build and maintain a probabilistic map, the parameters of the kernels of the Gaussian mixture, the mixing weights, and the number of Gaussian kernels must be estimated from all the sensor readings at each robot configuration.

In the following we describe the proposed sensor model and also an estimation procedure that is based on the Maximum Likelihood technique for density es-

*The author is currently with the University of Amsterdam, Dept. of Computer Science, Kruislaan 403, NL-1098 SJ Amsterdam, The Netherlands, http://www.wins.uva.nl/research/ias.

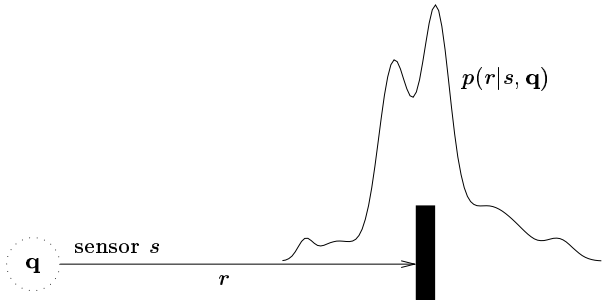


Figure 1: A non-Gaussian density $p(r|s,\mathbf{q})$ for a sensor s firing from configuration $\mathbf{q}$ modeled as a finite mixture of Gaussian kernels.

timation, combined with statistical tests to make it capable of handling nonstationary sensor distributions. This way, dynamic environments, i.e., environments where the sensor distributions over the robot's configuration space change with time, can be effectively modeled.

2 The sensor model

Consider a proximity sensor s measuring a distance r between a robot configuration $\mathbf{q}$ and a nearby obstacle, as shown in Fig. 1. To allow non-Gaussian noise in the sensor readings we decide to approximate the density of the measurements with a finite weighted sum of Gaussian densities, or kernels, as

$$p(r|s,\mathbf{q}) = \sum_{j=1}^{K_{sq}} \pi_{sqj} f_j(r|s,\mathbf{q}), \tag{1}$$

where $f_j(r|s,\mathbf{q})$ denotes the Gaussian density $N(\mu_{sqj}, \sigma_{sqj})$

$$f_j(r|s,\mathbf{q}) = \frac{1}{\sigma_{sqj}\sqrt{2\pi}} \exp[-\frac{1}{2\sigma_{sqj}^2}(r-\mu_{sqj})^2], \tag{2}$$

parametrized on the mean μ_{sqj} and the variance σ_{sqj}^2, while K_{sq} denotes the number of kernels, all parameters conditioned on s and $\mathbf{q}$. To ensure that the integral of $p(r|s,\mathbf{q})$ over the measurements space equals 1, we set the additional constraints

$$\sum_{j=1}^{K_{sq}} \pi_{sqj} = 1, \quad \pi_{sqj} \geq 0. \tag{3}$$

2.1 Maximum Likelihood estimation

During robot navigation the map must be continuously updated to conform with the new sensory information, and for a sensor s and a robot configuration $\mathbf{q}$ this corresponds directly to updating $p(r|s,\mathbf{q})$ based on the incoming sensor readings. Let us assume that while robot is at $\mathbf{q}$, a series of contiguous sensor readings $\{r^1,\ldots,r^n\}$ arrive.[1] To estimate $p(r|s,\mathbf{q})$ from these readings we use the Maximum Likelihood method. Dropping temporarily the dependence on s and $\mathbf{q}$ of (1) we aim at maximizing the likelihood or equivalently the logarithm

$$\log p(r^1,\ldots,r^n) = \sum_{i=1}^{n} \log p(r^i). \tag{4}$$

Differentiating with respect to a parameter θ_j, with $\theta_j = \mu_j$ or $\theta_j = \sigma_j^2$, and using (1) and (4) we get

$$\sum_{i=1}^{n} \frac{1}{p(r^i)} \frac{\partial p(r^i)}{\partial \theta_j} = \sum_{i=1}^{n} \frac{\pi_j}{p(r^i)} \frac{\partial f_j(r^i)}{\partial \theta_j} = 0. \tag{5}$$

Using the continuous version of Bayes' theorem [5] we can estimate the probability $P\{j|r^i\}$ that a sensor reading r^i originated from kernel j as

$$P\{j|r^i\} = \frac{\pi_j f_j(r^i)}{p(r^i)}. \tag{6}$$

Eq. (5) then reads

$$\sum_{i=1}^{n} \frac{P\{j|r^i\}}{f_j(r^i)} \frac{\partial f_j(r^i)}{\partial \theta_j} = \sum_{i=1}^{n} P\{j|r^i\} \frac{\partial \log f_j(r^i)}{\partial \theta_j} = 0. \tag{7}$$

Substituting from (2) yields the Maximum Likelihood (ML) estimates of the mean μ_j and variance σ_j^2 of each kernel j as

$$\mu_j = \frac{\sum_{i=1}^{n} P\{j|r^i\} r^i}{\sum_{i=1}^{n} P\{j|r^i\}}, \tag{8}$$

$$\sigma_j^2 = \frac{\sum_{i=1}^{n} P\{j|r^i\} (r^i - \mu_j)^2}{\sum_{i=1}^{n} P\{j|r^i\}}. \tag{9}$$

The ML estimates for the weights π_j are obtained by introducing a Lagrange multiplier λ for the condition $\sum_{j=1}^{K} \pi_j = 1$ and finding the roots of the first partial derivative with respect to π_j of the quantity

$$\sum_{i=1}^{n} \log p(r^i) - \lambda(\sum_{j=1}^{K} \pi_j - 1), \tag{10}$$

which after simple derivations yields

$$\pi_j = \frac{1}{n} \sum_{i=1}^{n} P\{j|r^i\}. \tag{11}$$

[1] We have made here the implicit assumption that sensing has a higher rate than control. Cf. [8] where sensing and motion are alternated.

After some algebraic manipulation on (8), (9), and (11) one arrives at iterative approximate formulas for the quantities μ_j, σ_j^2, and π_j of a kernel j as

$$\mu_j := \mu_j + \frac{P\{j|r\}}{n\pi_j}(r - \mu_j), \tag{12}$$

$$\sigma_j^2 := \sigma_j^2 + \frac{P\{j|r\}}{n\pi_j}[(r - \mu_j)^2 - \sigma_j^2], \tag{13}$$

$$\pi_j := \pi_j + \frac{1}{n}(P\{j|r\} - \pi_j). \tag{14}$$

Although the above formulas (12)–(14) approximate well the ML estimates of the weights π_j and the parameters μ_j and σ_j of a kernel j, they provide no means for deriving the proper number of kernels for the sensor distribution. For this, we form a test statistic on a weighted formula of the *kurtosis*, or fourth moment, of a kernel j as

$$k_j := k_j + \frac{P\{j|r\}}{n\pi_j}\left[\left(\frac{r - \mu_j}{\sigma_j}\right)^4 - k_j - 3\right], \tag{15}$$

with μ_j and σ_j the current ML estimates for the parameters of the kernel. On the hypothesis of normality we expect that the random variable

$$k_j\sqrt{n\pi_j/96} \tag{16}$$

approximately follows Gaussian distribution $N(0, 1)$, and thus we can split a kernel j if the absolute of this value becomes higher than a specific threshold, e.g., 3 [10]. After splitting a kernel we create two kernels with means $\mu_j + \sigma_j$ and $\mu_j - \sigma_j$ and variances and weights both equal to the original variance and weight, respectively. The weights of all kernels are renormalized to sum one.

Similarly, we maintain that two kernels j and k can join in one if the ratio of the larger to the smaller variance, e.g., σ_j^2/σ_k^2, and the random variable

$$\sqrt{n\pi_j}\,\frac{\mu_j - \mu_k}{\sigma_j\sigma_k\sqrt{2}} \tag{17}$$

are below some thresholds, e.g., 1.5 and 3, respectively. Also we delete a kernel j if its weight π_j falls below $1/n$, a threshold which ensures that the terms in (12) and (13) remain bounded. After join or deletion all kernels update their weights to unity. For more detailed derivations the reader may refer to [10].

We should note here that the number n in the above formulas is constant, e.g., $n = 100$. This has two effects: first, even nonstationary sensor distributions can be handled appropriately with our model, because the previous gradient descent method for maximizing the likelihood does not need to converge to a stochastic steady state. When the sensor distribution changes, the statistical tests (15)–(17) ensure a fast adaptation to the new state by removing and creating kernels where needed, while at the same time the iterative formulas (12)–(14) tune the parameters of the new kernels appropriately (see examples below).

Second, constant n means that the robot does not have to wait at each configuration for a specific number of sensor readings before moving on. As long as it remains at a particular configuration and sensor readings arrive, the respective parameters are iteratively updated using the formulas (12)–(15). We show next how this process is interleaved with robot localization.

3 Robot localization and map update

The method we described in the previous section can be used for building and maintaining a probabilistic map from the sensor readings when the exact configuration of the robot is known. In this section we extend the previous derivations to cases where there is some sort of uncertainty associated with the robot configuration at any moment. It turns out that robot localization and map estimation must be interleaving tasks.

One approach that is often used for mobile robot navigation and localization under uncertainty [6, 1, 4, 8] is to maintain a probability distribution over the robot's configuration space $\mathcal{C}$ and continuously update it using the motion and sensing information. The discrete equivalent is to assign to each configuration $\mathbf{q}$ a probability mass $P(\mathbf{q})$ expressing at any moment the robot's belief for being at $\mathbf{q}$, while it must hold

$$\sum_{\mathbf{q}\in\mathcal{C}} P(\mathbf{q}) = 1. \tag{18}$$

Whenever the robot moves, the entire distribution is first shifted by the associated displacement and then it is blurred by a convolution kernel f, e.g., a discrete multivariate Gaussian density $N(\mathbf{0}, \mathbf{\Sigma})$ with zero mean and covariance matrix $\mathbf{\Sigma}$, implied by the assumed motion error, as

$$P(\mathbf{q}_k) = \sum_{\mathbf{q}_l\in\mathcal{C}} f(\mathbf{q}_k - \mathbf{q}_l)P(\mathbf{q}_l), \quad \forall\mathbf{q}_k \in \mathcal{C}. \tag{19}$$

When the robot senses, the probability masses at each configuration $\mathbf{q}$ must be updated for each sensor s from the estimated density $p(r|s, \mathbf{q})$ (cf. Eq. (1)). If

we assume independence of sensor readings we can use the joint density of the sensor signature $\mathbf{r}$ at each $\mathbf{q}$, i.e.,

$$p(\mathbf{r}|\mathbf{q}) = \prod_s p(r|s,\mathbf{q}) \tag{20}$$

where each measurement r is modified by simple geometrical transformations according to the distance and angle of each $\mathbf{q}$ from the mode of $P(\mathbf{q})$ and excluding from the product those measurements that imply a moved obstacle, i.e., those with almost zero likelihood. We then update the configuration probability masses using the Bayes' formula as

$$P(\mathbf{q}) := \frac{p(\mathbf{r}|\mathbf{q})P(\mathbf{q})}{\sum_{\mathbf{q}_l \in \mathcal{C}} p(\mathbf{r}|\mathbf{q}_l)P(\mathbf{q}_l)}. \tag{21}$$

After updating the configuration probabilities, and until the next motion command is issued, we update the density $p(r|s,\mathbf{q})$ of each sensor s and configuration $\mathbf{q}$ based on the new-coming sensor readings r^i. For this, and following [4], we maximize the expected log-likelihood of the sensor readings, where the expectation is taken with respect to the probability masses at each $\mathbf{q}$, i.e.,

$$E\{\log p(r|s)\} = \sum_{\mathbf{q}\in\mathcal{C}} P(\mathbf{q}) \log p(r|s,\mathbf{q}). \tag{22}$$

Maximizing each term in the above sum separately and reasoning as in section 2.1 yields iterative formulas for the ML estimates of the parameters of each density $p(r|s,\mathbf{q})$. The term $P(\mathbf{q})$ is independent of the parameters of the mixture densities $p(r|s,\mathbf{q})$ in (1) and thus drops out of the derivations and enters the iterative formulas (12)–(15) as an additional weight as

$$\mu_{sqj} := \mu_{sqj} + \frac{P\{j|r\}P(\mathbf{q})}{n\pi_{sqj}}(r - \mu_{sqj}), \tag{23}$$

$$\sigma^2_{sqj} := \sigma^2_{sqj} + \frac{P\{j|r\}P(\mathbf{q})}{n\pi_{sqj}}[(r - \mu_{sqj})^2 - \sigma^2_{sqj}], \tag{24}$$

$$\pi_{sqj} := \pi_{sqj} + \frac{P(\mathbf{q})}{n}(P\{j|r\} - \pi_{sqj}), \tag{25}$$

$$k_{sqj} := k_{sqj} + \frac{P\{j|r\}P(\mathbf{q})}{n\pi_{sqj}}\left[\left(\frac{r - \mu_{sqj}}{\sigma_{sqj}}\right)^4 - k_{sqj} - 3\right], \tag{26}$$

for a measurement r of sensor s and for all kernels j of $p(r|s,\mathbf{q})$ and all configurations $\mathbf{q} \in \mathcal{C}$, where each r is modified appropriately as in (20).

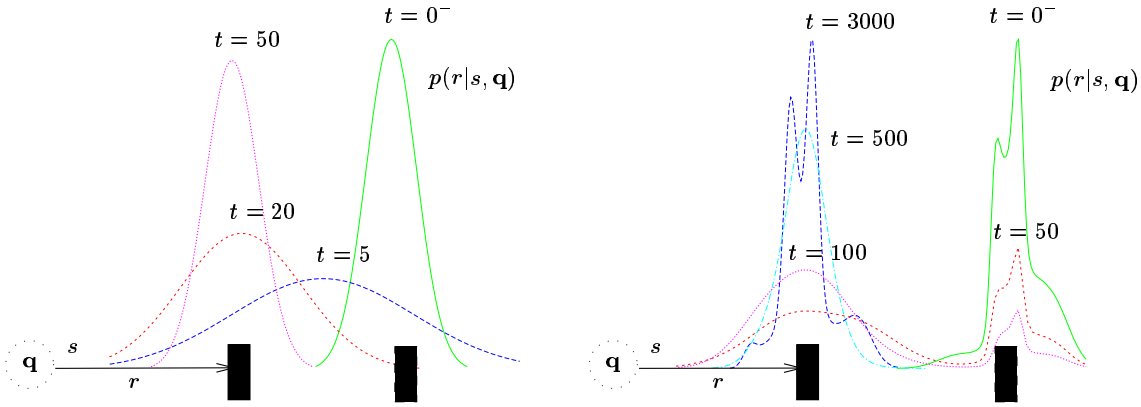
Based on the above procedure, a probabilistic map of the robot's environment can be built and maintained while the robot moves around and at the same time it can be used for robot localization and navigation. For each robot configuration $\mathbf{q}$ and each sensor s a mixture Gaussian model (1) is used. When the robot moves, the configuration probability distribution $P(\mathbf{q})$ is shifted by the traveled distance and blurred from (19), and it is subsequently updated from (21) based on the previously estimated density $p(r|s,\mathbf{q})$. Finally, from the sequential sensory information, and until the next motion command is issued, the density $p(r|s,\mathbf{q})$ of each sensor is iteratively updated using (23)–(26), while the statistical tests (16) and (17) decide on-line on the total number of kernels. The complete algorithm for map update and robot localization is shown in Fig. 2.

Initialization
An initial sensory probability map is assumed, e.g., created by using a CAD description of the environment
Localization and map update
Assume a current configuration probability distribution $P(\mathbf{q})$
While there is no motion command from the controller
  Map update
  For each sensor reading r *of a sensor* s *and*
  For each configuration $\mathbf{q}$ *in* $\mathcal{C}$
    For each kernel of $p(r|s,\mathbf{q})$
      update the parameters of the kernel using (23)–(26)
    update the number of kernels using (16) and (17)
When a motion command is issued that moves robot from $\mathbf{q}$ *to* $\mathbf{q}'$
  Configuration probability update
  Update from motion
  Shift $P(\mathbf{q})$ *by* $\mathbf{q}' - \mathbf{q}$*:*
    $P(\mathbf{q}_l + \mathbf{q}' - \mathbf{q}) := P(\mathbf{q}_l)$ *for all* $\mathbf{q} \in \mathcal{C}$
  Blur $P(\mathbf{q})$ *by motion noise* f*:*
    update $P(\mathbf{q}_k)$ *using (19) for all* $\mathbf{q}_k \in \mathcal{C}$.
  Update from sensing
  For each configuration $\mathbf{q}$ *in* $\mathcal{C}$
    compute from (20) the likelihood of the first signature
    update $P(\mathbf{q})$ *from (21)*
    eliminate those $\mathbf{q}_l$ *with* $P(\mathbf{q}_l)$ *close to 0*

Figure 2: The combined map update and robot localization algorithm.

The creation and deletion of kernels is controlled by the threshold parameters and the value of the constant n. However, n basically plays the role of a 'moving window' on previous sensor signals, and thus must be set according to the expected speed of change of the sensor distributions. Simulations revealed the expected trade-off between adaptability (small n) and precision (large n) in approximating the input distributions.

Assuming a sensor signature obtained by a ring of M proximity sensors around the robot and an average of K kernels for each mixture, the total number of parameters that have to be stored and processed at



a. one Gaussian and n=10 b. mixture of 10 Gaussians and n=100

Figure 3: Adaptability vs. precision in approximating an unknown sensor distribution.

any moment for each $\mathbf{q}$ are in the order of $4MK$. How large this number is depends on the precision we desire in sensing (large M) and on the precision in density approximation (large K).

In any case, however, a major speed-up can be achieved by exploiting the inherent parallelism of (23)–(26) and the independence of the M sensor readings. A distributed neural network implementation for the mixture density estimation problem is proposed in [10].

4 Results

In Fig. 3 we show a situation where a sensor s fires from a configuration $\mathbf{q}$, and we assume that at time $t = 0$ a new obstacle is positioned between the robot and the old obstacle, causing the sensor density to change. We have assumed that the initial density $p(r|s, \mathbf{q})$ is of the form shown in Fig. 1. In Fig. 3a we show the behavior of our algorithm when a single Gaussian kernel is used for approximating the input density and the constant $n = 10$. We note that in this case our algorithm adapts fast (after 50 sensor readings) to the new density.

In Fig. 3b we show the behavior of our algorithm with $n = 100$ and smaller split-join thresholds. In this case the initial density is better approximated but the adaptation is slower. The new density is very well approximated after 3000 steps, however after 500 steps the approximation can be considered adequate. This example reveals the trade-off between adaptability (small n) and precision in the approximation (large n).

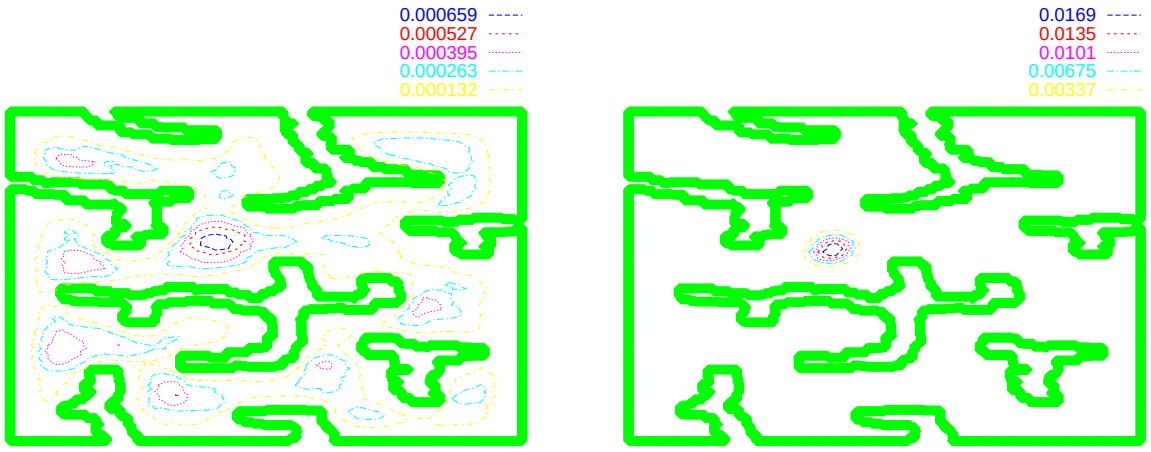
In Fig. 4 we show the behavior of the probabilistic localization algorithm when applied to a simulated mobile robot. Initially we assume that the robot's configuration density $P(\mathbf{q})$ is uniformly distributed over the free space and we show its updated values after 4 and 30 steps of random walk of the robot, respectively. In this example we assumed a static probabilistic map and simple Gaussian sensor models.

5 Conclusions-discussion

We have presented a method for building and maintaining sensory probabilistic maps that can be used for mobile robot navigation and localization in metric maps. The method is based on Maximum Likelihood estimation of the sensors densities at each robot configuration $\mathbf{q}$. A general Gaussian mixture density allows even non-Gaussian sensor noise to fit into the model, while sequential statistical tests make possible to model even dynamic maps, i.e., sensor distributions that are changing with time. Finally, we have proposed a combined algorithm for robot localization and map updating.

With the proposed mixture model the system is more robust to sensor noise than with other conventional techniques since new kernels of the mixture can potentially by employed for modelling outlier sensor measurements. The respective mixing weights are then continuously tuned so as to fade out the erroneous outliers. This property is particularly useful in dynamic environments where the sensor distributions change with time.

The major drawback of this method, a drawback prevalent in most Markov models described in the literature, is that a fine resolution is needed for cor-



a. contours of $P(\mathbf{q})$ after 4 steps

b. contours of $P(\mathbf{q})$ after 30 steps

Figure 4: Localization on a known probabilistic map: initially we assume uniform $P(\mathbf{q})$ over the total free space. The robot performs random walk.

rectly approximating unstructured or cluttered environments and which, considering that the total number of configurations in $\mathcal{C}$ could be large, requires a proportionally large number of parameters and consequently large processing time. However, due to the distributed character of our method we believe that parallel architectures can help reducing the processing time. Currently we are investigating this possibility and also try to fit our model in a general regression model over $\mathcal{C}$ that would significantly reduce the total number of parameters.

References

[1] W. Burgard, D. Fox, D. Hennig, and T. Schmidt. Estimating the absolute position of a mobile robot using position probability grids. In *Proc. 13th Nat. Conf. on Artificial Intelligence*, pages 896–901, Portland, Oregon, 1996.

[2] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *IEEE Computer Magazine, Special Issue on Autonomous Intelligent Machines*, June 1989.

[3] J. C. Latombe. *Robot Motion Planning*. Cluwer Academic Publishers, Boston, MA, 1991.

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

[5] A. Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, 3rd edition, 1991.

[6] R. Simmons and S. Koenig. Probabilistic navigation in partially observable environments. In *Proc. 14th Int. Joint Conf. on Artificial Intelligence*, pages 1080–1087, 1995.

[7] H. Takeda, C. Facchinetti, and J.-C. Latombe. Planning the motions of a mobile robot in a sensory uncertainty field. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16(10), Oct. 1994.

[8] S. Thrun, W. Burgard, and D. Fox. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning*, 31:29–53, 1998.

[9] D. M. Titterington, A. F. Smith, and U. E. Makov. *Statistical Analysis of Finite Mixture Distributions*. Wiley, 1985.

[10] N. A. Vlassis, G. Papakonstantinou, and P. Tsanakas. Mixture density estimation based on maximum likelihood and test statistics. *Neural Processing Letters*, 9(1), Feb. 1999, to appear.

[11] N. A. Vlassis and P. Tsanakas. A sensory uncertainty field model for unknown and non-stationary mobile robot environments. In *Proc. ICRA'98, IEEE Int. Conf. on Robotics and Automation*, Leuven, Belgium, May 1998.