# Simultaneous Learning of Motion and Sensor Model Parameters for Mobile Robots

Teddy N. Yap, Jr. and Christian R. Shelton
Department of Computer Science and Engineering
University of California, Riverside, CA 92521, USA
{tyap, cshelton}@cs.ucr.edu

*Abstract*— **Motion and sensor models are crucial components in current algorithms for mobile robot localization and mapping. These models are typically provided and hand-tuned by a human operator and are often derived from intensive and careful calibration experiments and the operator's knowledge and experience with the robot and its operating environment. In this paper, we demonstrate how the parameters of both the motion and sensor models can be automatically estimated during normal robot operations via machine learning methods thereby eliminating the necessity of manually tuning these models through a laborious calibration process. Results from real-world robotic experiments are presented that show the effectiveness of the estimation approach.**

## I. INTRODUCTION

With the current algorithms for mobile robot localization and mapping, robots can now successfully track their pose relative to their environment as well as build maps of their surroundings based from sensor data during navigation [1], [2], [3], [4], [5]. Crucial to these algorithms are the robot's motion and sensor models. The robot's motion model describes the effect the control input has on the state of the robot (e.g. its pose) while the sensor model relates the actual sensor measurements (e.g. sonar readings or laser scans) to the state of the robot or the environment. Since robot motion is not perfect and sensor measurements are almost always subject to random noise and error, the motion and sensor models are appropriately represented as probability distributions [6], [7], [8], [9].

The process of tuning the parameters of the motion and sensor models of a mobile robot is called *calibration*. Robot calibration is an important step in robotics since a calibrated robot performs better than an uncalibrated one. However, the parameters of the models are typically specified and manually tuned by a human operator and are often the results of intensive and careful calibration experiments as well as the operator's knowledge and experience with the robot and its operating environment. Since the models are influenced by the robot's characteristics and environmental properties, they are manually re-calibrated whenever there is a significant change in the robot or the environment.

In this paper, we demonstrate how the parameters of both the motion and sensor models of a mobile robot can be automatically estimated during normal operation via machine learning methods. We assume that the robot has access to the map of the environment as well as the historical account of its motion (i.e. odometry information) and perception (i.e. sonar readings). We propose using the Expectation Maximization (EM) [10] approach to estimate the motion and sensor model parameters of the robot. To obtain the likely trajectory of the robot through the environment, we perform particle filtering and smoothing [11], [12] using the actual sensor data obtained by the robot during its normal operation. We then compute the maximum likelihood estimates of the parameters given the estimated robot trajectory and actual data.

The contributions of this paper are as follows. We demonstrate how the parameters of a mobile robot can be automatically estimated from data, thus eliminating the need for manual tuning. As opposed to calibrating a robot through specific calibration experiments, our estimation approach does not require any special calibration setup and can be performed by the robot as it operates with little or no human intervention and without disruption to its normal operation. Since the parameters are estimated from actual data, the robot can learn an appropriate set of parameter values rather than relying on preset parameter guesses. Unlike previous approaches that calibrate only the motion model parameters [13], [14], [7], [15], [16], we also aim for the estimation of the sensor model parameters. Results from some actual robotic experiments are presented that show the effectiveness of the estimation approach as well as the advantages of calibrating both the motion and sensor models.

## II. RELATED WORK

Calibration has been an active research area in mobile robotics. UMBmark [13] is a method for the quantitative measurement of systematic odometry errors in a mobile robot that involves performing a series of simple calibration experiments in which the robot traverses a square path in both clockwise and counter-clockwise directions and manually measuring the absolute position of the vehicle to compare with the robot's calculated position based on odometry. Roy and Thrun [16] proposed a statistical method for the online self-calibration of the odometry of mobile robots which eliminates the need for explicit measurements of actual robot motion by a human or some external device. Unlike UMBmark, the method of Roy and Thrun is automatic and calibrates a robot's odometers continuously during its everyday operation allowing the robot to adapt to changes that might occur over its lifetime. Eliazar and Parr [7] have a similar goal where they proposed a method that can start with a crude motion model and bootstrap itself towards a more refined motion model thus, allowing the robot to adapt to changing motion parameters. But unlike [16], Eliazar and Parr used a more general motion model which incorporates interdependence between motion terms including the influence of turns on lateral movement and vice-versa. Instead of dealing with systematic errors, they also estimated non-systematic errors through the variance in the different motion terms. Other notable work on the automatic calibration of robot odometry include the self-calibrating extended Kalman filter ($\text{EKF}_{\text{SC}}$) approach by Caltabiano, Muscato, and Russo [14] and the observable filter (OF) (used in conjunction with the augmented Kalman filter (AKF)) by Martinelli et al. [15]. Foxlin [17] introduced a general architectural framework that enables systems to simultaneously track themselves, construct a map of landmarks in the environment, and calibrate sensor intrinsic and extrinsic parameters. Recently, Stronger and Stone [18] presented a technique for the simultaneous calibration of action and sensor models (SCASM) on a mobile robot. However, the models used in [18] are not probabilistic and their method makes use of careful calibration setup and requires the robot to go through a training phase thereby disrupting the robot from its normal operation.

Our work is similar in spirit with that of [7] and [16] but instead of only estimating motion model parameters, we also aim for the estimation of sensor model parameters. Unlike [18], our models are expressed probabilistically and our estimation method can be performed by the robot during its normal operation thus skipping the need for a separate training phase. Our results demonstrate the advantage of co-calibrating both models.

## III. PROBABILISTIC MOTION AND SENSOR MODELS

In this section, we discuss the probabilistic motion and sensor models we employ for our mobile robot. Note that there are several ways one can describe the motion and sensor models of a mobile robot probabilistically. The specific probabilistic motion and sensor models presented here are chosen because they are found to work well with our current robot and should not be seen as the only models that go with our automated calibration technique. In fact, in this paper, we propose a general framework for estimating the parameters of a mobile robot that can be used for other motion and sensor models as well.

### A. Motion Model

The purpose of the motion model is to describe the effect the control input has on the robot's configuration (e.g. its pose). In this paper, we assume that the robot moves in a planar environment so that its configuration at any given time $t$ can be represented as a three-dimensional column vector $\mathbf{x}_t = (x_t, y_t, \theta_t)^T$, where $(x_t, y_t)^T$ is the robot's two-dimensional Cartesian coordinates and $\theta_t$ denotes the robot's heading or orientation. Since robot motion is not perfect and uncertain (i.e. the same control command will never produce the same effect on the robot's configuration), the motion model is expressed as a probability distribution of the form $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{c}_t)$ (also called the *state transition probability*), where $\mathbf{x}_{t-1}$ and $\mathbf{x}_t$ are the robot's states for two consecutive time steps $t-1$ and $t$, respectively, and $\mathbf{c}_t$ is the control command executed by the robot during the time interval $[t-1, t)$. The control command $\mathbf{c}_t$ is given by the pair $(d_t, r_t)^T$, where $d_t$ is the distance traveled by the robot and $r_t$ is the rotation made by the robot, as reported by the wheel encoders. As suggested by the probability distribution $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{c}_t)$, the next state of the robot depends stochastically on its previous state one time step earlier and the control input $\mathbf{c}_t$. Although not explicitly included as part of the conditioning variables in $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{c}_t)$, the map of the environment as well as the set of parameters that define the motion model also determine the robot's next state.

There are several ways one can describe the motion model. In [16], Roy and Thrun suggested the following.

$$\begin{aligned} x_t &= x_{t-1} + d_t \cos(\theta_{t-1} + r_t) \\ y_t &= y_{t-1} + d_t \sin(\theta_{t-1} + r_t) \\ \theta_t &= (\theta_{t-1} + r_t) \bmod 2\pi \ , \end{aligned}$$

with the assumption that the drive and turn commands are independent. A more complex motion model proposed by Eliazar and Parr [7] that can account for simultaneous turning and lateral movement decomposes the movement into two principal components

$$x_t = x_{t-1} + D_t \cos(\theta_{t-1} + \frac{T_t}{2}) + C_t \cos(\theta_{t-1} + \frac{T_t + \pi}{2}) \quad (1)$$

$$y_t = y_{t-1} + D_t \sin(\theta_{t-1} + \frac{T_t}{2}) + C_t \sin(\theta_{t-1} + \frac{T_t + \pi}{2}) \quad (2)$$

$$\theta_t = (\theta_{t-1} + T_t) \bmod 2\pi \ , \quad (3)$$

where $\theta_{t-1} + \frac{T_t}{2}$ is referred to as the major axis of movement, $\theta_{t-1} + \frac{T_t+\pi}{2}$ is the minor axis of movement (orthogonal to the major axis), and $C_t$ is an extra lateral translation term to account for the shift in the orthogonal direction to the major axis. In their motion model, the variables $D_t, T_t$, and $C_t$ are all independent and conditionally Gaussian given $d_t$ and $r_t$:

$$\begin{aligned} D_t &\sim \mathcal{N}(d_t\mu_{D_d} + r_t\mu_{D_r}, d_t^2\sigma_{D_d}^2 + r_t^2\sigma_{D_r}^2) \\ T_t &\sim \mathcal{N}(d_t\mu_{T_d} + r_t\mu_{T_r}, d_t^2\sigma_{T_d}^2 + r_t^2\sigma_{T_r}^2) \\ C_t &\sim \mathcal{N}(d_t\mu_{C_d} + r_t\mu_{C_r}, d_t^2\sigma_{C_d}^2 + r_t^2\sigma_{C_r}^2) \ , \end{aligned}$$

where $\mathcal{N}(a, b)$ is a Gaussian distribution with mean $a$ and variance $b$, $\mu_{A_b}$ is the coefficient for the contribution of the odometry term $b$ to the mean of the distribution over $A$. Thus, the 12 parameters $\mu_{D_d}, \mu_{T_d}, \mu_{C_d}, \mu_{D_r}, \mu_{T_r}, \mu_{C_r}, \sigma_{D_d}^2, \sigma_{T_d}^2, \sigma_{C_d}^2, \sigma_{D_r}^2, \sigma_{T_r}^2$, and $\sigma_{C_r}^2$, define this motion model.

In this paper, we adopt the motion model of Eliazar and Parr but with a slightly different noise model. We still use (1), (2), and (3) for our state update equations except that

$$D_t \sim \mathcal{N}(d_t, d_t^2\sigma^2_{D_d} + r_t^2\sigma^2_{D_r} + \sigma^2_{D_1})$$
$$T_t \sim \mathcal{N}(r_t, d_t^2\sigma^2_{T_d} + r_t^2\sigma^2_{T_r} + \sigma^2_{T_1})$$
$$C_t \sim \mathcal{N}(0, d_t^2\sigma^2_{C_d} + r_t^2\sigma^2_{C_r} + \sigma^2_{C_1}) \ .$$

Our noise model is similar to the noise model of Eliazar and Parr with $\mu_{D_d} = 1$, $\mu_{D_r} = 0$, $\mu_{T_d} = 0$, $\mu_{T_r} = 1$, $\mu_{C_d} = 0$, $\mu_{C_r} = 0$. We added extra constant terms to the variances using additional parameters $\sigma^2_{D_1}$, $\sigma^2_{T_1}$, and $\sigma^2_{C_1}$. These parameters are added to account for errors that are not proportional to the translation or rotation of the robot. The motion parameters we wish to estimate from data are $\sigma^2_{D_d}, \sigma^2_{T_d}, \sigma^2_{C_d}, \sigma^2_{D_r}, \sigma^2_{T_r}, \sigma^2_{C_r}, \sigma^2_{D_1}, \sigma^2_{T_1}$, and $\sigma^2_{C_1}$.

## B. Sensor Model

Another crucial component of current algorithms for localization and mapping in mobile robotics is the sensor model. The sensor model, also called the *measurement or observation model*, describes the process by which sensor measurements are generated. Robot sensors are inherently noisy and uncertain, thus it is not uncommon to define the observation model as a conditional probability distribution $p(\mathbf{s}_t|\mathbf{x}_t)$, where $\mathbf{s}_t$ is the set of measurements received at time $t$ and $\mathbf{x}_t$ is the robot pose at time $t$. Just like in the motion model, even though we do not explicitly include the environment map as part of the conditioning variables in $p(\mathbf{s}_t|\mathbf{x}_t)$, it also determines the sensor measurement. Of course, the actual definition of the distribution $p(\mathbf{s}_t|\mathbf{x}_t)$ depends on the type of sensor used by the robot (e.g. cameras, range sensors). In this study, our mobile robot is equipped with a cyclic array of 16 ultrasound sensors with eight front sonars and eight back sonars. Thus, in our case, $\mathbf{s}_t = \{s_t^1, s_t^2, ..., s_t^K\}$, where $s_t^k$ is the $k$th sensor measurement received at time $t$ and $K = 16$ is the total number of sensor measurements. We assume that the errors in the sensor measurements are independent.

The distance reported by the range finder is often subject to random noise and error. In this paper, we make use of the sensor model for range finders described in [19] that represents the distribution as a mixture of four distributions corresponding to the four types of measurement errors typically observed in range readings. The four types of measurement errors are small measurement noise, errors due to unexpected objects or obstacles, errors due to failure to detect objects, and random unexplained noise. We let $s_t^{k*}$ denote the true distance to an obstacle, $s_t^k$ denote the recorded measurement, and $s_{\text{max}}$ denote the maximum possible reading (e.g. 5000mm).

In order to model small measurement noise associated with range readings, we define a narrow Gaussian distribution $p_{\text{hit}}$ over the range $[0, s_{\text{max}}]$ with mean $s_t^{k*}$ and standard deviation $\sigma_{\text{hit}}$. $\sigma_{\text{hit}}$ is an intrinsic parameter of the distribution $p_{\text{hit}}$. Formally

$$p_{\text{hit}}(s_t^k|\mathbf{x}_t) = \begin{cases} \eta \frac{1}{\sigma_{\text{hit}}\sqrt{2\pi}} e^{-\frac{(s_t^k - s_t^{k*})^2}{2\sigma^2_{\text{hit}}}} & \text{if } 0 \le s_t^k \le s_{\text{max}} \\ 0 & \text{otherwise} \end{cases} , \tag{4}$$

where $\eta$ is a normalizing factor due to clipping.

Although we assume that the map of the environment is static and does not include moving objects such as people, actual robot environments (such as buildings and hallways) are highly non-static and are often populated with dynamic entities other than the robot itself. These dynamic entities often block the robot's range sensors' "line-of-sight" thus causing them to return measurements shorter than the true range. This particular type of measurement error is modeled by a truncated exponential distribution $p_{\text{short}}$ with parameter $\lambda_{\text{short}}$. Specifically,

$$p_{\text{short}}(s_t^k|\mathbf{x}_t) = \begin{cases} \eta\lambda_{\text{short}} e^{-\lambda_{\text{short}} s_t^k} & \text{if } 0 \le s_t^k \le s_t^{k*} \\ 0 & \text{otherwise} \end{cases} , \tag{5}$$

where $\eta = 1/(1 - e^{-\lambda_{\text{short}} s_t^{k*}})$.

Sometimes, range finders fail to detect obstacles. For sonar sensors, this type of error can happen due to specular reflections when the echo fails to return to the sonar. Thus, the obstacle appears invisible from the robot's perspective. Sonar sensors are typically programmed to return the maximum sensor range $s_{\text{max}}$ when this happens. This particular type of measurement error is modeled by a pseudo point-mass distribution $p_{\text{max}}$ centered at $s_{\text{max}}$:

$$p_{\text{max}}(s_t^k|\mathbf{x}_t) = \begin{cases} 1 & \text{if } s_t^k = s_{\text{max}} \\ 0 & \text{otherwise} \end{cases} . \tag{6}$$

Finally, range finders can return totally unexplainable measurements. This can be caused by interference or cross-talk between different sensors or incomplete knowledge about ranging technologies. This type of measurement error is modeled by a uniform distribution $p_{\text{rand}}$ over the entire measurement range:

$$p_{\text{rand}}(s_t^k|\mathbf{x}_t) = \begin{cases} \frac{1}{s_{\text{max}}} & \text{if } 0 \le s_t^k \le s_{\text{max}} \\ 0 & \text{otherwise} \end{cases} . \tag{7}$$

The four distributions defined in (4) – (7) are combined by a weighted average through the mixing parameters $\alpha_{\text{hit}}, \alpha_{\text{short}}, \alpha_{\text{max}}$, and $\alpha_{\text{rand}}$

$$p(s_t^k|\mathbf{x}_t) = \alpha_{\text{hit}} p_{\text{hit}}(s_t^k|\mathbf{x}_t) + \alpha_{\text{short}} p_{\text{short}}(s_t^k|\mathbf{x}_t) + \alpha_{\text{max}} p_{\text{max}}(s_t^k|\mathbf{x}_t) + \alpha_{\text{rand}} p_{\text{rand}}(s_t^k|\mathbf{x}_t) \ ,$$

such that

$$\alpha_{\text{hit}} + \alpha_{\text{short}} + \alpha_{\text{max}} + \alpha_{\text{rand}} = 1 \ .$$

It is often appropriate to think of the mixing parameters as the a priori probabilities that a particular sensor reading is caused by one of the four types of measurement errors discussed above. Note that the mixing parameters and $\sigma_{\text{hit}}$ and $\lambda_{\text{short}}$ are the intrinsic parameters of this particular sensor model which we wish to learn from actual data.

## IV. PARTICLE FILTERING AND SMOOTHING

Fig. 1 illustrates the stochastic evolution of the state of the robot and its environment as defined by the state transition probability and measurement probability for three consecutive time steps, $t-1$, $t$, and $t+1$. In the figure, the

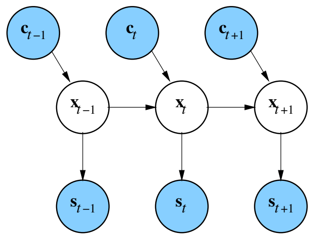

Fig. 1. The stochastic evolution of the robot's state, controls, and measurements.

robot pose $\mathbf{x}_t$ at time $t$ is stochastically dependent on its pose $\mathbf{x}_{t-1}$ at time $t-1$ and the control data $\mathbf{c}_t$, while the sensor data $\mathbf{s}_t$ at time $t$ is stochastically dependent on the robot pose $\mathbf{x}_t$ at time $t$. In the figure, shaded circles represent observable variables while unshaded circles represent unobservable or latent variables.

In order to estimate the motion and sensor parameters of the robot, we need to know its likely trajectory through the environment. As Fig. 1 shows, the actual trajectory of the robot is not directly observable. In this paper, we estimate the actual trajectory by performing particle filtering and smoothing using the observed control and sensor data. The following subsections provide a brief review of particle filtering and smoothing. Interested readers are referred to [20] and [11] for a more detailed treatment on particle filtering and smoothing, respectively.

### A. Particle Filtering

Filtering is the task of computing the belief state, that is, the posterior distribution over the current state of the system given all available data to-date. In mobile robot localization, filtering computes the probability $p(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t})$[1] that the robot is at position $\mathbf{x}_t$ given all control and sensor data from time 1 to time $t$ ($\mathbf{c}_{1:t}$ and $\mathbf{s}_{1:t}$, respectively) assuming that the control and sensor data arrive starting at time 1.

The most general framework for computing beliefs is given by the Bayes' filter. Bayes' filter recursively computes the posterior distribution over the state of a dynamical system conditioned on all available data to-date. Various implementations of the Bayes' filter exist (e.g. Kalman filter, extended Kalman filter, particle filter) with each implementation relying on different assumptions regarding the state transition and measurement probabilities and the initial belief.

Particle filters implement Bayes' filter by approximating the required distribution by a finite number of samples called *particles*, where each particle is a concrete state instantiation. At each time step $t$, upon receipt of the latest control data $\mathbf{c}_t$ and sensor data $\mathbf{s}_t$, a particle filter goes through the following stages in order to update the set of $N$ particles $\mathbf{X}_{t-1} = \{\mathbf{x}^1_{t-1}, \mathbf{x}^2_{t-1}, ..., \mathbf{x}^N_{t-1}\}$, representing the posterior distribution over the state one time step earlier, to the set of particles $\mathbf{X}_t = \{\mathbf{x}^1_t, \mathbf{x}^2_t, ..., \mathbf{x}^N_t\}$, representing the posterior distribution over the current state. The notation $\mathbf{x}^n_t$ denotes the $n$th particle in the particle set $\mathbf{X}_t$ at time $t$.

- *Particle Updating*. Update each particle $\mathbf{x}^n_{t-1}$ by sampling from the state transition probability $p(\mathbf{x}_t|\mathbf{x}^n_{t-1}, \mathbf{c}_t)$ to get the new particle $\mathbf{x}^n_t$.
- *Particle Weighting*. Weight each new particle $\mathbf{x}^n_t$ according to the measurement probability $p(\mathbf{s}_t|\mathbf{x}^n_t)$.
- *Particle Resampling*. Resample from the set of weighted particles with replacement with each particle having the probability of being selected proportional to its weight.

### B. Particle Smoothing

After performing particle filtering, we now have a particle representation of the posterior distribution over the state of the robot for each time step, given the observations prior to that time.

Smoothing is the task of computing the posterior distribution over a past state, given all evidence up to the present. Therefore, unlike filtering, smoothing provides a better estimate of the state of the system since it incorporates more (later) evidence.

In this paper, we rely on a simple and efficient technique presented in [12] for generating samples from the smoothing density $p(\mathbf{x}_{1:T}|\mathbf{c}_{1:T}, \mathbf{s}_{1:T})$ where $T$ denotes the last time step. The technique assumes that particle filtering has already been carried out on the entire dataset resulting in a particle approximation of the posterior distribution at each time step $t$ consisting of a weighted particle set.

The algorithm starts by drawing a particle at the last time step with probability proportional to its forward filtering weight. The algorithm then proceeds backward in time modifying the weights of the particles at each time step by multiplying their forward filtering weight by the probability that they lead to a transition to the drawn particle at the next time step. The algorithm then draws a particle with probability proportional to its modified weight at every time step. The sequence of particles $\widetilde{\mathbf{x}}_t$ drawn from time 1 to time $T$ ($1 \leq t \leq T$) constitutes a sampled robot trajectory $\widetilde{\mathbf{x}}_{1:T} \triangleq (\widetilde{\mathbf{x}}_1, \widetilde{\mathbf{x}}_2, ..., \widetilde{\mathbf{x}}_T)$ from the smoothing density $p(\mathbf{x}_{1:T}|\mathbf{c}_{1:T}, \mathbf{s}_{1:T})$.

## V. PARAMETER ESTIMATION FRAMEWORK

To estimate the set of motion and sensor model parameters of the robot, we propose using the Expectation Maximization (EM) algorithm [10]. EM is a standard machine learning method for finding the maximum likelihood estimates of parameters in probabilistic models where the models include unobservable or latent variables. It is an iterative optimization method for estimating unknown parameters given partial data.

In the expectation step or E-step, we perform particle filtering and smoothing and obtain sample robot trajectories from the smoothing density $p(\mathbf{x}_{1:T}|\mathbf{c}_{1:T}, \mathbf{s}_{1:T})$ as discussed in the previous section. In the maximization step or M-step, we treat the sampled trajectories obtained in the E-step as the ground truth to compute the maximum likelihood parameters of the models. Starting with some initial values of the parameters, we repeatedly alternate between performing the E-step and M-step until convergence. It has been shown [10] that the

[1]We use $\mathbf{z}_{m:n}$ to denote the sequence $\mathbf{z}_m, \mathbf{z}_{m+1}, ..., \mathbf{z}_n$.

EM algorithm is guaranteed to reach a local optimum. We should emphasize that our E-step is just an approximation of the "true" E-step of the EM algorithm since we are sampling from the posterior distribution $p(\mathbf{x}_{1:T}|\mathbf{c}_{1:T}, \mathbf{s}_{1:T})$ over possible robot paths instead of computing the exact expected robot trajectory to be used in the M-step.

To compute the maximum likelihood values of the motion model parameters in the M-step, we calculate the motion errors for $t = 1, 2, ..., T-1$ (based from the sampled robot trajectory $\widetilde{\mathbf{x}}_{1:T}$ obtained in the E-step) as well as the contributions of the odometry values $d_t$ and $r_t$ to the variances of these errors. We then maximize the likelihood function via conjugate gradient ascent with respect to the motion model parameters.

To compute the maximum likelihood values of the sensor model parameters in the M-step, we first calculate soft assignments of each individual sensor reading $s_i$ to the four components of our sensor model. That is, we calculate the probability $\rho_{i,\text{ccc}}$ that the sensor reading $s_i$ was generated by component ccc of our sensor model, where ccc $\in \{\text{hit}, \text{short}, \text{max}, \text{rand}\}$. That is

$$\begin{aligned} \rho_{i,\text{hit}} &= \eta p_{\text{hit}}(s_i|\mathbf{x}_i) \\ \rho_{i,\text{short}} &= \eta p_{\text{short}}(s_i|\mathbf{x}_i) \\ \rho_{i,\text{max}} &= \eta p_{\text{max}}(s_i|\mathbf{x}_i) \\ \rho_{i,\text{rand}} &= \eta p_{\text{rand}}(s_i|\mathbf{x}_i) \ , \end{aligned}$$

where $\mathbf{x}_i$ is the location where the sensor reading was taken and $\eta$ is a normalization constant to ensure that the above four values sum to 1. We then calculate the maximum likelihood values of the parameters (where $\mathbf{S}$ denotes the set of all sensor measurements $s_i$)

$$\begin{aligned} \alpha_{\text{hit}} &= \frac{\sum_i \rho_{i,\text{hit}}}{|\mathbf{S}|} \\ \alpha_{\text{short}} &= \frac{\sum_i \rho_{i,\text{short}}}{|\mathbf{S}|} \\ \alpha_{\text{max}} &= \frac{\sum_i \rho_{i,\text{max}}}{|\mathbf{S}|} \\ \alpha_{\text{rand}} &= \frac{\sum_i \rho_{i,\text{rand}}}{|\mathbf{S}|} \\ \sigma_{\text{hit}} &= \sqrt{\frac{\sum_i \rho_{i,\text{hit}}(s_i - s_i^*)^2}{\sum_i \rho_{i,\text{hit}}}} \\ \lambda_{\text{short}} &= \frac{\sum_i \rho_{i,\text{short}}}{\sum_i \rho_{i,\text{short}} s_i} \ , \end{aligned}$$

where $s_i^*$ is the true range of the object which can be easily computed by performing ray tracing on the map. See [19] for a detailed derivation of the above formulas. We should point out that the process of computing the maximum likelihood values of the sensor model parameters constitute an EM algorithm itself. In our experiments, we compute the maximum likelihood sensor parameters only once per iteration of our EM framework using the above formulas rather than having another EM algorithm nested within our EM framework. The derivation of the EM algorithm in [21] justifies this partial optimization step as well as the approximate E-step; our EM algorithm can still be viewed as optimizing the likelihood of the robot's trajectory. Fig. 2 shows the block diagram for the parameter estimation framework used in this paper.

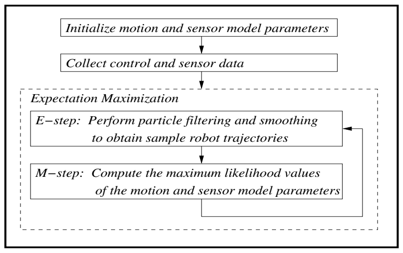
Initialize motion and sensor model parameters
Collect control and sensor data
Expectation Maximization
E−step: Perform particle filtering and smoothing to obtain sample robot trajectories
M−step: Compute the maximum likelihood values of the motion and sensor model parameters

Fig. 2. The block diagram for the parameter estimation framework.

## VI. EXPERIMENTAL RESULTS

To demonstrate the effectiveness of the estimation approach discussed in the previous section, we present the results of robotic experiments we conducted. In our experiments, we used an ActivMedia Robotics P3-DX as our testbed robot. The robot is equipped with a front sonar array with eight sensors, one on each side and six forward at 20$^o$ intervals. It also has a rear sonar array with eight sensors, one on each side and six rear at 20$^o$ intervals. It is controlled by an IBM ThinkPad X32 notebook computer.

In our experiments, we considered two test environments (see Fig. 3 and Fig. 4). The first test environment is a makeshift environment we set-up that represents a scaled-down version of a typical office environment. The associated map of the environment has an approximate size of 6.7m $\times$ 6.7m. The second test environment is a portion of the south wing of the third floor of our Computer Science building. Compared to the first test environment, the second test environment is significantly bigger with a map size of approximately 38m $\times$ 30m. The second test environment is more realistic than the first test environment in that it is dynamic and contains many unmodeled objects and obstacles (e.g. people walking around, trash bins). For both test environments, we instructed the robot to navigate through the environment by visiting predefined waypoints and returning to the starting position while collecting control and sensor data along the way. In our experiments, the parameter estimation routine is carried out offline on a different machine for computational reasons. Table I provides summary information about our experiments. Notice that the parameter estimation routine currently runs approximately 10 times slower than real-time (i.e. data collection time) but with further advances in processor speed-up and code optimization we can expect the parameter estimation routine to achieve real-time execution. Another way to speed-up the estimation process is to consider only portions of the collected data set instead of using the entire data set.

Table II shows the values of the parameters we obtained after performing our estimation method starting from some initial crude (uncalibrated) model parameters and using the historical account of the robot's motion and perception. As can be seen in Table II, the estimated values of the motion parameters for both test environments are similar except for the values of the parameters $\sigma^2_{T_1}$, $\sigma^2_{D_1}$, and $\sigma^2_{C_1}$. These are all related to the non-proportional variance in the robot's

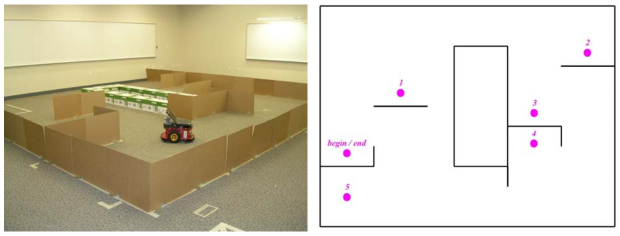

Fig. 3. The first test environment (left) and its associated map with waypoints (right).

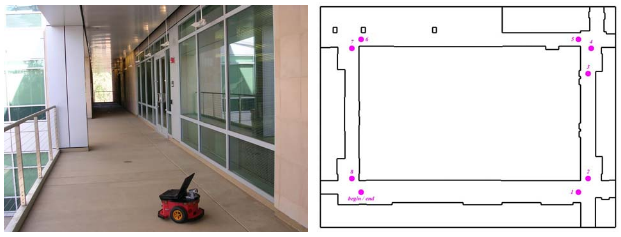

Fig. 4. The second test environment (left) and its associated map with waypoints (right).

translational and rotational motions. The drastic difference between the estimated values in the two environments is natural given the differences in floor material: carpet in the first test environment and concrete with expansion joints in the second test environment.

The differences in sensor model parameters are similarly explainable. Recall that the parameter $\alpha_{\text{short}}$ represents the probability that a particular sensor measurement is caused by unexpected objects in the environment such as people walking along the corridors and other unmodeled objects (e.g. trash bins) not included in the map. Since the second test environment is highly dynamic and contains a lot of unexpected objects, the learned value for parameter $\alpha_{\text{short}}$ is $3.4$ times larger than the value obtained from the first test environment. We also notice that the value of $\alpha_{\text{max}}$, the probability that the range finder would fail to detect obstacles, is higher by approximately $17\%$ in the second test environment than the value in the first test environment. The second test environment also contains smooth glass walls along the corridors that can cause specular reflections thus making the walls invisible to the sensor.

After obtaining the estimated parameters, we also tested their effect on the speed of robot navigation. In these experiments, we compared the times it took the robot to navigate through the same path in both environments when: A) using the uncalibrated motion and sensor models, B) using only the calibrated motion model, and C) using both calibrated motion and sensor models. Table III shows the navigation times for these experiments. As can be seen in Table III, using both calibrated motion and sensor models led to noticeable speed-ups in robot navigation.

Finally, to demonstrate the appropriateness of the estimated parameter values in estimating the robot path, we constructed sonar maps by plotting the endpoints of sonar readings with respect to the estimated path of the robot[2]. Fig. 5 shows the generated sonar maps displayed on top of the true map of the environment for easier visual inspection. It is easy to see that the sonar map generated using both calibrated motion and sensor models is better than those generated using the uncalibrated motion and sensor models or using only the calibrated motion model since the sonar readings are well-aligned with the actual map of the environment. For a more quantitative comparison, we generated the cumulative plot in Fig. 6 which shows the percentage of sonar endpoints that are closer than a given distance from the nearest wall in the environment. As Fig. 6 shows, over $50\%$ of the sonar endpoints are closer than $d = 50$mm to the nearest wall for the sonar map generated using both calibrated motion and sensor models while only about $33\%$ of the sonar endpoints are closer than $d = 50$mm to the nearest wall for the sonar

TABLE I
EXPERIMENTS SUMMARY

| | Test Environment 1 | Test Environment 2 |
|---|---|---|
| Map Sizes | $\approx 6.7\text{m} \times 6.7\text{m}$ | $\approx 38\text{m} \times 30\text{m}$ |
| Data Collection Times (1 round) | $\approx$ 5minutes | $\approx$ 13minutes |
| Total Time Step $T$ | 601 | $1,008$ |
| Number of Sonar Measurements | $9,616$ | $16,128$ |
| Parameter Estimation Times | 53m38s | 2h7m52s |

TABLE II
THE INITIAL AND ESTIMATED VALUES OF THE PARAMETERS.

| Initial Parameter Values | | Estimated Parameter Values | |
|---|---|---|---|
| | | Test Environment 1 | Test Environment 2 |
| $\sigma^2_{T_r}$ | 0.01 | 0.338267 | 0.348720 |
| $\sigma^2_{T_d}$ | 0.01 | 0.000345 | 0.000318 |
| $\sigma^2_{T_1}$ | 0.01 | **0.666048** | **0.007811** |
| $\sigma^2_{D_r}$ | 0.01 | 0.010731 | 0.007965 |
| $\sigma^2_{D_d}$ | 0.01 | 0.021869 | 0.044325 |
| $\sigma^2_{D_1}$ | 0.01 | **0.000001** | **0.010533** |
| $\sigma^2_{C_r}$ | 0.01 | 0.013427 | 0.026958 |
| $\sigma^2_{C_d}$ | 0.01 | 0.008588 | 0.005744 |
| $\sigma^2_{C_1}$ | 0.01 | **0.000014** | **0.001121** |
| $\alpha_{\text{hit}}$ | 0.30 | 0.434601 | 0.319870 |
| $\alpha_{\text{short}}$ | 0.20 | **0.029356** | **0.100010** |
| $\alpha_{\text{max}}$ | 0.30 | **0.348269** | **0.408525** |
| $\alpha_{\text{rand}}$ | 0.20 | 0.187774 | 0.171595 |
| $\sigma_{\text{hit}}$ | 500.00 | 31.18050 | 23.65530 |
| $\lambda_{\text{short}}$ | 0.15 | 0.001094 | 0.000651 |

TABLE III
THE NAVIGATION TIMES OF THE ROBOT.

| | Navigation Times | | |
|---|---|---|---|
| | A | B | C |
| Test Environment 1 (5 rounds) | 25m47s | 25m41s | **22**m**43**s |
| Test Environment 2 (2 rounds) | 26m19s | 25m58s | **25**m**3**s |

[2]We do not include the maximum sonar readings $s_{\text{max}}$ in the plot since they provide us with little information as to the location of the objects or obstacles.

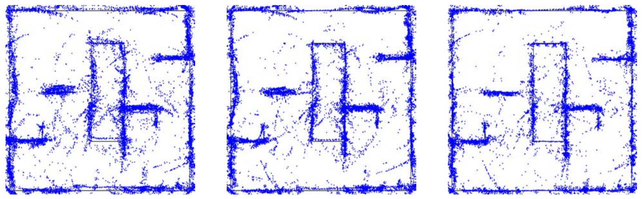
Fig. 5. The generated sonar maps using uncalibrated motion and sensor models (left), calibrated motion model only (middle), calibrated motion and sensor models (right). The (blue) dots represent the endpoints of sonar readings with respect to the estimated path of the robot. The true map is displayed for easier visual inspection.

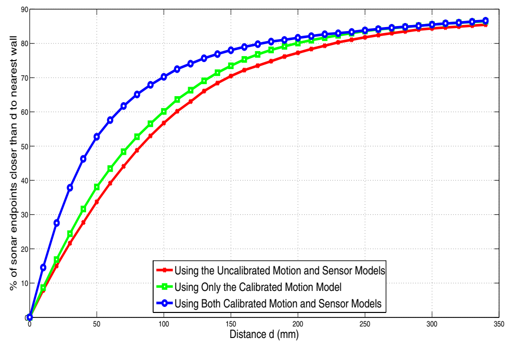

Fig. 6. Percentage of sonar endpoints that are less than a certain distance from the nearest wall in the environment.

map generated with uncalibrated motion and sensor models.

## VII. Conclusions and Future Work

Robot calibration is an important activity in mobile robotics. However, current calibration techniques often involve intensive and carefully set-up calibration experiments as well as significant amount of human effort. In this paper, we presented an automated technique for calibrating a mobile robot's motion and sensor models based from the control and sensor data obtained naturally during robot operations. Our method is based on a standard machine learning method called the EM algorithm. Starting from some initial parameter values, our method iteratively optimizes the parameters based from data collected during normal robot operation. Unlike other calibration techniques, our method does not require any special calibration setup and can be performed by the robot as it operates with little or no human intervention. Since the parameters are estimated from actual data, the robot can learn a more appropriate set of parameter values. The estimation procedure can be readily invoked by the robot at a later time when there is a need to re-calibrate the models. The results of some actual robotic experiments are presented that show the effectiveness of our approach. As pointed out, our parameter estimation framework is general in that it is not tied to any particular motion and sensor models and can be used for other motion and sensor models as well.

Although the results of the experiments are quite promising, we believe that there are several avenues for future work. One possible future extension would be to solve the problem of calibrating the robot's motion and sensor models without the benefit of an a priori map. In this case, the map of the environment has to be estimated from scratch together with the robot's motion and sensor models and at the same time being able to correctly localize a robot. We call this problem *Simultaneous Localization, Mapping, and Calibration* (or SLAM-C for short). This problem is relatively more difficult and complex and a solution to this problem would actually open up other several important and practical applications for mobile robots.

## References

[1] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (SLAM) problem," *IEEE J. Robot. Automat.*, vol. 17, no. 3, pp. 229–241, June 2001.

[2] A. Eliazar and R. Parr, "DP-SLAM: Fast, robust simultaneous localization and mapping without predetermined landmarks," in *Proc. 18th IJCAI'03*, Aug. 2003, pp. 1135–1142.

[3] A. I. Eliazar and R. Parr, "DP-SLAM 2.0," in *Proc. IEEE ICRA'04*, Apr. – May 2004, pp. 1314–1320.

[4] M. Montemerlo and S. Thrun, "Simultaneous localization and mapping with unknown data association using FastSLAM," in *Proc. IEEE ICRA'03*, vol. 2, Sept. 2003, pp. 1985–1991.

[5] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A factored solution to the simultaneous localization and mapping problem," in *Proc. 18th AAAI'02*, July – Aug. 2002, pp. 593–598.

[6] A. Elfes, "Sonar-based real-world mapping and navigation," *IEEE J. Robot. Automat.*, vol. RA-3, no. 3, pp. 249–265, June 1987.

[7] A. I. Eliazar and R. Parr, "Learning probabilistic motion models for mobile robots," in *Proc. 21st ICML'04*, July 2004, pp. 32–39.

[8] D. Fox, W. Burgard, and S. Thrun, "Markov localization for mobile robots in dynamic environments," *Journal of Artificial Intelligence Research*, vol. 11, pp. 391–427, Nov. 1999.

[9] S. Thrun, "Learning occupancy grid maps with forward sensor models," *Autonomous Robots*, vol. 15, no. 2, pp. 111–127, Sept. 2003.

[10] 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. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.

[11] A. Doucet, S. J. Godsill, and M. West, "Monte Carlo filtering and smoothing with application to time-varying spectral estimation," in *Proc. IEEE ICASSP'00*, vol. 2, June 2000, pp. 701–704.

[12] S. J. Godsill, A. Doucet, and M. West, "Monte Carlo smoothing for nonlinear time series," *Journal of the American Statistical Association*, vol. 99, no. 465, pp. 156–168, Mar. 2004.

[13] J. Borenstein and L. Feng, "UMBmark: A benchmark test for measuring odometry errors in mobile robots," in *Proc. SPIE Conf. on Mobile Robots*, Oct. 1995, pp. 113–124.

[14] D. Caltabiano, G. Muscato, and F. Russo, "Localization and self calibration of a robot for volcano exploration," in *Proc. IEEE ICRA'04*, Apr. – May 2004, pp. 586–591.

[15] A. Martinelli, N. Tomatis, A. Tapus, and R. Siegwart, "Simultaneous localization and odometry calibration for mobile robot," in *Proc. IEEE/RSJ IROS'03*, Oct. 2003, pp. 1499–1504.

[16] N. Roy and S. Thrun, "Online self-calibration for mobile robots," in *Proc. IEEE ICRA'99*, May 1999, pp. 2292–2297.

[17] E. M. Foxlin, "Generalized architecture for simultaneous localization, auto-calibration, and map-building," in *Proc. IEEE/RSJ IROS'02*, Oct. 2002, pp. 527–533.

[18] D. Stronger and P. Stone, "Simultaneous calibration of action and sensor models on a mobile robot," in *Proc. IEEE ICRA'05*, Apr. 2005, pp. 4563–4568.

[19] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA: MIT Press, 2005.

[20] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Trans. Signal Processing*, vol. 50, no. 2, pp. 174–188, Feb. 2002.

[21] R. Neal and G. E. Hinton, "A view of the EM algorithm that justifies incremental, sparse, and other variants," in *Learning in Graphical Models*, M. I. Jordan, Ed. MIT Press, 1999, pp. 355–368.