Bayesian Calibration for Monte Carlo Localization

Armita Kaboli* and **Michael Bowling†** and **Petr Musilek***
University of Alberta
Edmonton, Alberta, Canada
armita@ece.ualberta.ca, bowling@cs.ualberta.ca, musilek@ece.ualberta.ca

Abstract

Localization is a fundamental challenge for autonomous robotics. Although accurate and efficient techniques now exist for solving this problem, they require explicit probabilistic models of the robot's motion and sensors. These models are usually obtained from time-consuming and error-prone measurement or tedious manual tuning. In this paper we examine automatic calibration of sensor and motion models from a Bayesian perspective. We introduce an efficient MCMC procedure for sampling from the posterior distribution of the model parameters. We also present a novel extension of particle filters to make use of our posterior parameter samples. Finally, we demonstrate our approach both in simulation and on a physical robot. Our results demonstrate effective inference of model parameters as well as a paradoxical result that using posterior parameter samples can produce more accurate position estimates than the true parameters.

Introduction

Estimation of a mobile robot's position from sensor and odometry readings, or *localization*, is a prerequisite of most tasks for autonomous robots. Solutions to robot localization have become well established over the past decade. In situations where a unimodal distribution can adequately represent position uncertainty, the Kalman filter (Kalman 1960) and extensions (Julier & Uhlmann 2004), have proven effective. In situations requiring a multimodal representation, Monte Carlo localization (MCL) based on particle filtering (Fox *et al.* 1999; Thrun *et al.* 2000) is a robust solution.

All of these techniques for localization require probabilistic models of the robot's motion and sensors. The accuracy of the resulting position estimates naturally depends upon the accuracy of these models. Finding accurate models, or *calibration*, traditionally involves a combination of expert knowledge, manual measurement, and hand tuning. As a result, calibration of a robot's motion and sensor models can be tedious and error-prone. It becomes a more serious problem when one considers that a robot's physical properties (*e.g.*, tire inflation and motor wear) and the details of the environment (*e.g.*, surface friction, lighting, and obstacle reflectivity) are always changing. When faced with new sensors, locomotion mechanisms, or even environmental conditions, calibration can be a serious hurdle. For example, consider modelling a laser range-finder outdoors with falling snow, or a legged robot walking on a very soft surface for the first time.

This paper proposes a technique for automatic calibration of motion and sensor models for use with Monte Carlo localization. We take a Bayesian approach to the problem. First, we encode our expertise as a prior distribution over a parameterization of possible models. Then, after gathering data from the robot, we compute the posterior distribution over model parameters conditioned on this data. In particular, we describe an efficient Markov chain Monte Carlo (MCMC) procedure for sampling from this posterior distribution. We also introduce a novel extension to Monte Carlo localization that makes use of our posterior model samples.

The rest of the paper is organized as follows. First, we introduce Monte Carlo localization along with a brief summary of Markov chain Monte Carlo techniques. We then present our Bayesian calibration algorithm and describe how to use the posterior model samples in localization. We present results of our algorithm both in simulation and on a quadruped robot with vision as the primary sensor. Not only do we demonstrate effective calibration, but we also show paradoxical results that localization using the posterior can result in better accuracy than localization with the true parameters. Finally, we relate our approach to other work on automatic calibration, and then conclude.

Background

Monte Carlo localization (Fox *et al.* 1999) is basically the application of particle filtering to the problem of robot pose estimation. Let x_t be the vector of the robot's pose (*e.g.*, position and orientation) at time t. Let u_t be the robot's action to reach time t and z_t be the subsequent vector of sensor readings. Let $x_{1:t}$ be the sequence $x_1, \ldots, x_t$ (similarly for $u_{1:t}$ and $z_{1:t}$). Then, Monte Carlo localization is concerned with characterizing the distribution $\Pr(x_T|z_{1:T}, u_{1:T})$, based on a given motion and sensor model.

A common practical assumption of motion and sensor models are that they are Markovian, *i.e.*, future pose and current observation are conditionally independent of any pre-

*Department of Electrical and Computer Engineering.
†Department of Computing Science.

vious pose and observation given the present pose. Therefore, the motion model can be defined simply as the distribution $\Pr(x_t|x_{t-1}, u_t)$ and the sensor model as the distribution $\Pr(z_t|x_t)$. This also allows localization to be performed in an online fashion. $\Pr(x_{t+1}|z_{1:t+1}, u_{1:t+1})$ is computed from $\Pr(x_t|z_{1:t}, u_{1:t})$, u_{t+1}, and z_{t+1}.

Particle Filtering. Particle filters approximate the target distribution with a set of samples: $x_t^{(i)}\ \forall i = 1 \ldots n$. These samples are computed recursively using importance sampling. A sample is first generated from a candidate distribution based on the motion model. Its importance sampling weight is then computed using the sensor model.

$$\hat{x}_t^{(i)} \sim \Pr(x_t|x_{t-1}^{(i)}, u_t) \quad (1)$$

$$w_t^{(i)} = \Pr(z_t|\hat{x}_t^{(i)}) \quad (2)$$

In Equation 1 $x_{t-1}^{(i)}$ is assumed, by the recursive nature of particle filters, to be sampled according to $\Pr(x_{t-1}|z_{1:t-1}, u_{1:t-1})$. Therefore $\hat{x}_t^{(i)}$ is a sample of $\Pr(x_t|z_{1:t-1}, u_{1:t})$, which is not quite the target distribution but differs from it by a factor proportional to $\Pr(z_t|x_t)$. This is corrected by the importance sampling weight. Our new particles are then obtained by drawing n samples from $\hat{x}_t^{(i)}$ proportionally to $w_t^{(i)}$ with replacement.

Particle Smoothing. In some situations one wishes to sample a complete trajectory, *i.e.*, sample $\bar{x}_{1:T}$ according to $\Pr(x_{1:T}|z_{1:T}, u_{1:T})$. Particle smoothing is a method for drawing a sample trajectory by first performing particle filtering to compute $\hat{x}_{1:T}^{(i)}$ and $w_{1:T}^{(i)}$. The sample trajectory is then computed (backwards) recursively using importance sampling. Specifically, $\bar{x}_{t-1}$ is sampled from $\hat{x}_{t-1}^{(i)}$ with probability proportional to $w_{t-1}^{(i)}\Pr(\bar{x}_t|\hat{x}_{t-1}^{(i)}, u_t)$.

We can derive this procedure from the recursive distribution we are seeking to sample from.

$$\begin{aligned}
&\Pr(x_{t-1}|z_{1:T}, u_{1:T}, x_{t:T}) \\
&= \Pr(x_{t-1}|z_{1:t-1}, u_{1:t}, x_t) && (3)\\
&= \Pr(x_t|z_{1:t-1}, u_{1:t}, x_{t-1})\Pr(x_{t-1}|z_{1:t-1}, u_{1:t})/Z && (4)\\
&= \Pr(x_t|x_{t-1}, u_t)\Pr(x_{t-1}|z_{1:t-1}, u_{1:t-1})/Z && (5)\\
&= \Pr(x_t|x_{t-1}, u_t)\Pr(z_{t-1}|z_{1:t-2}, u_{1:t-1}, x_{t-1}) \\
&\quad \Pr(x_{t-1}|z_{1:t-2}, u_{1:t-1})/Z' && (6)\\
&= \Pr(x_t|x_{t-1}, u_t)\Pr(z_{t-1}|x_{t-1}) \\
&\quad \Pr(x_{t-1}|z_{1:t-2}, u_{1:t-1})/Z', && (7)
\end{aligned}$$

where Z and Z' are normalization constants, Equations 3, 5, and 7 follow from the conditional independence of the Markov assumption, and Equations 4 and 6 follows from Bayes' rule. Since $\hat{x}_{t-1}^{(i)}$ is sampled according to $\Pr(x_{t-1}|z_{1:t-2}, u_{1:t-1})$, the importance sampling correction terms $w_{t-1}^{(i)}\Pr(\bar{x}_t|\hat{x}_{t-1}^{(i)}, u_t)$ account for the remaining terms in 7.

Parameterized Models. We are interested in calibrating motion and sensor models and so we assume that we are given models that depend on some vector of parameters θ. Our motion model now takes the form $\Pr(x_t|x_{t-1}, u_t, \theta)$, and similarly our sensor model takes the form $\Pr(z_t|x_t, \theta)$. In the above description of particle filtering and particle smoothing, all of the probabilities now carry an implicit conditional on θ.

Markov Chain Monte Carlo Methods. As Markov chain Monte Carlo (MCMC) methods are an important foundation of our Bayesian calibration algorithm, we give a brief synopsis of the basic ideas. Neal's technical report (1993) should be consulted for a full tutorial. MCMC techniques seek to efficiently sample from probability distributions that are expensive or impossible to do so directly. They construct an ergodic Markov chain whose stationary distribution is the target distribution for sampling. This chain is then simulated for a sufficiently long period of time to be sure its distribution is near the stationary distribution.

In particular, consider sampling from the joint distribution $\Pr(q_1, \ldots, q_n)$, where the q_i's are not independent. Let $q^i = q_1^i, \ldots, q_n^i$ be the ith state in the Markov chain. Define,

$$\begin{aligned}
q^i_{j=(i \bmod n)} &\sim \Pr(q_j|q_1^{i-1}, \ldots, q_{j-1}^{i-1}, q_{j+1}^{i-1}, \ldots, q_n^{i-1}) \\
q^i_{j\neq(i \bmod n)} &= q_j^{i-1}
\end{aligned}$$

In other words, the $(i \bmod n)$th variable is sampled from the conditional distribution given the other variables, and so each variable is conditionally sampled in turn. This is known as Gibbs sampling. It is easy to observe that the joint distribution is the stationary distribution of this Markov chain, as each step of the chain either leaves a variable (and therefore its distribution) unchanged or samples from the desired conditional. As long as all the conditional probabilities are nonzero, then this chain is also ergodic, and therefore simulating the Markov chain can be used to approximately sample the joint distribution $\Pr(q_1, \ldots, q_n)$.

Gibbs sampling is ideal for sampling from difficult joint distributions when the conditional distribution can be sampled easily. In some situations this is still expensive or impossible. Metropolis sampling provides an alternative when the joint density can be computed efficiently. The idea is to generate a possible next state in the chain $\hat{q}^i$ from a candidate distribution that typically involves small Gaussian sampled perturbations to the $(i \bmod n)$th variable of q^{i-1}. The candidate is *accepted* and becomes q^i with probability equal to the acceptance function,

$$A(q^{i-1}, \hat{q}^i) = \min\left(1, \frac{\Pr(\hat{q}^i)}{\Pr(q^{i-1})}\right). \quad (8)$$

Otherwise it remains unchanged, *i.e.*, $q^i = q^{i-1}$. As long as the candidate distribution is non-zero everywhere, the chain can be shown to be ergodic with the stationary distribution equal to the joint distribution. We will make use of both Gibbs and Metropolis sampling in the next section.

Bayesian Calibration

In this section we describe our procedure for automatic calibration of motion and sensor models. In our Bayesian approach, we are given training data $\mathcal{D} = (\bar{u}_{1:T}, \bar{z}_{1:T})$ and a prior distribution over model parameters $\Pr(\theta)$. Our goal is to characterize the distribution, $\Pr(\theta|\mathcal{D})$. As this distribution does not have a simple closed form, we will employ Markov chain Monte Carlo techniques to sample from this posterior distribution.

We begin by reformulating the posterior distribution by marginalizing over the robot's unknown trajectory,

$$Pr(\theta|\mathcal{D}) = \int_{\bar{x}_{1:T}} Pr(\theta, \bar{x}_{1:T}|\mathcal{D}). \tag{9}$$

Our approach is to draw samples from the joint distribution,

$$(\theta^{(i)}, x^{(i)}_{1:T}) \sim Pr(\theta, \bar{x}_{1:T}|\mathcal{D}), \tag{10}$$

and as a result $\theta^{(i)}$ will be sampled according to the desired posterior distribution $\Pr(\theta|\mathcal{D})$. The joint distribution in Equation 10 can be sampled using the Gibbs method, where each variable is repeatedly sampled in turn, conditioned on the other variables. Gibbs' sampling requires that the conditional distributions,

$$\bar{x}_{1:T} \sim \Pr(\bar{x}_{1:T}|\theta, \mathcal{D}) \tag{11}$$

$$\theta \sim \Pr(\theta|\bar{x}_{1:T}, \mathcal{D}), \tag{12}$$

be easy to sample. Recall that particle smoothing, for particular model parameters, samples complete trajectories given sequences of observations and actions. This is precisely what is required in Equation 11. Unfortunately, Equation 12 is not so conveniently sampled.

The other approach is to use the Metropolis sampling method, which requires that the joint density be easily computed. We can manipulate the joint density into a product of the prior and motion and sensor model densities.

$$\begin{aligned}
&\Pr(\theta, \bar{x}_{1:T}|\mathcal{D}) \\
&= \Pr(\bar{x}_{1:T}, \bar{z}_{1:T}|\theta, \bar{u}_{1:T})\Pr(\theta)/Z && (13)\\
&= \frac{\Pr(\theta)}{Z}\prod_{t=1}^{T}\Pr(\bar{x}_t, \bar{z}_t|\bar{x}_{1:t-1}, \bar{z}_{1:t-1}, \bar{u}_{1:T}, \theta) && (14)\\
&= \frac{\Pr(\theta)}{Z}\prod_{t=1}^{T}\begin{pmatrix}\Pr(\bar{x}_t|\bar{x}_{1:t-1}, \bar{z}_{1:t-1}, \bar{u}_{1:T}, \theta)\\ \Pr(\bar{z}_t|\bar{x}_{1:t}, \bar{z}_{1:t-1}, \bar{u}_{1:T}, \theta)\end{pmatrix} && (15)\\
&= \frac{\Pr(\theta)}{Z}\prod_{t=1}^{T}\Pr(\bar{x}_t|\bar{x}_{t-1}, \bar{u}_t, \theta)\Pr(\bar{z}_t|\bar{x}_t, \theta), && (16)
\end{aligned}$$

where Z is a normalization constant, Equation 13 follows from Bayes' rule and the assumption that the robot's actions are independent of the model parameters, and Equation 16 follows from the conditional independence of the Markov assumption given the model parameters.

Our Algorithm. Since we can efficiently sample from one of the conditional distributions and efficiently evaluate the joint density, we employ a hybrid MCMC approach. In one step, we use particle smoothing to sample a complete trajectory as in the Gibbs method. In the second, we use a candidate and acceptance function based on the joint density as in the Metropolis method. Specifically, we generate a candidate by making a small Gaussian perturbation to a single dimension of the parameter vector. The candidate is accepted using the acceptance function from Equation 8. Combined with Equation 16 we get the acceptance probability,

$$A(\theta, \hat{\theta}) = \min\left(1, \frac{\Pr(\hat{\theta}, \bar{x}_{1:T}|\mathcal{D})}{\Pr(\theta, \bar{x}_{1:T}|\mathcal{D})}\right) = \min\left(1, \frac{\Pr(\hat{\theta})}{\Pr(\theta)}\prod_{t=1}^{T}\frac{\Pr(\bar{x}_t, |\bar{x}_{t-1}, \bar{u}_t, \hat{\theta})\Pr(\bar{z}_t|\bar{x}_t, \hat{\theta})}{\Pr(\bar{x}_t, |\bar{x}_{t-1}, \bar{u}_t, \theta)\Pr(\bar{z}_t|\bar{x}_t, \theta)}\right). \tag{17}$$

This process of accepting or rejecting a candidate perturbation is then repeated for each dimension of the parameter vector. The Markov chain is generated by simply alternating between these two steps.

The stationary distribution of this hybrid MCMC approach remains the joint distribution, since each individual step is known not to affect the stationary distribution (Neal 1993). As long as the conditional distributions and candidate perturbation distributions are everywhere non-zero then the chain is also ergodic, and therefore samples of the chain $(\theta^{(i)}, \bar{x}^{(i)}_{1:T})$ approximate samples from the target joint distribution. The full procedure is outlined in Table 1.

Table 1: Bayesian calibration algorithm. The parameter N is the number of steps of change after burn-in; r is the rate of including data, and m is the number of Metropolis iterations per Gibbs step.

1. Given training data $\mathcal{D}$ and parameters N, r, m.
2. Initialize Markov chain.
$$\theta^{(0)} \sim \Pr(\theta)$$
3. For $i \leftarrow 1$ up to $(N + \frac{T}{r})$:
   (a) Sample a trajectory.
   $$\ell \leftarrow \min(T, i \times r)$$
   $$x^{(i)} \leftarrow \text{PARTICLESMOOTHING}(\bar{z}_{1:\ell}, \bar{u}_{1:\ell}, \theta^{(i-1)})$$
   (b) Sample model parameters.
   $$\theta^{(i)} \leftarrow \theta^{(i-1)}$$
   Repeat m times: For $j \leftarrow 1$ up to d:
   i. Generate candidate.
   $$\delta_k \leftarrow \text{SAMPLENORMAL}(0, \sigma_k^2)$$
   $$\hat{\theta}_k \leftarrow \theta_k^{(i)} + \begin{cases}\delta_k & \text{if } k = j\\ 0 & \text{otherwise}\end{cases}$$
   ii. Accept candidate.
   $$p \leftarrow \text{SAMPLEUNIFORM}(0, 1)$$
   $$\theta^{(i)} \leftarrow \begin{cases}\hat{\theta} & \text{if } p < \min\left(1, \frac{\Pr(\hat{\theta}, \bar{x}_{1:\ell}|\mathcal{D}_{1:\ell})}{\Pr(\theta, \bar{x}_{1:\ell}|\mathcal{D}_{1:\ell})}\right)\\ \theta^{(i)} & \text{otherwise}\end{cases}$$
4. Return samples $(\theta^{(i)}, x^{(i)})$ for $i = (\frac{T}{r} + 1) \ldots (\frac{T}{r} + N)$.

Additional Details. There are two additional details that can speed convergence and improve robustness. First, notice that Step 3(a) is the costly operation in Table 1. It involves $2Tn$ evaluations of the motion and sensor models, where n is the number of particles used in particle filtering. Step 3(b) only requires $2Td$ evaluations of the motion and sensor models, where $d \ll n$ is the number of model parameters. We repeat Step 3(b) multiple times (no more than

n/d) before returning to Step 3(a). This makes the Metropolis sampling step closer to a true Gibbs step, and speeds convergence, while having little effect on running time.

Second, we don't use the entire training sequence when computing the acceptance probability in early steps of the chain. Instead, we only use the first $\min(T, r \times i)$ observations and actions of $\mathcal{D}$ in the ith step of the chain, where r is a parameter of the algorithm. The idea is that when a small amount of training data is added, the new posterior distribution of θ is not far from the posterior distribution without the additional data. By adding in training data slowly, the chain's current sample of θ (with less data) will be closely distributed to the new posterior (with more data). This will result in candidate parameter perturbations having higher acceptance probabilities and therefore faster convergence.

Posterior Parameter Samples

In the previous section we described a procedure for drawing samples of model parameters $\theta^{(i)}$ from the model posterior. What still remains is to show how to use the parameter samples with Monte Carlo localization. The simplest approach is to choose some single parameter vector based on the samples and use this when doing particle filtering. One straightforward method using this approach is to use the parameters' sample mean,

$$\theta_{\text{mean}} = \textstyle\sum_{i=1}^{N} \theta^{(i)}/N. \tag{18}$$

Another simple method is to choose the maximum a posteriori (MAP) sample from the Markov chain,

$$\theta_{\text{MAP}} = \operatorname*{argmax}_{i=\{1,\ldots,n\}} \Pr(\theta^{(i)}, x_{1:T}^{(i)} | \mathcal{D}), \tag{19}$$

using Equation 16. We present accuracy results using both methods in the next section. Neither, though, are properly Bayesian, making use of the uncertainty inherent in the posterior samples. The Bayesian approach is to construct posterior motion and sensor models. A fully Bayesian approach, though, would result in non-Markov models since the next state and observation would no longer be independent of history. As we are taking a batch approach to calibration, we only base our posterior models on the training data, ignoring new observations. Essentially, we are assuming that the model parameters are conditionally independent of new data given the training data (we call this the *batch assumption* in the derivation below.) As the size of the training data set increases this assumption becomes more and more justified.

Posterior Motion Model. We derive a posterior motion model by integrating over the unknown model parameters,

$$\begin{aligned}
&\Pr(x_t|x_{t-1}, u_t, \mathcal{D}) \\
&\quad = \int_\theta \Pr(x_t|x_{t-1}, u_t, \mathcal{D}, \theta)\Pr(\theta|x_{t-1}, u_t, \mathcal{D}) && (20)\\
&\quad = \int_\theta \Pr(x_t|x_{t-1}, u_t, \theta)\Pr(\theta|\mathcal{D}), && (21)
\end{aligned}$$

where Equation 21 follows from conditional independence of the training data given the model parameters and the batch assumption. We can sample from this distribution by simply sampling $\theta \sim \Pr(\theta|\mathcal{D})$ and then sampling $x_t \sim \Pr(x_t|x_{t-1}, u_t, \theta)$. Since $\theta^{(i)}$ are samples from the model parameter posterior, we can do approximate sampling by choosing $i \in \{1, \ldots N\}$ with uniform probability and then drawing the sample of x_t using the parameters $\theta^{(i)}$.

Posterior Sensor Model. For the posterior sensor model, we can again derive a posterior by integrating over the unknown model parameters,

$$\begin{aligned}
\Pr(z_t|x_t, \mathcal{D}) &= \int_\theta \Pr(z_t|x_t, \mathcal{D}, \theta)\Pr(\theta|x_t, \mathcal{D}) && (22)\\
&= \int_\theta \Pr(z_t|x_t, \theta)\Pr(\theta|\mathcal{D}), && (23)
\end{aligned}$$

using conditional independence and the batch assumption as was done with the posterior motion model. This can be approximated using a Monte Carlo estimate with samples of θ drawn from the posterior distribution. Since $\theta^{(i)}$ are samples from the posterior,

$$\Pr(z_t|x_t, \mathcal{D}) \approx \sum_{i=1}^{N} \frac{1}{N} \Pr(z_t|x_t, \theta^{(i)}). \tag{24}$$

We make one additional practical assumption. Rather than summing over all of the posterior model parameter samples, we will subsample $k < N$, and use the mean observation likelihood as our approximate posterior likelihood.

Summary. We have described three methods for incorporating the posterior model samples into Monte Carlo localization. The first two involve selecting a single vector of model parameters using either the mean of the samples or the maximum a posteriori sample. The third method approximates the full posterior models given the training data. We now examine the effectiveness of our Bayesian calibration procedure as well as the resulting localization accuracy of all three of these approaches.

Results

We applied our Bayesian calibration technique to two different robots: a simulated wheeled robot, and a real quadruped robot. In both scenarios, the calibration procedure was applied to two different amounts of training data, one small and one large, to examine the effect of the amount of training. The posterior samples from calibration were then used with MCL (with $n = 500$) using the three described methods. The resulting localization accuracy is compared to localization with both the mean of the prior (called "prior") and, in the case of the simulated robot, the true parameter values (called "true") as baselines.

Simulated Amigobot. Amigobot is a wheeled robot, equipped with eight sonar sensors, six on the front side and two on the back. We examined localization on a simulated Amigobot using all eight sonar sensors and odometry. The robot was given a map of an asymmetric "L"-shaped room 3×3 meters in size. The motion model consisted of independent zero-mean Gaussian noise added to the odometry's measurements of forward movement, rotation, and sideways

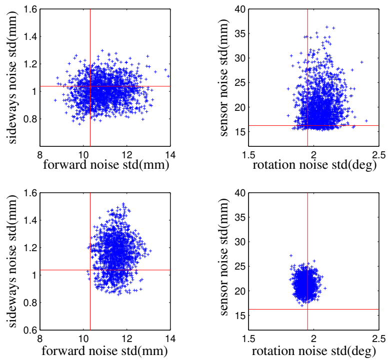

Figure 1: Amigobot posterior parameter samples with small (top) and large (bottom) amount of training. The lines show the true parameters used in the training data.

drift (always zero). The sensor model consisted of independent identically distributed zero-mean Gaussian noise added to the sonar's ideal reading. Thus, there are four parameters corresponding to the standard deviations of the model noise.

For calibration we constructed a prior distribution over the parameters using independent gamma distributions. The true parameter vector, by which the training and test data sets were generated, was then sampled from this distribution. Bayesian calibration was applied to both sets of short (400 seconds) and long training data (2000 seconds). The parameter samples returned from a sample run of calibration, along with lines corresponding to the true parameters, are shown in the scatter plots of Figure 1. One might expect that the posterior samples will focus around the true parameters, with tighter distributions with more training. The posterior is getting more concentrated with training, but not necessarily around the true parameters. What's more important, though, is localization accuracy.

The posterior parameter samples were used in MCL with all three methods: mean, MAP, and posterior, measuring the accuracy of the estimated position on 10 minutes of test data. The localization accuracy for all three methods as well as the true and prior baselines are shown in Figure 2. Accuracies were averaged over six runs of calibration and evaluated with thousands of test trajectories. Notice that the mean and MAP methods result in a significant improvement over the starting prior, and continue to improve with more data. In addition, they outperform localization using the true values! This paradox will be considered in more detail below, but this may suggest why the posterior parameters do not concentrate around the true parameters.

AIBO. The Sony AIBO ERS-7 is a four-legged robot with a CMOS color camera as the primary sensing device. This robot is used in the RoboCup Legged League. Our experiments used a similar environment, *i.e.*, bi-colored landmarks at the corners of a 2.7×1.8 meter field for localization. We extract two visual features for each visible landmark in the camera: the number of pixels and the relative angle to the landmark. The motion model is identical to the one used with Amigobot. The sensor model assumes independent Gaussian noise is added to the angle and the ratio of

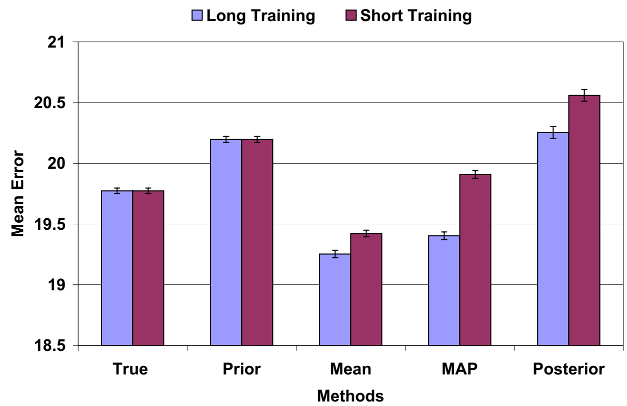

Figure 2: Amigobot localization results with 95% confidence intervals.

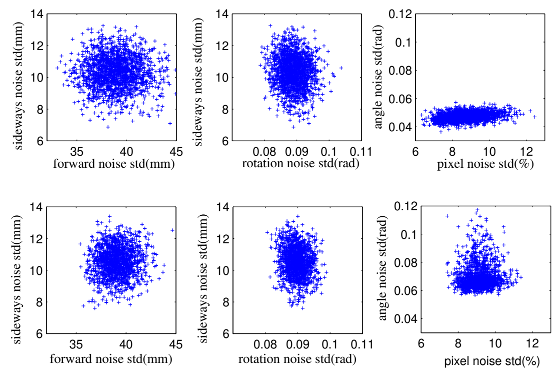

Figure 3: AIBO posterior parameter samples with small (top) and large (bottom) amount of training.

the number of observed pixels and the ideal number of pixels. Thus, there are five standard deviation parameters in the model, and we used independent gamma distributions for the priors based on our existing experience with the robot.

Like the Amigobot experiment, calibration is performed using a short (400 seconds) and long training set (1000 seconds). Figure 3 shows the posterior samples of a sample run of calibration on both sized training sets. Localization accuracy was then measured on a single sixteen minute test trajectory with ground truth gathered from an overhead camera with results shown in Figure 4. Localization over this same test trajectory was repeated and the 95% confidence intervals are shown. In this case, all three posterior sample methods outperform the mean of the prior. As before, we also see that error improves with more training data.

Outperforming the True Parameters. The results in Figure 2 seem at first glance impossible: if we use the true parameters then MCL should compute the true posterior, and error should be as low as possible. Recall, though, that particle filtering is only an approximation to the true posterior. With a finite number of particles, the inexactness of the approximation leaves room for improvement. Since our Bayesian calibration algorithm uses the same approximation when performing particle smoothing, it can actually compensate. In particular, the MCMC methods will favor param-

Figure 4: AIBO localization results with 95% confidence intervals.

eters that are more robust to the approximation as they will have higher posterior probability. This paradox deserves further exploration but it suggests that Bayesian calibration can find parameters well-suited to the number of particles.

Related Work

Although accurate motion and sensor models are key ingredients to accurate localization, the problem of calibration has not received much attention in the robotics literature. Roy and Thrun (1999) use a maximum likelihood estimator for odometry calibration. They identify bias in the robot's odometry online. The procedure, though, does not estimate a full probability distribution, instead assuming the odometry noise model is known. Eliazar and Parr (2004) go a step further, using an Expectation-Maximization (EM) approach to estimate a maximum likelihood motion model including its full distribution. Neither approach though deals with calibrating the sensor model. Stronger and Stone (2005) calibrate both the sensor and motion model, but as with Roy and Thrun their model is not probabilistic: only describing the mean instead of a full distribution. Consequently, it cannot readily be used in Monte Carlo localization.

In the more general state estimation community, calibration is more commonly referred to as model estimation. In model estimation, one traditional approach is to include model parameters in the state vector. State estimation will then, in theory, estimate both the posterior distribution of the system's state and the posterior distribution of the model parameters. This method performs very poorly in the context of particle filtering, as the only model parameters that are ever considered are those sampled at the start of the filter. One solution is to add artificial noisy dynamics to the model to avoid impoverishment (Liu & West 2000). Another solution is to use a particle's state trajectory as an implicit representation of the distribution of parameters (Storvik 2002). Sufficient statistics can be tracked for each particle to make this efficient, but it prevents its use with common models employed in robotics (*e.g.*, mixtures of distributions).

Andrews (2005) takes an approach very similar to ours, using MCMC to estimate the posterior distribution of model parameters conditioned on observed data. He focuses on the class of Nonlinear State-Space Models, which prevents its use with many robotics models (*e.g.*, bimodal noise distributions due to false positives.) In addition, he uses sets of trajectories and samples in his Markov chain, which can add to the computational cost.

Conclusion

In this paper, we presented a Bayesian approach to calibration for Monte Carlo localization. We described an efficient MCMC procedure for sampling from the posterior distribution of the model parameters conditioned on robot data. We also showed a novel method for using the posterior samples in localization. We demonstrated the effectiveness of our technique both in simulation and on a physical robot. We also presented the paradoxical result that using posterior samples in localization can result in better accuracy than the true parameters. In the future we hope to gain a better understanding of this counter-intuitive result. We also plan to apply this technique to more complex model spaces that involve a larger numbers of parameters.

References

Andrews, M. W. 2005. Bayesian learning in nonlinear state-space models. Unpublished manuscript.

Eliazar, A. I., and Parr, R. 2004. Learning probabilistic motion models for mobile robots. In *Twenty-First International Conference on Machine learning*.

Fox, D.; Burgard, W.; Dellaert, F.; and Thrun, S. 1999. Monte carlo localization: Efficient position estimation for mobile robots. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*. 343–349.

Julier, S. J., and Uhlmann, J. K. 2004. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE* 92(3):401–422.

Kalman, R. E. 1960. A new approach to linear filtering and prediction problems. *Journal of Basis Engineering* 82:35–45.

Liu, J., and West, M. 2000. Combined parameter and state estimation in simulation-based filtering. In Doucet, A.; Freitas, J. F. G. D.; and Gordon, N. J., eds., *Sequential Monte Carlo Methods in Practice. New York*. Springer-Verlag, New York.

Neal, R. 1993. Probabilistic inference using markov chain monte carlo methods. Technical Report CRG-TR-93-1, University of Toronto.

Roy, N., and Thrun, S. 1999. Online self-calibration for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2292–2297.

Storvik, G. 2002. Particle filters for state-space models with the presence of unknown static parameters. *IEEE Transactions on Signal Processing* 50:281–289.

Stronger, D., and Stone, P. 2005. Simultaneous calibration of action and sensor models on a mobile robot. In *IEEE International Conference on Robotics and Automation*.

Thrun, S.; Fox, D.; Burgard, W.; and Dellaert, F. 2000. Robust monte carlo localization for mobile robots. *Artificial Intelligence* 128(1-2):99–141.