Learning Probabilistic Motion Models for Mobile Robots

Austin I. Eliazar ELIAZAR@CS.DUKE.EDU
Ronald Parr PARR@CS.DUKE.EDU
Department of Computer Science, Duke University, Durham, NC, 27708 USA

Abstract

Machine learning methods are often applied to the problem of learning a map from a robot's sensor data, but they are rarely applied to the problem of learning a robot's motion model. The motion model, which can be influenced by robot idiosyncrasies and terrain properties, is a crucial aspect of current algorithms for Simultaneous Localization and Mapping (SLAM). In this paper we concentrate on generating the correct motion model for a robot by applying EM methods in conjunction with a current SLAM algorithm. In contrast to previous calibration approaches, we not only estimate the mean of the motion, but also the interdependencies between motion terms, and the variances in these terms. This can be used to provide a more focused proposal distribution to a particle filter used in a SLAM algorithm, which can reduce the resources needed for localization while decreasing the chance of losing track of the robot's position. We validate this approach by recovering a good motion model despite initialization with a poor one. Further experiments validate the generality of the learned model in similar circumstances.

1. Introduction

Advances in the areas of robot localization and Simultaneous Localization and Mapping (SLAM) have come a long way towards bringing the prospect of truly autonomous robot operation closer to reality (Thrun, 2002). With these techniques, mobile robots can create maps and position themselves in mapped environments with low risk of getting lost. However, an infrequently discussed but important input into these methods is the set of parameters for the robot's motion model. This model is provided to the robot by a human, based upon a combination of intuitions about the environment and experience with the robot. The model is usually refined in successive iterations (through intensive human effort) for improved performance.

Appearing in *Proceedings of the 21st International Conference on Machine Learning*, Banff, Canada, 2004.

In this paper, we present a novel method of automating the process of acquiring this motion model. This not only relieves some of the work needed from the robot operator, but can be extremely useful when the robot moves to a new type of terrain (e.g., moving from asphalt to gravel) or when the robot's behavior changes due to malfunctions or wear. In familiar territory, this method can provide a much more refined motion model, resulting in more accurate localization and mapping results with less computation.

The contributions of this paper can be summarized as follows. First, we develop a probabilistic motion model that is more sophisticated than previous learned motion models for mobile robots. Second, by applying the Expectation Maximization (EM) framework, we are able to learn the parameters of this model without the benefit of a known map. In effect, we estimate the motion model and map together as part of the same learning procedure. Finally, we validate our approach on several maps, demonstrating the ability to correct and refine inaccurate motion models.

2. Motion Models for Localization

Robot motion models play an important role in modern robotic algorithms. The main goal of a motion model is to capture the relationship between a control input to the robot and a change in the robot's configuration.[1] Good models will capture not only systematic errors, such as a tendency of the robot to drift left or right when directed to move forward, but will also capture the stochastic nature of the motion. The same control inputs will almost never produce the same results and the effects of robot actions are, therefore, best described as distributions (Thrun, 2000). These distributions play an important role in algorithms that use particle filters for localization and mapping. Specifically, they form the proposal distribution for the particle filter.

[1] In robotics, one typically treats the odometry as the control input since it is possible to specify control inputs to the robot in terms of target odometry.

A particle filter is a Monte Carlo method for estimating and propagating a probability distribution through a Markov model. We briefly review particle filters here, but refer the reader to excellent overviews of this topic (Doucet et al., 2001) and its application to robotics (Thrun, 2000) for a more complete discussion.

A particle filter maintains a weighted (and normalized) set of sampled states, $\mathbf{s} = \{s_1 \dots s_m\}$, called *particles*. At each step, upon observing evidence E, the particle filter:

1. Samples m new states $\mathbf{s}' = \{s'_1 \dots s'_m\}$ from from the weighted set of particles $\mathbf{s}$ with replacement.

2. Propagates each new state through a Markovian transition (or simulation) model: $P(s''|s')$. This entails sampling a new state from the conditional distribution over next states given the sampled previous state.

3. Weighs each new state according to a Markovian observation model: $P(E|s'')$

4. Normalizes the weights for the new set of states.

Particle filters are easy to implement and have been used to track multimodal distributions for many practical problems (Doucet et al., 2001). For robot localization, the distribution $P(s''|s')$ comes from the robot's motion model. (In full generality, the distribution from which next states are sampled is referred to as the *proposal distribution*, because it need not match $P(s''|s')$). The observation probabilities, $P(E|s'')$ come from a combination of the robot's sensor model, typically laser or sonar, and a map.

While the SLAM algorithm itself is not the focus of this paper, SLAM involves an extra step beyond that which is performed for localization. For SLAM, the map becomes part of the state that is estimated at each iteration. The particles therefore represent a joint distribution over maps and robot states (Cheeseman et al., 1990; Montemerlo & Thrun, 2002; Eliazar & Parr, 2004). The problem of efficiently maintaining this distribution is the source of much of the subtlety in SLAM algorithm research.

We used the DP-SLAM 2.0 algorithm (Eliazar & Parr, 2004) to construct our maps. DP-SLAM 2.0 is well suited to this problem because it is able to maintain a joint distribution over maps and robot positions by very efficiently maintaining large sets of particles. With a good motion model, DP-SLAM 2.0 is accurate enough to close large loops without any explicit map correction techniques. For these reasons, we chose to treat the set of particles maintained by DP-SLAM as a good representation of the probability distribution at any time step[2]. This lets us treat the SLAM algorithm as a black box that provides a distribution over robot positions during the map making process. The maps themselves are effectively marginalized out in the procedure for learning motion models. An alternative approach to this method could use a landmark based SLAM algorithm with a variant of a Kalman filter for state estimation.

In general, it is well known that a poor proposal distribution may require a prohibitively large number of particles to track the state of a system successfully. If the true behavior is not well within the sampling region of the particle filter, the probability of generating a particle consistent with the state of the system will be very low. In application, a robot localization procedure with a poor proposal distribution will require an excessive number of particles and yet may still lose track of the robot state. Thus, the motivation for acquiring a good motion model is quite strong.

Previous work in automatic acquisition of motion models for mobile robots has been fairly sparse. Most of the efforts have dealt with the problem of systematic errors, rather than the levels of noise that can be present. Borenstein and Feng (1994) describe a method for calibrating odometry to account for systematic errors. This method assumes a fairly smooth surface for calibration, with low non-systematic errors, and attempts to model each wheel independently. This method would become significantly more difficult for a robot with more than two drive wheels. Voyles and Khosla (1997) use shape from motion to learn the motion model parameters, but instead of using the shaft encoders, attempt to model the observation of applied force vectors directly. This would require an additional sensor which is not typically available on many robots, and has limited accuracy. Roy and Thrun (1999) propose a method which is more amenable to the problems of localization and SLAM. They treat the systematic errors in turning and movement as independent, and compute these errors for each time step by comparing the odometric readings with the position estimate given by a localization method. They can then use an exponential estimator to learn these two parameters online, assuming that short term localization results will be accurate enough to refine the motion model.

The goals of our approach are most similar to those of Roy and Thrun. We aim to have a method that can start with a crude model and bootstrap itself towards a more refined motion model, giving the robot the ability to adapt to changing motion parameters. Instead of merely learning two simple parameters for the motion model, as with the method proposed by Roy and Thrun, we seek to use a more general model which incorporates interdependence between motion terms, including the influence of turns on

[2] We note that for very bad initial models, this may not be a good assumption. In practice, we are able to recover from poor initial models and we discuss our unusually good performance in such cases in the conclusion.

lateral movement, and vice-versa. Furthermore, the proposed method extends the scope of the calibration beyond the systematic errors dealt with in previous methods. We believe that great gains in performance can be achieved by estimating the non-systematic errors, through variance in the different movement terms. This can be crucial to the motion model of SLAM methods, as different amounts of noise in the movement terms can produce vastly different proposal distributions (Burgard et al., 1999). A properly calibrated set of variance parameters will provide the localization algorithm with a more appropriate proposal distribution, allowing it to better focus its resources on the most likely poses for the robot.

The algorithm for learning the motion model is integrated with a SLAM algorithm, giving increased autonomy to the system. The robot now has the potential to learn the most appropriate model based upon recent experiences, and in direct conjunction with its current task. This is especially useful as the robot's motion model will change over time, both from changes in the terrain and from general wear on the robot. It is also important that this calibration method can be performed in a remote location, without the need of external sensors to measure the robot's true motion. A rover landing on another planet with unknown surface conditions would be an obvious application of this approach.

With this view in mind, we can identify two categories of hidden variables in our problem formulation. We are attempting to learn both the map of the environment and the set of motion model parameters that describe stochastic relationship between the odometry and the actual movement of the robot. To estimate the parameters of this model, we propose using an EM algorithm: The expectation step is provided by a SLAM algorithm, implemented with some initial motion model parameters. The possible trajectories postulated are then used in the maximization step to create a set of parameters which best describe the motions represented by these trajectories.

3. Motion Model Details

Let the robot's pose at any given time step be represented as $\phi = (x, y, \theta)$, where θ is the facing angle of the robot. The motion model then seeks to determine $P(\phi'|\phi, o)$, where ϕ' is the robot's pose one time step in the future, and $o = (d, t)$ is the amount of lateral and rotational movement (respectively) that odometry has reported over that time interval.

Roy and Thrun (1999) propose the following model:

$$\begin{aligned} x' &= x + D\cos(\theta + T) \\ y' &= y + D\sin(\theta + T) \\ \theta' &= \theta + T \mod 2\pi. \end{aligned}$$

Here, D is the actual distance traveled by the robot, and T is the actual turn performed. This is correct only if the turn and drive commands are performed independently, a simplifying assumption which even their own experiments violate. A simple improvement to account for simultaneous turning and lateral movement would be:

$$\begin{aligned} x' &= x + D\cos(\theta + (T/2)) \\ y' &= y + D\sin(\theta + (T/2)) \\ \theta' &= \theta + T \mod 2\pi. \end{aligned}$$

This model assumes that the turning velocity of the robot is constant throughout the time step, and that the robot can only move in the direction it is facing. These improved equations do not take into account that even in this case, the distance traveled will actually be an arc, and not a straight line. However, when T is reasonably small, this error is minor and can be absorbed as part of the noise.

A better model would take into account the ability of the robot to move in a direction that is not solely determined by the beginning and end facing angle of the robot. Such a model would be able to account for variable speed turns and sideways shifts, both of which have been apparent with our robots, even on the best of surfaces:

$$\begin{aligned} x' &= x + D\cos(\theta^*) \\ y' &= y + D\sin(\theta^*) \\ \theta' &= \theta + T \mod 2\pi. \end{aligned}$$

Here θ^* is the true movement angle of the robot. In this method, the direction of movement has been expressed separately from θ and T, which permits movement in a direction distinct from the facing angle of the robot. In practice it is often difficult to determine this independently from θ and T, but with some robots, the shaft encoders on each wheel can be read independently, and can give a more direct observation of this parameter.

Even in the rare cases where it might be possible to observe θ^*, it would be very difficult to develop a good noise model. Representing the noise in θ^* as a Gaussian would require some choice for a mean. For a robot which can perform holonomic turns, the lateral shift of the robot could very easily be in any direction, while the lateral movement reported would be negligible. In this case, θ^* would more accurately be modeled as a uniform distribution. For these reasons, we prefer a slightly different model that decomposes the movement into two principle components:

$$\begin{aligned} x' &= x + D\cos(\theta + \frac{T}{2}) + C\cos(\theta + \frac{T+\pi}{2}) \\ y' &= y + D\sin(\theta + \frac{T}{2}) + C\sin(\theta + \frac{T+\pi}{2}) \\ \theta' &= \theta + T \mod 2\pi. \end{aligned}$$

We approximate θ^* with $(\theta + \frac{T}{2})$ and refer to this direction as the *major axis* of movement. C is an extra lateral translation term, which is present to model shift in the orthogonal direction to the major axis, which we call the *minor axis*. This axis is at angle $(\theta + \frac{T+\pi}{2})$, and is defined so as to have a consistent (left-hand) orientation[3].

This motion model lends itself to a fairly natural noise model. We expect that the true values of D and T will be distributed normally with respect to the reported values, d and t, but that the mean of each will scale linearly with both d and t while the variance will scale with d^2 and t^2. This is plausible if the total noise is the sum of two independent noise sources with magnitude that scales linearly with d and t. We expect that C will have a similar dependence on d and t. In this view, C, D and T are all conditionally Gaussian given d and t:

$$\begin{aligned} C &\sim \mathcal{N}(d\mu_{C_d} + t\mu_{C_t}, d^2\sigma^2_{C_d} + t^2\sigma^2_{C_t}) \\ D &\sim \mathcal{N}(d\mu_{D_d} + t\mu_{D_t}, d^2\sigma^2_{D_d} + t^2\sigma^2_{D_t}) \\ T &\sim \mathcal{N}(d\mu_{T_d} + t\mu_{T_t}, d^2\sigma^2_{T_d} + t^2\sigma^2_{T_t}), \end{aligned}$$

where μ_{A_b} is the coefficient for the contribution of odometry term b to the mean of the distribution over A. It is these sets of mean and variance terms that we propose to learn.

4. Parameter Estimation

The learning problem for our robot is that of discovering the parameters of the distribution $P(\phi'|\phi, o)$, where o is the reported odometry. With this in mind, consider a SLAM algorithm, which uses a particle filter to produce a distribution over maps and poses at each time step. For a given set of motion model parameters, our particle filter provides a set of possible trajectories with forward probabilities (normalized particle weights) at each time step. To complete the E step, we must perform backward smoothing over our particles. There are many ways to do this with a particle filter, but we use the simplest, which is to compute the probability of each trajectory and average across successor trajectories for particles that are resampled multiple times.

To complete the M step of our EM procedure, we must compute the maximum likelihood values of the parameters in our model. The means in our model have linear contributions from the reported odometry values. We therefore determine the influence of each term on the motion parameters using a weighted least squares method. For example, let $\mu_{\mathbf{C}}$ be the column vector $[\mu_{C_d}\ \mu_{C_t}]^T$, $\mathbf{O}$ be an $N \times 2$ matrix, where each row is the reported odometric movement $[d_i\ t_i]$, and $\mathbf{C}$ be the $N \times 1$ matrix of the estimated C_i terms. We obtain the least squares solution of $\mu_{\mathbf{C}}$ from the overdetermined system:

$$\mathbf{WO}\mu_{\mathbf{C}} = \mathbf{WC},$$

where $\mathbf{W}$ is an $N \times N$ diagonal weight matrix with diagonal element i as $\sqrt{w_i}$. The process can be repeated for the other two motion terms.

The variance in our model has a quadratic dependence in the odometry terms. To compute the variance parameters for the C term in our model, $\sigma^2_{\mathbf{C}} = [\sigma^2_{C_d}\ \sigma^2_{C_t}]^T$, we define $\mathbf{O^2}$ as an $N \times 2$ matrix whose rows are the squared odometry readings, $O_i = [d_i^2\ t_i^2]$. We define $\mathbf{C}^{\sigma^2}$ as the $N \times 1$ matrix such that $C_i^{\sigma^2} = (O_i^2\mu_{\mathbf{C}} - C_i)^2$. As before, we are interested in the least squares solution to an overdetermined system of linear equations:

$$\mathbf{WO^2}\sigma^2_{\mathbf{C}} = \mathbf{WC}^{\sigma^2}.$$

The calculation is similar for the variance parameters of the other motion model terms.

The least squares solution for all 12 parameters of the motion model constitutes the M step of our EM procedure. The new model parameters can now be used for a new run of the SLAM algorithm on the same set of sensor data, and the process can be repeated until (near) convergence. This method can be applied to varying quantities of motion data. Run over the entire set of data, it can be applied off-line as a means of determining the best motion model for a robot in future deployments in the same, or similar, environment. This is useful, as it allows the algorithm to learn, with high confidence, the proper set of motion parameters, due to the large amount of training data. It also provides the operator the ability to check the performance of final motion parameters by observing the accuracy of the final trajectory.

An alternative, quasi real time application of this method would run EM on a smaller set of data, allowing the robot to learn the motion model as it explores. In this case, the robot would use a fixed size chunk of recent observations to fine tune the motion model to changes in its behavior. For example, the robot might apply this technique if it encounters a type of terrain that it has never seen before. This can slow any mapping activities undertaken by the robot. If the robot is using a SLAM algorithm for mapping, the EM nature of the model learning algorithm will require that the localization be run multiple times over each section and the mapping will no longer be real time. In practice, we expect that model tuning procedures would not be used continuously, but would be used primarily at sparse intervals or when there is some reason to believe that an inaccurate model is degrading mapping accuracy.

5. Empirical Results

We tested our algorithm on sensor logs generated by an iRobot ATRV Jr. in a cyclic hallway environment, with observations made approximately every 15cm. The robot is equipped with a SICK laser range finder, which scans

[3]There is nothing special about the left-hand choice.

at one degree increments along a semi-circle at a height of 7cm from the floor.

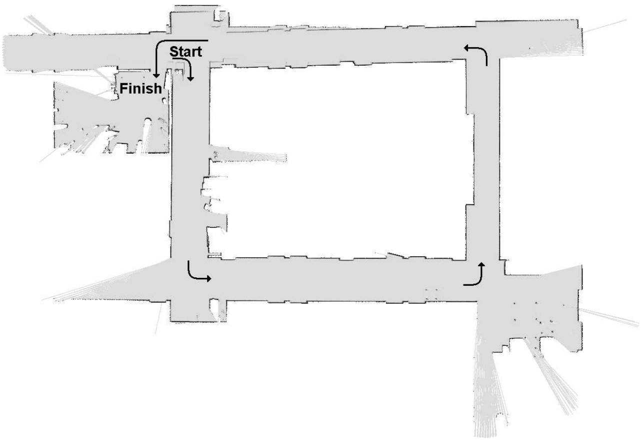

Figure 1. A complete loop of hallway, generated using a naive motion model. The robot starts at the top left and moves counterclockwise. Each pixel in this map represents 3cm in the environment. The total path length is approximately 60 meters. White areas are unexplored. Shades between gray and black indicate increasing probability of an obstacle.

In these experiments, we noticed a small anomaly where laser readings sometimes changed in a manner implying motion, when no changes were reported in odometry. This could possibly be caused by readings from the laser range finder not being perfectly synchronized with the readings from the odometers, or some other anomaly in our robot or data collection technique. Since the motion model described is directly dependent on the magnitude of reported motion, the variance in these situations would be zero, and the SLAM algorithm would have no ability to recover the correct motion for that time step. To handle this problem, we found it necessary to set a minimum amount of noise that must be present at each time step. These levels were small (variances less than 2cm along the major axis and less than $4°$ in facing), and the model exceeded these variance levels in all but a few time steps.

The first experiment demonstrates the ability of the proposed method to calibrate the motion model parameters for a robot with little or no previous knowledge of the environment. The robot is driven around an indoor test environment, eventually completing a loop of hallway, while collecting data from its sensors and odometers. Note that the completion of a loop is not necessary for either the SLAM algorithm or the learning method, but merely serves to help illustrate the quality of the map at each EM iteration. The motion model is set initially with no systematic biases, but high variances. Figure 1 shows the highest probability map produced at the end of the first run of EM. The resulting map has the right general shape, but in the top left area where the robot returns to its starting position there is a significant error in the map, resulting in double walls. A closeup of this region is shown in Figure 2. After three EM iterations, the model parameters are refined to the point where the SLAM algorithm successfully closes the loop without any blemishes in the map. A closeup of the same area is shown in Figure 3.

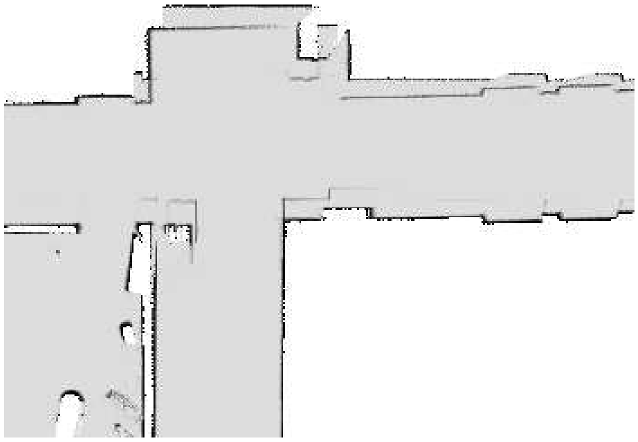

Figure 2. Close up of the area where the loop is closed, using the naive motion model. Double walls reflect an accumulated error of approximately one half meter over the path of the robot.

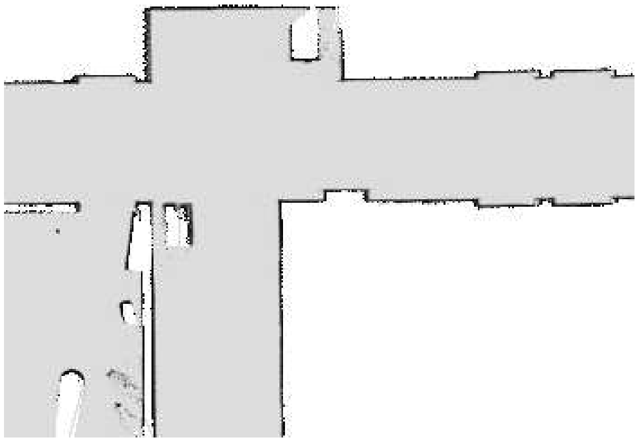

Figure 3. Close up of the same area as Figure 2, using the learned motion model learned by EM.

One concern that we had when learning a motion model was the possibility of overfitting the specific trajectory that was supplied to the SLAM algorithm. We would like the learned parameters to be tuned to the properties of the robot and environment, but not the quirks of individual data collection runs, since it would be it would be inefficient and contrary to the spirit of SLAM to learn a new motion model with EM every time that the robot is redeployed. To verify this generality of the motion model, we used one run of the robot to learn the parameters in the same indoor environment as before. Then, using this set of learned motion parameters, we had the robot remap the same environment using data collected from several days later. The resulting map shown in Figure 4 is the same high quality as if we had learned the motion model directly from the second trajectory itself.

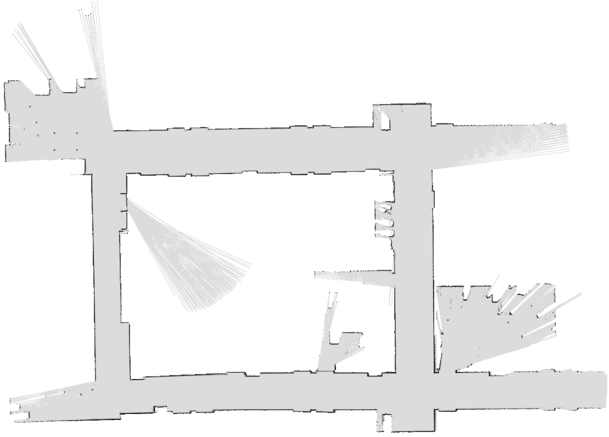

Figure 4. Map created using the motion model learned from one sensor log, applied to a different log generated several days later.

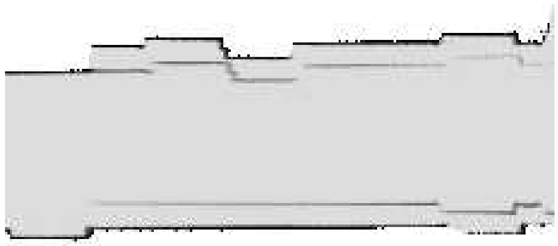

Figure 5. Map resulting from the first iteration of EM initialized with an out-of-date motion model. This is a close up of the area where the loop is closed. The environment is the same as shown for in other experiments, with a top, center starting position.

A strong test of the robustness of our method is its ability to recover from a poor motion model. This is also important to the applicability of the method since changes in the environment or in the robot itself can cause the appropriate motion model to change. In this experiment, we used a set of data collected in an office environment to learn a good motion model. We then tested the model using a second set of data collected in the same area, but with approximately a year of time separating the two data sets. When attempting to use the same model on the second data set, we quickly notice that the map produced by the SLAM algorithm is obviously flawed where it attempts to complete the loop (Figure 5). A year of use and some rough handling during shipping caused significant wear in the robot and changes in its behavior, resulting in an altered motion model. In the next iteration (Figure 6), the learned motion model can be seen to be improving the quality of the map as a result of increased accuracy. Figure 7 depicts the map from the next and final iteration, where the two ends of the loop are seamlessly aligned.

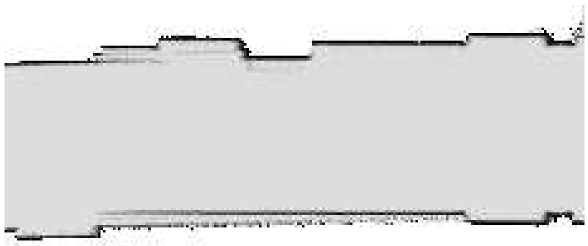

Figure 6. Second iteration of EM.

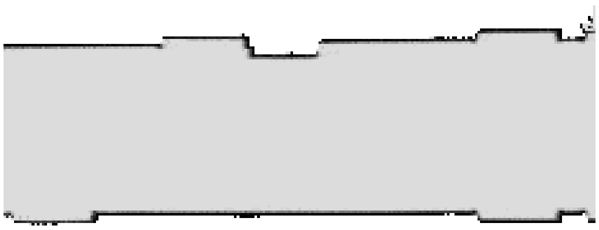

Figure 7. Final iteration of EM.

Most of our results are visual or anecdotal, since the actual parameters would be fairly meaningless to all but those very familiar with this model of robot. However, in this experiment the difference in variances is particularly telling. Predictably, the variances have all increased significantly over the course of the year, as wear on the robot has caused movements to become more erratic. The most significant of these is the σ_{C_t} term, which changes from $6.7\frac{cm}{rad}$ to $14.4\frac{cm}{rad}$, indicating significantly more erratic lateral shifts during turns, a result consistent with our observations of the robot in action.

A graphical depiction of the change in distributions is shown in (Figure 8). This graph shows the first standard deviation of the noise in the major and minor axes (along the x and y axes respectively) for a single unit of lateral motion. In the progression from the first iteration to the second, the mean is shifted significantly, while the variances decrease. The third iteration shows a negligible shift in the mean, but the variance along the major axis experiences a large increase. In comparing the difference of coverage between the first and the third iteration, it is clear that a dramatically larger number of particles would be needed for the first distribution in order to cover high probability regions of the third distribution effectively.

We also performed an experiment on a smaller segment of sensor data. We wanted to use a section with a significant amount of both lateral motion and turning within its trajectory, so we chose an area consisting of two corners

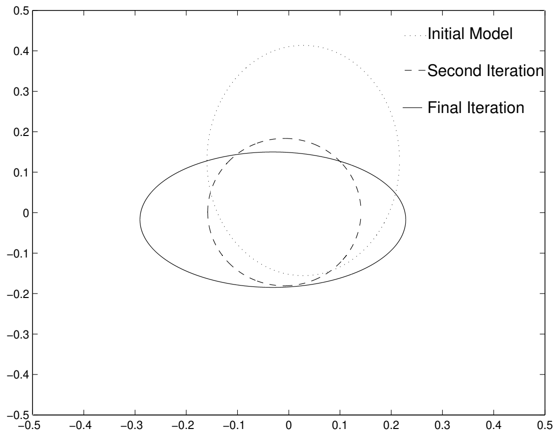

Figure 8. A plot of the first standard deviation for the third experiment. The major axis of motion is plotted along the x-axis, and the minor axis is plotted along the y-axis, with results shown for a single unit of lateral motion.

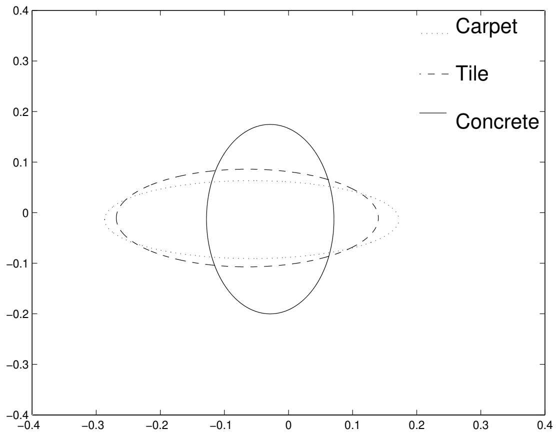

Figure 9. Motion models learned for different terrains.

connected by one lateral stretch of hallway, for a total of approximately one quarter of a full sensor log. The initial model provided was the same naive model presented in the first experiment. We performed this experiment to verify the ability to learn a model with less information and to illustrate that traversing a loop is not necessary for accurate performance of the learning method. This experiment took ten iterations to converge while those based upon full sensor logs typically took less than five. However, the final motion model parameters upon convergence of EM were accurate enough to result in seamless mapping when the algorithm was presented with a full sensor log. The resulting maps are indistinguishable from those produced with models learned from full sensor logs and are not shown.

To determine the effect of terrain type on the motion model we learned motion models for three different surfaces: carpet, tile and concrete. We used the same robot in each experiment, and all information was gathered within the period of a day, to minimize the effects of wear on the motion model. The models were then learned on a trajectory at least 20m in length, and containing at least 90 degrees of rotation. Plots showing a single standard deviation for one unit of lateral (x axis) motion for each of these motion models are shown in Figure 9. The results suggest that the greater friction of a concrete surface reduces wheel slip. However, there is also greater drift along the minor axis of movement. This is consistent with the observed behavior of the robot, but we have not diagnosed the exact cause. One possibility could be slants in sections of concrete traversed by the robot.

Finally, we considered the possibility of using very large variances and a large number of particles as an alternative to learning a good model. The problem with this approach is that adequately covering the configuration space of the robot when the model parameters have high variance is quite difficult. Even with 25 times as many samples as our refined models, our naive model was unable to produce seamless maps.

6. Conclusion

By using an EM learning algorithm coupled with a flexible motion model, we have shown how existing SLAM algorithms can greatly improve their performance by refining the stochastic motion model. We believe that our approach is the first to capture both systematic errors and variance in odometry. This technique has the potential to significantly increase the autonomy of mobile robots by eliminating the human effort required to produce a motion model.

The results presented show that the method is capable of learning accurate motion models with very little user input. Beginning with a general, naive set of motion parameters, we demonstrated the ability to refine the model to be significantly more accurate. In addition, this model was shown to be generally applicable in similar environments. Furthermore, when presented with an incorrect model, the proposed method quickly adapted, and was able to successfully learn more appropriate parameters. Finally, we demonstrated the power of this method to learn a good model that is applicable to a large area when presented with data from only a small piece of this area.

This research was motivated by our own practical considerations after investing significant amounts of time and effort into hand tuning appropriate motion models for our different robots and test environments. Beyond saving time and resources, this method was also inspired by practical concerns of remote deployment for robots. The ability to learn a complete motion model using only onboard sensors is crucial for isolated robots in unknown environments, or ones which suffer malfunction in the field.

One surprising aspect of this approach was its ability to learn good motion models despite initially poor models that lead to poor maps. One might expect that little could be learned from SLAM runs that result in poor maps since poor maps imply that the particle trajectories have failed to capture the true motion of the robot. We speculate that such poor runs still tend to contain useful information that can push the model parameters in the right direction. The reasons for this are twofold. First, although SLAM posterior distributions are not unimodal, they often appear to have strong peaks surrounded by relatively shallow local optima. Thus, on poor runs, the trajectories that are closer to truth will often still tend to have higher probability. Second, poor runs typically assign particles far from the mean higher probability, which has the effect of increasing variance on the the next iteration. On runs which are initialized with poor models, we have observed an "expand-contract" pattern in the model parameters, where the variance grows until the mean is well covered and then contracts to reflect the true variance given the correct mean.

We recognize that the calibration technique described here would not be efficient to run at all times. Instead, it would best be run at intervals when the motion has notably changed from the existent model. In the future, we would like to develop principled methods for automatically determining when a motion model needs to be updated.

This method was developed using an assumption that the true motion of the robot can be described as the sum of two independent normal distributions, arising from rotational movement and lateral movement. We would like to investigate relaxing this assumption, and allow the motion to be described by a single multivariate Gaussian with a full covariance matrix. Through learning the complete set of parameters for this multivariate distribution, we could determine to what degree this assumption of independence is valid.

In experiments, we noticed that the amounts of noise present at certain time steps were significantly lower than at others, due to differing amounts of motion. Thus we were using a number of particles consistent with sufficient coverage for the greater noise, even during time steps where the noise, and thus the necessary number of particles, was much lower. A great practical speed up might be achieved by understanding how many particles are required given the noise predicted by the model. The SLAM algorithm could then use a variable number of particles depending on the amount of noise currently present. This function could be significantly dependent on the type of environment that the robot is sensing, and could require further machine learning techniques, but is definitely an avenue of further research which could yield useful results.

Our emphasis in this paper has been the development of motion models for SLAM algorithms that use odometry and sensor data produce densely populated maps. We believe that a similar approach could be applied to landmark based SLAM algorithms, or to robots that use GPS data instead of odometry as an initial measure of robot motion.

Acknowledgements

This work supported in part by the National Science Foundation, the Sloan Foundation, and SAIC.

References

Borenstein, J., & Feng, L. (1994). *UMBmark - A method for measuring, comparing, and correcting dead-reckoning errors in mobile robots* Technical Report UM-MEAM-94-22). University of Michigan.

Burgard, W., Cremers, A., Fox, D., Hähnel, D., Lakemeyer, G., Schulz, D., Steiner, W., & Thrun, S. (1999). Experiences with an interactive museum tour-guide robot. *Artificial Intelligence*, *114*, 3–55.

Cheeseman, P., Smith, P., & Self, M. (1990). Estimating uncertain spatial relationships in robotics. In *Autonomous robot vehicles*, 167–193. Springer-Verlag.

Doucet, A., de Freitas, N., & Gordon, N. (2001). *Sequential monte carlo methods in practice*. Berlin: Springer-Verlag.

Eliazar, A., & Parr, R. (2004). DP-SLAM 2.0. *IEEE 2004 International Conference on Robotics and Automation (ICRA-04)*.

Montemerlo, M., & Thrun, S. (2002). Simultaneous localization and mapping with unknown data association using FastSLAM. *IEEE 2002 International Conference on Robotics and Automation (ICRA-02)*.

Roy, N., & Thrun, S. (1999). Online self-calibration for mobile robots. *IEEE 1999 International Conference on Robotics and Automation (ICRA-99)*.

Thrun, S. (2000). Probabilistic algorithms in robotics. *AI Magazine*, *21*, 93–109.

Thrun, S. (2002). Robotic mapping: A survey. In G. Lakemeyer and B. Nebel (Eds.), *Exploring artificial intelligence in the new millenium*. Morgan Kaufmann.

Voyles, R., & Khosla, P. (1997). Collabrative calibration. *IEEE 1997 International Conference on Robotics and Automation (ICRA-97)*.