To appear in *IEEE International Conference on Robotics and Automation* (ICRA '08), Pasadena, CA, May 2008.

Maximum Likelihood Estimation of Sensor and Action Model Functions on a Mobile Robot

Daniel Stronger and Peter Stone
Department of Computer Sciences
The University of Texas at Austin
{stronger,pstone}@cs.utexas.edu

***Abstract*— In order for a mobile robot to accurately interpret its sensations and predict the effects of its actions, it must have accurate models of its sensors and actuators. These models are typically tuned manually, a brittle and laborious process. Autonomous model learning is a promising alternative to manual calibration, but previous work has assumed the presence of an accurate action or sensor model in order to train the other model. This paper presents an adaptation of the Expectation-Maximization (EM) algorithm to enable a mobile robot to learn both its action and sensor model functions, starting without an accurate version of either. The resulting algorithm is validated experimentally both on a Sony Aibo ERS-7 robot and in simulation.**

I. Introduction

As an autonomous robot moves through its environment, it uses its sensors to gain information about the state of the world, and it takes actions to influence that state. In order to be effective, however, the robot must have accurate *models* of its sensors and actions, so that it can correctly interpret its raw observations and predict the changes that are caused by its actions. This paper considers the situation of a mobile robot in a fixed, known environment, where the robot's pose (location and orientation) is the relevant world state. In this context, its action model describes the robot's relative movements as a function of the action command being executed. The sensor model is a function from the robot's pose to a probabilistic distribution of its observations, such as from a laser range finder or a camera.

One way to construct a robot's action and sensor models is through manual calibration. However, this process can be laborious, inaccurate, and in particular, brittle, due to the propensity of the correct models to change over time from robot wear or environmental changes. This paper presents a method for a robot to autonomously learn its action and sensor models, starting without an accurate model for either. In this way, it contrasts from previous work that relies on the presence of either an accurate action or sensor model to learn the other model. Specifically, assuming the presence of a control policy that explores the full range of actions and states, the algorithm is able to learn an accurate action and sensor model for a mobile robot starting with no action model and a very poor sensor model. A robot's ability to learn accurate models starting with such little knowledge is a key factor in its overall autonomy and versatility.

The robot's action and sensor models are learned as probabilistic functions. For the sensor model, a polynomial function is learned that maps landmark distances to the means of the corresponding observation distributions. Additionally, two variance parameters are learned: one for those distance-based observations, and another for the observed horizontal angle to the landmark. For the action model, a table-based function is learned, with each of a set of 40 actions being mapped to the robot's corresponding actual velocities (forward, sideways, and turning). These results could additionally be interpolated for intermediate actions, as in [1], [2]. Notably, there are many different possible sets of parameters that could be learned. The derivations of the action and sensor learning procedures in Section III are specific to this one, but the principles used could be applied to learning a wide range of parameter sets, including settings with a continuous range of actions.

The above functions and variances are learned within the frameworks of Kalman Filtering [3] and the Expectation-Maximization (EM) algorithm [4]. The adaptation of the E-step to the problem described above is achieved by an extended Kalman filter and smoother (EKFS) [5], described in Section II. However, the adaptation of the M-step in Section III is a contribution of this paper. The resulting technique has been empirically validated both on a Sony Aibo ERS-7 and in simulation. These results are described in Section IV. Section V discusses previous related work and Section VI discusses future work and concludes.

II. Background

A mobile robot can be modeled by a hidden Markov model (HMM) with a continuous state space. The method presented in this paper applies the EM algorithm to such an HMM to learn the robot's action and sensor models, which are parameterized functions describing a nonlinear dynamical system. This section presents notation (adapted from that used in [6] and [7]) for discussing HMMs in a continuous domain, the EM algorithm, and the EKFS.

At time steps 0 through T, the robot's state vector is its pose, $s_t = (x_t, y_t, \theta_t)^\top$. At time steps 1 through T, an observation vector o_t is recorded by the robot. The parameters of the HMM consist of transition probability distribution functions, $a(s_t, s_{t+1}) = \mathrm{p}(s_{t+1}|s_t, c(t))$, where $c(t)$ is the action command being executed at time t,[1] the emission probabilities, $b(s_t, o_t) = \mathrm{p}(o_t|s_t)$, and the initial state distribution: $\pi(s) = \mathrm{p}(s_0)$. We denote the complete sequence of observations, o_1 through o_T, as O, and the complete set of HMM parameters as $\lambda = (a, b, \pi)$. Since

[1]The dependence of a on $c(t)$ is omitted for the sake of brevity.

the HMM represents the robot's state over time, the function a can be thought of as a model of the robot's actions, while b is a model of its sensations. The goal of this work is to find the parameters λ with the maximum overall likelihood given the observations, namely $\mathrm{p}(O|\lambda)$. This maximization is achieved by the EM algorithm, where the hidden variables are the HMM states over time (the robot's pose), and the parameters being estimated are the HMM parameters, λ.

EM alternates between two steps, expectation (E) and maximization (M). In the E-step, a probability distribution over the hidden variables is determined assuming the current parameter estimate is correct. The M-step uses the hidden variable distribution computed in the E-step to compute a new set of parameter estimates. The new parameters are defined to be those that maximize the expected log likelihood of the observed data (in our case the robot's observations), where the expectation is taken over the probability distribution computed in the E-step. The adaptation of the M-step to learning the robot's action and sensor models, presented in Section III, is a primary contribution of this paper.

The goal of the E-step is to compute the *a posteriori* probability distribution of s at each time step, denoted as $\gamma_t(s_t)$, given O and λ. In this paper, all state distributions are approximated as multivariate Gaussians. In linear systems, the E-step then amounts to an optimal smoother such as the forward-backward smoother [8]. In the nonlinear case, taking a first-order approximation results in an EKFS.

The EKFS algorithm can be understood in terms of the distributions $\alpha_t(s_t) = \mathrm{p}(o_1, \ldots, o_t, s_t|\lambda)$ and $\beta_t(s_t) = \mathrm{p}(o_{t+1}, \ldots, o_T|s_t, \lambda)$. Bayes' Law yields: $\gamma_t(s_t) = \alpha_t(s_t)\beta_t(s_t)/\mathrm{p}(O|\lambda)$. The distributions α and β are represented as multivariate Gaussians, with means denoted by $\mu_{\alpha,t}$ and $\mu_{\beta,t}$ and covariance matrices denoted by $\Sigma_{\alpha,t}$ and $\Sigma_{\beta,t}$. This factorization of γ_t is useful because α_t and β_t can both be computed through an extended Kalman filter as follows [9]. The values of $\mu_{\alpha,t}$ and $\Sigma_{\alpha,t}$ are computed by running the standard EKF forwards in time, starting with $\mu_{\alpha,0}$ and $\Sigma_{\alpha,0}$ equal to μ_π and Σ_π, the mean and covariance of the initial distribution $\pi(s)$. Similarly, the values of $\mu_{\beta,t}$ and Σ_{β_t} are computed by running the EKF *backwards in time*, starting with $\Sigma_{\beta,T} = \infty$. After α and β have been computed for each time step, $\mu_{\gamma,t}$ and $\Sigma_{\gamma,t}$ are given by:

$$\mu_{\gamma,t} = \mu_{\alpha,t} + \Sigma_{\alpha,t}(\Sigma_{\alpha,t} + \Sigma_{\beta,t})^{-1}(\mu_{\beta,t} - \mu_{\alpha,t}) \quad (1)$$

$$\Sigma_{\gamma,t} = (\Sigma_{\alpha,t}^{-1} + \Sigma_{\beta,t}^{-1})^{-1} \quad (2)$$

The computed means and variances of α_t, β_t, γ_t are used in the following section as the inputs to the M-step. Additionally, to learn an action model the joint distribution over two consecutive time steps is needed: $\mathrm{p}(s_t, s_{t+1}|O, \lambda)$. This distribution is discussed further in Section III-A.

Finally, it is also valuable to be able to compute $\mathrm{p}(O|\lambda)$, the overall likelihood of λ, which is what we are trying to maximize. This likelihood is the product of the likelihood of each observation, given all of the ones that preceded it. Each such observation likelihood can be computed from the state distribution given the preceding observations, namely α_{t-1}. Denoting the linearized sensation function as $o = Js + k$ gives us that $o_t \sim N(J\mu_{\alpha,t-1} + k, J\Sigma_{\alpha,t-1}J^\top)$. The likelihood of o_t according to this distribution can be multiplied into a running overall parameter likelihood as the successive values of α_t are computed.

III. The M-Step

We wish to determine the value of λ that maximizes the expected log likelihood of the observation sequence. Writing the total sequence of states as $S = (s_0, \ldots, s_T)$ and using $\hat{\mathrm{p}}$ to refer to probabilities according to the E-step distribution, this expected log likelihood is given by:

$$\begin{aligned}
&\mathrm{E}_{\hat{\mathrm{p}}}[\log \mathrm{p}(O|\lambda)] \\
&= \int_S \hat{\mathrm{p}}(S) \log \mathrm{p}(S, O|\lambda)\, dS \\
&= \int_S \hat{\mathrm{p}}(S) \log \left[\pi(s_0) \prod_{t=1}^{T} \mathrm{p}(s_t|s_{t-1}, \lambda)\mathrm{p}(o_t|s_t, \lambda)\right] dS \\
&= \int_S \hat{\mathrm{p}}(S) \left[\log \pi(s_0) + \sum_{t=1}^{T} (\log \mathrm{p}(s_t|s_{t-1}, \lambda) + \log \mathrm{p}(o_t|s_t, \lambda))\right] dS \\
&= \int_S \hat{\mathrm{p}}(S) \log \pi(s_0)\, dS + \sum_{t=1}^{T} \int_S \hat{\mathrm{p}}(S) \log \mathrm{p}(s_t|s_{t-1}, a)\, dS \\
&\quad + \sum_{t=1}^{T} \int_S \hat{\mathrm{p}}(S) \log \mathrm{p}(o_t|s_t, b)\, dS \qquad (3)
\end{aligned}$$

This expression decomposes the expected log likelihood into a sum of three terms that are functions of the three components of λ: π, a, and b, respectively. Maximizing this expression with respect to λ consists of maximizing each term with respect to the corresponding component. The distribution π that maximizes the first term is equal to the distribution of s_0 according to $\hat{\mathrm{p}}$, namely $\gamma_0(s)$, as in [6]. Maximizing the other two terms corresponds to learning the action and sensor model, discussed in the next two sections.

A. Learning the Action Model

For a mobile robot, the action model is a function from action commands to combinations of forwards, sideways, and turning velocities. During each time step, these velocities cause a *relative displacement* between s_t and s_{t+1}. This displacement is given by $R(-\theta_t)(s_{t+1} - s_t)$, where R represents a counterclockwise rotation around the z-axis through a specified angle, putting the displacement in the reference frame of s_t. Additionally, the relative displacement is assumed to be normally distributed, so that $(s_{t+1}|s_t) \sim N(s_t + R(\theta_t)\mu_{c(t)}, R(\theta_t)\Sigma_{c(t)}R(\theta_t)^\top)$, where $\mu_{c(t)}$ and $\Sigma_{c(t)}$ are the mean and covariance of the relative displacement of the action taken at time t, $c(t)$. In this paper, a fixed, constant value is used for $\Sigma_{c(t)}$.[2] The action model function, a, is therefore determined by the mean relative displacement, $\mu_{c(t)}$, for each action $c(t)$. A closed form expression for these displacements is derived as follows.

The maximum likelihood action model a is the one that maximizes the second term in (3). For each action A, let $Q(A) = \{t : c(t) = A\}$, the set of time steps at which

[2]The value used for Σ corresponded to standard deviations of 10 mm in each direction and 0.1 radians, at each time step. In preliminary experiments, the algorithm was robust to these values being at least doubled or halved.

action A occurred. Then, the portion of the second term in (3) affected by action A is given by:

$$\begin{aligned}
&\sum_{t\in Q(A)} \int_S \hat{\mathrm{p}}(S) \log \mathrm{p}(s_t|s_{t-1}, a)\, dS \\
&= \sum_{t\in Q(A)} \int_{s_{t-1},s_t} \hat{\mathrm{p}}(s_{t-1}, s_t) \log \mathrm{p}(s_t|s_{t-1}, a)\, ds_{t-1} ds_t \\
&= \sum_{t\in Q(A)} \int_{s_{t-1},s_t} \hat{\mathrm{p}}(s_{t-1}, s_t) \log f_A(d(s_{t-1}, s_t))\, ds_{t-1} ds_t \\
&= \sum_{t\in Q(A)} \int_{s_{t-1},s_t} \hat{\mathrm{p}}(s_{t-1}, s_t)[C- \\
&\quad \frac{1}{2}(d(s_{t-1}, s_t) - \mu_A)^\top \Sigma_A^{-1}(d(s_{t-1}, s_t) - \mu_A)]\, ds_{t-1} ds_t
\end{aligned}$$

where $d(s_{t-1}, s_t)$ is the relative displacement, $R(-\theta_{t-1})(s_t - s_{t-1})$, f_A is the probability density function (pdf) of the normal distribution with mean μ_A and covariance Σ_A, and C is a constant with respect to μ_A. Completing the M-step requires finding the value of μ_A that maximizes this expression, μ_A^*. This value is the one that minimizes the sum of the weighted integral of $(d(s_{t-1}, s_t) - \mu_A)^\top \Sigma_A^{-1}(d(s_{t-1}, s_t) - \mu_A)$, where the weights are the $\hat{\mathrm{p}}(s_{t-1}, s_t)$ distributions. Taking the gradient with respect to μ_A and setting it equal to zero, we get that μ_A^* is the correspondingly weighted mean of $d(s_{t-1}, s_t)$:

$$\mu_A^* = \frac{\sum_{t\in Q(A)} \int_{s_{t-1},s_t} \hat{\mathrm{p}}(s_{t-1}, s_t) d(s_{t-1}, s_t)}{\sum_{t\in Q(A)} \int_{s_{t-1},s_t} \hat{\mathrm{p}}(s_{t-1}, s_t)} \tag{4}$$

$$= \frac{1}{|Q(A)|} \sum_{t\in Q(A)} \int_{s_{t-1},s_t} \hat{\mathrm{p}}(s_{t-1}, s_t) d(s_{t-1}, s_t) \tag{5}$$

The expression $\hat{\mathrm{p}}(s_{t-1}, s_t)$ represents the probability of going through the two states s_{t-1} and s_t at those times, according to the E-step distribution over S. To complete the computation of μ_A^*, note that the integral in (5) is the expected value of $d(s_{t-1}, s_t)$ with respect to $\hat{p}$.

In order to compute this expected value, we use the factorization [6] of $\hat{\mathrm{p}}(s_{t-1}, s_t|O, \hat{\lambda})$ as $\alpha_{t-1}(s_{t-1})\hat{b}(s_t, o_t)\beta_t(s_t)\hat{a}(s_{t-1}, s_t)/\hat{\mathrm{p}}(O|\hat{\lambda})$, where $\hat{\lambda} = (\hat{a}, \hat{b}, \hat{\pi})$ are the parameter values from the previous iteration of the EM algorithm. This factorization is useful because $\alpha_{t-1}(s_{t-1})$ is already known from the forwards sweep of the forward-backward smoother (see Section II), and $\hat{b}(s_t, o_t)\beta_t(s_t)$ is a normal distribution over s_t that was computed in the backwards sweep (in between the observation and time updates). We denote the mean and variance of $\hat{b}(s_t, o_t)\beta_t(s_t)$ as μ_δ and Σ_δ and store them for each t as they are computed.

Denoting the product $\alpha_{t-1}(s_{t-1})\hat{b}(s_t, o_t)\beta_t(s_t)$ as $f_{\alpha,\delta}(s_{t-1}, s_t)$, we get that $\hat{\mathrm{p}}(s_{t-1}, s_t) \propto f_{\alpha,\delta}(s_{t-1}, s_t)\hat{a}(s_{t-1}, s_t)$. When the function $f_{\alpha,\delta}$ is thought of as a probability distribution over the (6-dimensional) joint state (s_{t-1}, s_t), we denote its mean and variance as $\mu_{\alpha,\delta}$ and $\Sigma_{\alpha,\delta}$. These quantities can be expressed as:

$$\mu_{\alpha,\delta} = \begin{pmatrix} \mu_{\alpha,t-1} \\ \mu_{\delta,t} \end{pmatrix} \quad and \quad \Sigma_{\alpha,\delta} = \begin{pmatrix} \Sigma_{\alpha,t-1} & 0 \\ 0 & \Sigma_{\delta,t} \end{pmatrix} \tag{6}$$

Once we compute the expected value and variance of $d(s_{t-1}, s_t)$ with respect to $f_{\alpha,\delta}$, this distribution is multiplied by $\hat{a}(s_{t-1}, s_t)$, which is equal to $\hat{f}_A(d(s_{t-1}, s_t))$, where $\hat{f}_A$ is specified by the mean and variance of action A in the previous iteration of EM, $\hat{\mu}_A$ and $\hat{\Sigma}_A$. If the first-order approximation of d is $Ls_{t-1,t} + m$, then the mean and variance of $d(s_{t-1}, s_t)$ with respect to $\hat{f}_{\alpha,\delta}$ are $L\mu_{\alpha,\delta}+m$ and $L\Sigma_{\alpha,\delta}L^\top$ respectively. Multiplying this distribution by $\hat{f}_A$ and substituting the resulting mean into Equation 5 yields:[3]

$$\mu_A^* = \frac{1}{|Q(A)|} \sum_{t\in Q(A)} \hat{\mu}_A + \hat{\Sigma}_A(\hat{\Sigma}_A + L\Sigma_{\alpha,\delta}L^\top)^{-1}(L\mu_{\alpha,\delta} + m - \hat{\mu}_A)$$

Averaging over the $|Q(A)|$ relevant frames yields a value for μ_A^*, the new estimate for μ_A.

B. Learning a Sensor Model

The robot's observations correspond to sightings of the landmarks in the environment, which are assumed to be visually distinguishable. Each observation vector o_t has two components, $o_{1,t}$ and $o_{2,t}$. These are assumed to be generated by $o_{1,t} \sim N(f(\mathrm{dist}(s_t)), \sigma_1^2)$ and $o_{2,t} \sim N(\mathrm{ang}(s_t), \sigma_2^2)$, where the functions $\mathrm{dist}(s)$ and $\mathrm{ang}(s)$ represent the distance and horizontal angle respectively from a robot at state s to the landmark that is observed. The function f maps the actual landmark distance onto the mean of the distribution for o_1, while there is no such function for the angle; the mean of o_2 is the angle itself. The sensor model that is learned consists of the function f plus the variances σ_1^2 and σ_2^2.

To learn this sensor model, we maximize the third term in (3) with respect to overall transition distribution b:

$$\begin{aligned}
&\sum_{t=1}^{T} \int_S \hat{\mathrm{p}}(S) \log \mathrm{p}(o_t|s_t, b)\, dS \\
&= \sum_{t=1}^{T} \int_{s_t} \gamma_t(s_t) \log b(s_t, o_t)\, ds_t \\
&= \sum_{t=1}^{T} \int_{s_t} \gamma_t(s_t) \left[C - \frac{1}{2} \left(\log \sigma_1^2\sigma_2^2 + \left(\frac{f(\mathrm{dist}(s_t)) - o_{1,t}}{\sigma_1} \right)^2 \right.\right. \\
&\quad \left.\left. + \left(\frac{\mathrm{ang}(s_t) - o_{2,t}}{\sigma_2} \right)^2 \right) \right] ds_t
\end{aligned}$$

Maximizing the above expression is equivalent to minimizing both

$$\sum_{t=1}^{T} \int_{s_t} \gamma_t(s_t) \left(\log \sigma_1^2 + \left(\frac{f(\mathrm{dist}(s_t)) - o_{1,t}}{\sigma_1} \right)^2 \right) ds_t \tag{7}$$

$$\text{and} \quad \sum_{t=1}^{T} \int_{s_t} \gamma_t(s_t) \left(\log \sigma_2^2 + \left(\frac{\mathrm{ang}(s_t) - o_{2,t}}{\sigma_2} \right)^2 \right) ds_t \tag{8}$$

Minimizing the first expression with respect to f amounts to minimizing $\sum_t \int_{s_t} \gamma_t(s_t)(f(\mathrm{dist}(s_t)) - o_{1,t})^2$. The function f is represented as a polynomial and therefore is learned via weighted polynomial regression, where the γ distributions act as a weighting function. Instead of attempting to analytically compute the regression weighted by a series of γ_t's, we approximate this weighting by drawing a number

[3]The expression for the mean of the product of two normal distributions also appears in Equation 1.

of samples from each γ distribution, and fit the polynomial to the resulting $(\text{dist}(s), o_1)$ pairs.

The value of σ_1 that minimizes (7) can be determined by differentiating with respect to σ_1 and setting the result equal to 0. The result is $\sigma_1^2 = \sum_t \int_{s_t} \gamma_t(s_t)(\hat{f}(\text{dist}(s_t)) - o_{1,t})^2$, where $\hat{f}$ is the optimal value of f computed above (which does not depend on σ_1). Similarly, minimizing (8) with respect to σ_2 yields $\sigma_2^2 = \sum_t \int_{s_t} \gamma_t(s_t)(\text{ang}(s_t) - o_{2,t})^2$. In both cases, the integral is again approximated by sampling from the γ_t distributions. In the experiments reported in this paper, because of the large number of frames in the data set, only one sample was taken from each frame. The degree of the polynomial used for regression was three.

IV. Empirical Validation

The technique described in this paper was validated both on a physical mobile robot and in simulation. In both cases, accurate sensor and action models were learned, starting with no action model and a very poor sensor model. On the real robot, the learned translational velocities were not evaluated due to the difficulty in measuring the ground truth for these velocities. All of the other aspects of the action and sensor models were measured and compared to the learned models. In simulation, ground truth is known, and all components of the learned models were evaluated.

A. Real Robot Results

Experiments were performed on a Sony Aibo ERS-7. The robot is roughly 280 mm tall and 320 mm long. Its four legs each have three degrees of freedom, as does its neck. Its primary sensor is a color camera at the tip of its nose. The robot's field of operation, depicted in Figure 1a), measures 5.4m $\times$ 3.6m. The landmarks used in this experiment were four distinct cylindrical beacons in fixed, known locations.

The robot's action commands correspond to *attempted* velocities in the x, y and θ directions. These attempted velocities determine the robot's step sizes and directions. However, they are often significantly inaccurate because of inaccuracies in the robot's joint movement and its feet slipping against the ground. The action model learned by the robot maps these action commands onto actual velocity combinations. A set of 40 action commands was used, determined as follows. The action commands are specified by their x, y, and θ velocities (v_x, v_y, v_θ), where each velocity component is normalized to the range $[-1, 1]$.

The velocity combinations were all chosen to walk as fast as possible in a given direction (including turning as a component of the direction). Such combinations satisfy the equation $v_x^2 + v_y^2 + v_\theta^2 = 1$. The velocities are determined by the combination of this equation, an angular velocity $a \in \{-\frac{1}{2}, -\frac{1}{6}, 0, \frac{1}{6}, \frac{1}{2}\}$, and a direction $b \in \{0, \pm\frac{\pi}{4}, \pm\frac{\pi}{2}, \pm\frac{3\pi}{4}, \pi\}$. Specifically, $a = v_\theta$ and b is the angular direction of (v_x, v_y). For each of the 40 combinations of a and b, these constraints uniquely specify a set of attempted velocities (v_x, v_y, v_θ). This set of motions was designed to cover the range of possible motions, excluding ones that have a very high angular velocity, which all effectively just cause the robot to spin in place. This parameterization is based on the one used in [2].

As the robot walked, it scanned its head from side to side to see as many beacons as possible. The two components of each observation are derived from the robot's camera image. The first component is the height of the beacon in pixels in the image, as shown in Figure 1b). The other component is the robot's estimate of the landmark's horizontal angle from the center of its body. This angle depends on the robot's head pan angle and the position of the landmark in the image. To attenuate the effect of false positives in object recognition, outlier observations were pruned in the first iteration of EM by discarding observations that represented too large of an innovation in the forward Kalman filter. The robot's acting and sensing abilities were developed as part of a previous project, and they are described fully in a technical report [10].

a) b)

Fig. 1. a) The Sony Aibo ERS-7 in its field of operation. The landmarks used are the four distinct color-coded cylindrical beacons. b) A robot's-eye view of a beacon. Its height in the image in pixels is one component of each observation.

At each time step, the action command executed and any observation made was recorded. In some frames, no observation was made. When these frames were processed in the algorithm, no EKF observation updates are made, and those frames are omitted in the sensor model reestimation. The training run lasted for 15 minutes, with each of the 40 actions being executed roughly four times for five seconds at a time. As mentioned in the introduction, a control policy is needed that enables the robot to explore the full range of actions and states. To meet this constraint, after every five seconds a new action was selected randomly, with priorities being placed on staying on the field and on distributing the executed actions evenly. As a matter of convenience, a previously developed accurate localization module was used to help keep the robot on the field. This measure would not be necessary if a larger field were used.

The parameter estimation algorithm was run on the resulting data set until convergence, defined as follows. In an ideal setting, EM is guaranteed to converge with the overall likelihood increasing with every iteration. However, approximations in the algorithm cause the likelihood, $p(O|\lambda)$, to fluctuate after a time. The learning is considered finished when 50 iterations pass without a new highest likelihood, indicating that these fluctuations have started to overshadow the learning. On the data collected by the robot, the algorithm took 152 iterations to converge. The entire process took 15 minutes of data-collection plus about 10 minutes of offline processing on a 2.79GHz Pentium 4 processor.

The sensor model consists of a polynomial function from distances to beacon heights, variances for those beacon heights, and a variance for the observed landmarks' horizontal angles. For each quantity, a starting value was used that was very inaccurate and the learned value was compared to the measured actual value. The actual sensor model was measured as follows. The robot was placed at distances from the beacon every 100 mm from 1275 to 4175 mm, the range of distances at which the robot can recognize the beacon. At each distance, 100 observations were made, and the mean and variance of the beacon heights and angles were computed. The actual beacon height means are shown as the measured sensor model in Figure 2, along with the starting polynomial (a very poor linear model) and the learned model.

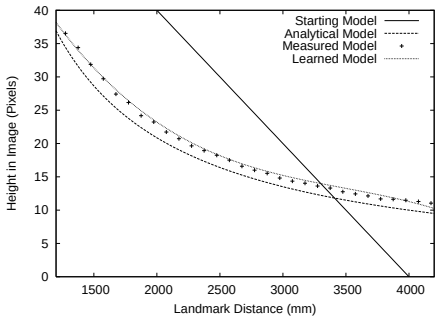

Fig. 2. The starting, learned, and measured sensor models. For comparison, an analytical sensor model is also shown, derived from the specifications of the camera. The learned model successfully approximates the measured one.

To obtain the measured beacon height and angle variances, the variances at each distance were averaged, each weighted by the frequency with which that beacon distance occurred during the learning. The starting, measured, and learned standard deviations for the beacon height were 10, 1.59, and 1.69 pixels respectively, an error of 6.3% in the learned value. The starting, measured, and learned angle standard deviations were 0.2, 0.0267, and 0.0116 radians, an error of a factor of 2.3. This error was likely due to the very small magnitude of the actual angle variances being dwarfed by the uncertainty in the robot's orientation at any time. This hypothesis is supported by the simulation results, in which a much larger angle variance was learned accurately.

For the action model, the starting velocity estimates were all zero. Rotational velocities were measured for each of the 40 actions by allowing the robot to execute that action for 30 seconds and measuring the total angular change. The RMS difference between the measured and learned angular velocities was 0.135 rad/s, a relative error of 3.2% of the measured range of angular velocities, 4.21 rad/s. By contrast, the original "attempted" angular velocities, which were manually calibrated (and not used by the robot), had an RMS error of 0.331 rad/s, a relative error of 7.9%.

B. Simulated Results

The above experiment was also run in the exact same way in simulation, using the same starting action and sensor models. The simulation engine models the robot's pose and observation vectors, but not its physical joint angles or camera image. This experiment verified that the method was able to learn accurate action and sensor models, including the translational velocities of the action model. The observations were computed by applying a simulated "actual" sensor model to the distances and adding gaussian noise to yield beacon height and angle observations. Random noise was additionally added to the robot's motion.

The algorithm converged on the simulated data after 959 iterations; the learning curve is shown in Figure 3a). Each action's learned velocities were compared to the ground truth. The final RMS errors in v_x, v_y, and v_θ were 23.06 mm/s, 18.34 mm/s, and 0.086 rad/s, relative errors of 3.2%, 2.2%, and 2.7%, respectively, with respect to the total range of velocities in those three directions, namely 710 mm/s, 840 mm/s and π rad/s. The x and y velocity RMS errors are shown decreasing over the course of the EM iterations in Figure 3b). Note that this RMS error improvement mirrors the overall log likelihood improvement shown in Figure 3a). This similarity provides confirmation that the log likelihood is a useful measure of the accuracy of the learned models.

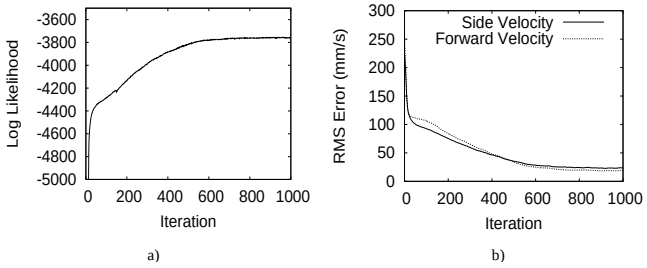

Fig. 3. a) The log likelihood improves over the course of 1000 iterations of EM. b) As the action model estimates converge, the average velocity errors decrease. The angular velocities (not pictured) converge within the first 100 iterations.

Additionally, the standard deviations of the Gaussian noise added to the two observation components in the simulator were 1 pixel for the beacon height and 0.5 radians for the observed horizontal angles. The learned values of these standard deviations were 0.980 pixels and 0.474 radians, errors of 2.0% and 5.3% respectively. The RMS error of the learned sensor model, over the range of observed distances, was 0.35 pixels, compared to an RMS error for the starting model of 12.71 pixels.

V. Related Work

This section situates the above approach to learning action and sensor model functions within the context of related previous work. First, work in developmental robotics aims to enable a robot to learn about its sensors and effectors, starting from as little innate knowledge as possible. For example, it is possible for the robot to start with out any knowlege about the structure of its own body [11], [12], or even the dimensionality of the outside world [13]. By contrast, the approach taken in this paper assumes that the robot has implicit innate knowledge about the structure of its body and state space. It instead aims to correlate the raw sensory input with the state of the world and learn how each of a set of actions effects that world state.

Learning a robot's action and sensor model functions in the context of the robot's unknown state over time is also a

form of dual estimation. Other approaches to dual estimation for a nonlinear dynamical system include the dual extended Kalman filter [14], the joint extended Kalman filter [15], [16], and discrimative training [17].

In this paper, the EM algorithm is used to learn a robot's action and sensor models. Ghahramani and Roweis discuss a number of advantages of using the EM algorithm for dual nonlinear estimation over the joint and dual EKF methods [18]. In particular, EM generalizes well to learning complex models or parameter combinations. This property makes it well-suited to learning the action and sensor model functions for a mobile robot.

When the EM algorithm is applied to dual estimation in a linear system, the E-step is an optimal smoother such as forward-backward smoothing [8]. The M-step yields new parameter settings that can be computed from summary statistics of the E-step distributions [19], [20]. In a nonlinear system, an EKFS can be used for the E-step, as in [18], [21], [22]. However, the methods used for the M-step vary as required by the domain. This paper contributes an adaptation of the M-step that enables learning a robot's action and sensor model functions. For the sensor model, sampling (a technique used in [21]) is combined with polynomial regression. For the action model, a closed form expression is derived for the mean relative displacements of each of a discrete set of actions.

Additionally, there is a wide range of previous work learning models of a robot's actions and sensors. A number of methods have been proposed for learning models of various sensors for a mobile robot [23], [24], [25], although these methods rely on the presence of an accurate action model. Conversely, a number of approaches have been taken to learning action models, assuming knowledge of an accurate sensor model, including [2], [26], [27], and adaptations of EM [28], [29], [30]. These applications of EM are specific to learning an action model.

There is little previous work that address the problem of learning an action and sensor model simultaneously. The authors have previously developed a bootstrapping method restricted to a one-dimensional domain [31], while Kaboli et al. present a Markov chain Monte Carlo method to learn four variance parameters of a probabilistic action and sensor model [32]. Compared to all of the above-mentioned work, the technique presented in this paper is unique in both the complexity of the model learned and the paucity of the knowledge with which the robot starts.

VI. Conclusion and Future Work

This paper introduces a technique that enables a mobile robot to simultaneously learn an accurate model of its action and sensors, starting with no action model and a very poor sensor model. The technique is an adaptation of the EM algorithm to a nonlinear dual estimation problem. The robot learns a table-based action model function, a polynomial sensor model function, and variances of the noise in the observation components. The technique presented in this paper is implemented and validated on data from a mobile robot traversing its environment. In both real robot and simulated experiments, the learned models match closely with the ground truth properties of the sensors and actions.

In this work, all of the probability distributions involved are multivariate Gaussians. One important area for future work is to extend the method to other distributions, such as combinations of Gaussians or histogram-based distributions. This enhancement would make the method more robust to multimodal state distributions. Another important area for future work is to combine the technique presented here with those used in SLAM, so that the robot can learn the layout of novel domains, as well as model its sensors and actions in them. Continued progress along these directions promises to greatly improve the autonomy and utility of mobile robots.

Acknowledgments

This research is supported in part by NSF CAREER award IIS-0237699 and ONR YIP award N00014-04-1-0545. The authors thank the UT Austin Villa team for developing the Aibo code base used in this work. Thanks also to Gregory Dudek, Ian Fasel, Todd Hester, Dana Ballard, Benjamin Kuipers, Michael Quinlan, and Nate Kohl for helpful discussions and feedback.

References

[1] M. Quinlan, C. Murch, T. Moore, R. Middleton, L. Li, R. King, and S. Chalup, "The 2004 NUbots team report," 2004, http://robots.newcastle.edu.au/publications/NUbotFinalReport2004.pdf.

[2] U. Duffert and J. Hoffman, "Reliable and precise gait modeling for a quadruped robot," in *RoboCup Symposium*, 2005.

[3] R. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, pp. 35–45, 1960.

[4] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society*, vol. 39, no. 1, pp. 1–38, 1977.

[5] A. Sage and J. Melsa, *Estimation theory with applications to communications and control*. New York: McGraw-Hill, 1971.

[6] L. Rabiner and B. Juang, "An introduction to hidden markov models," *ASSP Magazine*, vol. 3, no. 1, pp. 4–16, 1986.

[7] G. Welch and G. Bishop, "An introduction to the kalman filter," University of North Carolina at Chapel Hill, Department of Computer Science, Tech. Rep. 95-041, 2004.

[8] D. Fraser and J. Potter, "The optimum linear smoother as a combination of two optimum linear filters," *IEEE Transactions on Automatic Control*, vol. 14, no. 4, pp. 387–390, August 1969.

[9] D. Simon, *Optimal state estimation*. John Wiley and Sons, Inc.

[10] P. Stone, P. Fidelman, N. Kohl, G. Kuhlmann, T. Mericli, M. Sridharan, and S. en Yu, "The UT Austin Villa 2006 RoboCup four-legged team," The University of Texas at Austin, Department of Computer Sciences, AI Laboratory, Tech. Rep. UT-AI-TR-06-337, December 2006.

[11] D. Pierce and B. Kuipers, "Map learning with uninterpreted sensors and effectors," *Artificial Intelligence*, vol. 92, pp. 169–229, 1997.

[12] L. Olsson, C. Nehaniv, and D. Polani, "From unknown sensors and actuators to actions grounded in sensorimotor perceptions," *Connection Science*, vol. 18, no. 2, 2006.

[13] D. Philipona, J. O'Regan, and J.-P. Nadal, "Is there something out there? Inferring space from sensorimotor dependencies," *Neural Computation*, vol. 15, no. 9, 2003.

[14] L. W. Nelson and E. Stear, "The simultaneous on-line estimation of parameters and states in linear systems," *IEEE Transactions on automatic Control*, vol. AC-12, pp. 438–442, 1967.

[15] H. Cox, "On the estimation of state variables and parameters for noisy dynamic systems," *IEEE Transactions on Auromatic Control*, vol. 9, pp. 5–12, 1964.

[16] L. Ljung, "Asymptotic behavior of the extended kalman filter as a parameter estimator for linear systems," *IEEE Transactions on Automatic Control*, vol. 24, pp. 36–50, 1979.

[17] P. Abbeel, A. Coates, M. Montemerlo, A. Ng, and S. Thrun, "Discriminative training of Kalman filters," in *Proceedings of Robotics: Science and Systems*, Cambridge, USA, June 2005.

[18] Z. Ghahramani and S. Roweis, "Learning nonlinear dynamical systems using an EM algorithm," in *Advances in Neural Information Processing Systems 11*, M. Kearns, S. Solla, and D. Cohn, Eds. Cambridge, MA: MIT Press, 1999.

[19] R. H. Shumway and D. S. Stoffer, "An approach to time series smoothing and forecasting using the EM algorithm," *Journal of time series analysis*, vol. 3, pp. 253–264, 1982.

[20] Z. Ghahramani and G. Hinton, "Parameter estimation for linear dynamical systems," University of Toronto Department of Computer Science, Tech. Rep. CRG-TR-96-2, 1996.

[21] T. Briegel and V. Tresp, "Fisher scoring and a mixture of modes approach for approximate inference and learning in nonlinear state space models," in *Advances in Neural Information Processing Systems 11*, M. Kearns, S. Solla, and D. Cohn, Eds. Cambridge, MA: MIT Press, 1999.

[22] M. N. J de Freitas and A. Gee, "Nonlinear state space estimation with neural networks and the em algorithm," Cambridge University Engineering Department, Tech. Rep., 1999.

[23] H. Moravec and M. Blackwell, "Learning sensor models for evidence grids," *CMU Robotics Institure 1991 Annual Research Review*, pp. 8–15, 1993.

[24] D. Hahnel, W. Burgard, D. Fox, K. Fishkin, and M. Philipose, "Mapping and localization with RFID technology," in *International Conference on Robotics and Automation (ICRA-04)*, 2004.

[25] J. Burlet, O. Aycar, and T. Fraichard, "Robust navigation using Markov models," in *International Conference on Intelligent Robots and Systems*, August 2005.

[26] T. D. Larsen, M. Bak, N. Andersen, and O. Ravn, "Location estimation for an autonomously guided vehicle using an augmented Kalman filter to autocalibrate the odometry," in *FUSION98 Spie Conference*, Las Vegas, NV, July 1998.

[27] A. Martinelli, N. Tomatis, A. Tapus, and R. Siegwart, "Simultaneous localization and odometry calibration for mobile robot," in *Proceedings of the 2003 International Confreference on Intelligent Robots and Systems*, Las Vegas, NV, October 2003.

[28] N. Roy and S. Thrun, "Online self-calibration for mobile robots," in *Proceeding of the IEEE International Conference on Robotics and Automation*, vol. 3. Detroit, MI: IEEE Computer Society Press, May 1999, pp. 2292–2297.

[29] P. Beeson, A. Murarka, and B. Kuipers, "Adapting proposal distributions for accurate, efficient mobile robot localization," in *IEEE International Conference on Robotics and Automation (ICRA-06)*, 2006.

[30] M. Fox, M. Ghallab, G. Infantes, and D. Long, "Robot introspection through learned Hidden Markov Models," *Artificial Intelligence*, vol. 170, no. 2, pp. 59–113, 2006.

[31] D. Stronger and P. Stone, "Towards autonomous sensor and actuator model induction on a mobile robot," *Connection Science*, vol. 18, no. 2, pp. 1–23, 2006, special Issue on Developmental Robotics.

[32] A. Kaboli, M. Bowling, and P. Musilek, "Bayesian calibration for Monte Carlo localization," in *Twenty-First National Conference on Artificial Intelligence (AAAI)*, 2006, pp. 964–969.