Sensor Resetting Localization for Poorly Modelled Mobile Robots

Scott Lenser Manuela Veloso
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

We present a new localization algorithm called Sensor Resetting Localization which is an extension of Monte Carlo Localization. The algorithm adds sensor based resampling to Monte Carlo Localization when the robot is lost. The new algorithm is robust to modelling errors including unmodelled movements and systematic errors. The algorithm can be used in real time on systems with limited computational power. The algorithm has been used successfully on autonomous legged robots in the Sony legged league of the robotic soccer competition RoboCup '99. We present results from the real robots demonstrating the success of the algorithm and results from simulation comparing the algorithm to Monte Carlo Localization.

1 Introduction

This paper describes a new localization algorithm called Sensor Resetting Localization(SRL) which is an extension of Monte Carlo Localization. This localization algorithm was developed for use in the Sony legged league of the robotic soccer competition RoboCup '99. The algorithm has been tested in this domain, in separate tests with the same hardware and environment, and in simulation. The first part of this introduction is devoted to detailing the environment in which the localization algorithm operates. We start with a description of the robot and software and follow with a description of the soccer field on which the robot plays. The rest of the introduction explains what features of this domain make the localization problem especially challenging.

1.1 Hardware

The robots used in the competition and tests were generously provided by Sony [5]. The robots are a excellent platform for research and development and provide us exceptional hardware that is commercially viable. The robot consists of a quadraped designed to look like a small dog. The robot is approximately 20cm long plus another 10cm due to the head. The neck and four legs each have 3 degrees of freedom. The neck can pan almost 90° to each side, allowing the robot to scan around the field for markers. The robot stands 20cm high at the shoulder with the head extending another 10cm.

1.2 Vision

Vision is provided by a color CCD camera which a resolution of 88x60. The robot's field of view is about 60° horizontally and 40° vertically. The input from the camera is processed by color separation hardware and analyzed by the vision system to return distance and angle to any markers seen. Distance is estimated based on distance between color blob centers. Distance estimates vary by about ±15% when the robot is standing still. The mean value is usually within about 10% for distance and 7° for angle. Markers which are partially off of the camera look like noise to the vision system. The sides and edges of markers are commonly missed by the vision system. Because of these effects, sensor readings occasionally have much larger errors than usual. The vision system subtracts out the neck angle before passing the result to the localization algorithm. The neck angle is sensed using an on-board sensor that is very accurate but has a high granularity.

1.3 Locomotion

The locomotion used in all our tests was developed by Sony. Due to the complexities of legged locomotion, there is a lot of variability in the motion performed by the robot for the same motion commands. Repeatedly executing a path of approximately 5m in length with two turns results in final positions of the robot that vary by about 1m. As the robot moves, its feet slip randomly on the surface of the field contributing to the large movement error. The robot can transition among different types of motion such as forward, forward_right, right_forward, turn_right, backward, etc., each in different gaits. The robot slips randomly on the field surface when transitioning from one motion to another. Slippage from transitions can be as much as 5° but does not tend to affect x,y position much. Additional errors in movement are introduced by different results for the same robot behavior under slight changes to the environment such as slightly different field surface or somewhat lower battery charge. These

day to day variations in robot behavior are too expensive to model because of the time it takes to determine model parameters. Systematic errors of up to 25% have been observed in our lab.

1.4 The Field

The field on which the robot operates is 2.8m in length by 1.8m in width. The field is surrounded by low sloped walls that keep the robots and the ball on the field. There is a color coded goal at each end of the field, one cyan and one yellow, that were not used as input to the localization. The surface of the field is a short carpeting except in the goals where the surface is a smooth plastic.

The field is surrounded by 6 markers, 3 on each of the long sides of the field. There is a marker in each corner, and one at each end of the center line. Each marker is a color coded cylinder mounted 20cm from the surface of the field which is about eye level to the robot. Each cylinder is 20cm high and 10cm in diameter. The cylinder is divided vertically into two equal sized colored patches, one pink and either cyan, green, or yellow to indicate position along the long axis of the field (which we refer to as the x axis). Whether the pink is on the top or bottom indicates which side of the field the marker is on. Thus, there are 6 uniquely distinguishable markers to which the robot can estimate distance and angle.

1.5 Challenges

The Sony legged league of the robotic soccer competition RoboCup '99 provides a challenging domain in which to do localization. Due to the nature of legged locomotion, significant errors in odometry occur as the robot moves about. Further noise is introduced into odometry readings by frequent extended collisions with the walls and other robots. Since we are unable to detect collisions and are unable to locate other robots, we cannot model collisions with other robots introducing error in our system. Noise in the vision processing also makes localization more difficult. An additional complication is introduced by the rules which specify that under certain circumstances the robot is to be moved by the referee without telling the robot. The following challenges must be addressed by the localization system:

- errors(occasionally large) in sensor readings
- systematic errors in sensor readings
- large amounts of noise in odometry
- systematic errors in odometry
- collision induced unmodelled movements
- referee induced large unmodelled movements

2 Localization

This section gives a brief overview of the Bayesian approach to localization we used last year followed by a description of Monte Carlo Localization and Sensor Resetting Localization.

2.1 Previous Localization Approach

In previous work with the Sony dogs for the CMTrio-98 RoboCup-98 team, a vision-based navigation system was developed based on a Bayesian localization algorithm [8]. To be able to use the landmarks effectively, the CMTrio-98 robots used a probabilistic method of localization based on triangulation from two landmarks.

As described in [8], following a classical Bayesian approach, the field was discretized into grid locations and the continuous robot head angle was discretized into a set of possible values. These grid cells and robot headings define a state space. Observations of the landmarks are combined with the state space for the position calculation. The CMTrio-98 localization algorithm has two passes, and it uses the Bayes rule for the updates. The first pass incorporates observations into the probability distribution over the discretized state space of the grid cells and robot headings. The second pass updates the probabilities as a function of the movement actions selected [8].

2.2 Monte Carlo Localization

Monte Carlo Localization(MCL) [4] uses the general Bayesian approach to localization. Monte Carlo Localization represents the probability density for the location of the robot as a set of discrete samples. Since each sample is an exact point in locale space, updates of samples are easy to implement. The density of samples within an area is proportional to the probability that the robot is in that area. Since the points are not distributed evenly across the entire locale space, MCL focusses computational resources where they are most needed to increase the resolution near the believed location of the robot. The position of the robot is calculated from these samples by taking their mean. The uncertainty in this estimate can be estimated by calculating the standard deviation of the samples. There are two kinds of update operations that are applied to the sample set, movement updates and sensor updates.

Movement updates require a convolution of the locale probability density $P(l)$ with the movement probability density $P(l'|l,m)$. Movement updates are done by updating each of the samples in the sample set independently. A new sample is drawn from the move-

ment probability density $P(l'|l,m)$ to replace each locale sample. Note that l and m are known here and l' may depend on both.

Sensor updates require a multiplication of the locale probability density $P(l)$ with the sensor reading probability density $P(l|s)$. Sensor updates are done in two steps. First, the samples are given weights equal to the probability of the locale given the sensor readings $P(l|s)$. Second, a new set of unweighted locale samples is generated by randomly sampling with replacement from the old locale sample set, where the probability of a sample being selected is proportional to its weight. Note that this step never generates any new sample points. Optionally, some random noise can be added before each sensor update to help the algorithm recover from errors.

The total weight of the samples gives an indication of the approximation error in the resampling step. This total weight is also proportional to the probability of the the locale sample set given the sensor readings. One possible use of this information is to adjust the sample set size dynamically to try to keep errors roughly constant.

For a more complete description of Monte Carlo Localization, please see Fox et al. [4] and Dellaert et al. [2].

2.2.1 Summary of Monte Carlo Localization

Movement update.
$P(l^{j+1}|m,l^j) = P(l^j)$ convolved $P(l'|m,l)$

1. foreach sample s in $P(l^j)$
   2. draw sample s' from $P(l'|m,s)$
   3. replace s with s'

Sensor update.
$P(l^{j+1}|s,l^j) = P(l^j) * P(l|s)/\alpha$ where α is a constant.

1. [optional step] replace some samples from $P(l^j)$ with random samples
2. foreach sample s in $P(l^j)$
   3. set weight of sample equal to probability of sensor reading, $w = P(l|s)$
4. foreach sample s in $P(l^j)$
   5. calculate and store the cumulative weight of all samples below the current sample (cw(s))
6. calculate total weight of all samples (tw)
7. foreach sample s' desired in $P(l^{j+1})$
   8. generate a random number(r) between 0 and tw
   9. using a binary search, find the sample with maximum cw(s) $< r$
   10. add the sample found to $P(l^{j+1})$

2.2.2 Monte Carlo Localization discussion

We modelled all of our movements by 3 Gaussians, one for movement distance, one for movement direction, and one for heading change. We represented our sensor distributions as 2 Gaussians, one for distance from the landmark and one for egocentric angle to the landmark. We set deviation equal to a percentage of the mean for distance and a constant value for angle.

We attempted to do one sensor update stage for each movement update stage. However, we discovered that even when using only 400 samples the localization algorithm was too slow. To make the algorithm real time, we ignored sensor readings whenever the localization algorithm fell behind on movement updates. When throwing away sensor readings, we were able to execute about twice as fast but sacrificed a small amount of accuracy and a large amount of precision.

More samples are normally needed by Monte Carlo Localization during global localization than when tracking since similar resolution is needed and the samples are spread over a much larger area. If MCL is run with too few samples during global localization, the locale probability density prematurely collapses to the few most likely points. During testing with 1000 samples, we commonly observed the sample set collapse to 1-2 distinct samples during global localization from a single marker observance. The number of samples remains constant, of course, but the number of distinct samples is often reduced greatly during the sensor update phase. A single marker reading localizes the robot to a circle around the marker at a set distance and a heading relative to the center of the circle. Obviously, 1-2 distinct samples are not sufficient to represent this circle accurately. Adaptive sample set sizes do not help much here since only one movement update step is resampled from. If the robot takes 2 steps and then sees a marker, the probability density will consist of 1-2 tightly grouped blobs instead of 1-2 points which doesn't fix the problem. The premature collapse of the probability density results in increased time for global localization and more misleading error in position during global localization. Since our robots are continually losing track of their position due to collisions, being moved, falling down, etc., it is extremely important that the localization algorithm be capable of globally localizing quickly.

MCL is only capable of handling small systematic errors in movement. Every sensor reading gives MCL a chance to correct a small amount of systematic error. The amount of systematic error that can be corrected for increases with larger movement deviations

and larger numbers of samples. If the systematic error in movement gets too large, MCL will slowly accumulate more and more error. We need to handle systematic errors in movement because measuring the movement parameters for a robot is time consuming. Systematic errors in movement also occur when the environment changes in unmodelled ways. For example, if the robot moves from carpeting to a plastic surface such as the goal, the movement of the robot for the same motion commands is likely to change.

MCL does not handle unexpected/unmodelled robot movements very well. The time MCL takes to recover is proportional to the magnitude of the unexpected movement. During this time, MCL reports incorrect locations. Unexpected movements happen frequently in the robotic soccer domain we are working in. Collisions with other robots and the walls result in motions having unexpected results. Collisions are difficult to detect on our robots and thus cannot be modelled by the localization algorithm. Another unexpected movement we incur is teleportation due to application of the rules by the referee. MCL takes a long time to recover from this.

Some of the drawbacks of MCL can be alleviated by adding adaptive sample set sizing. Even with adaptive sample set sizing, MCL is still more sensitive to systematic errors in movement than SRL. MCL with adaptive sample set sizing requires different computational resources as the number of samples changes. Adaptive sample set sizing as described in Fox et al. [4] can take an extremely long time if the robot thinks it is one position and the sensor readings indicate a different position, especially if robot movements are very accurate. We were unable to apply MCL with adaptive sample set sizing since we are working in a real time domain and do not have any additional computational power available.

2.3 Sensor Resetting Localization

Sensor Resetting Localization(SRL) is an extension of Monte Carlo Localization. SRL is motivated by the desire to use fewer samples, handle larger errors in modelling, and handle unmodelled movements. SRL adds a new step to the sensor update phase of the algorithm. If the probability of the locale designated by the samples we have is low given the sensor readings $P(L|s)$, we replace some samples with samples drawn from the probability density given by the sensors $P(l|s)$. The number of samples kept is proportional to the average probability of a locale sample given the sensors divided by an expected average probability of locale samples. Thus if the average probability is above a threshold, all the samples are kept. As the average probability of the locale samples falls below this threshold, more and more samples are replaced by samples based on the sensor readings $P(l|s)$. We call this sensor based resampling. The logic behind this step is that the average probability of a locale sample is approximately proportional to the probability that the locale sample set covers the actual location of the robot, i.e. the probability that we are where we think we are. Taking one minus this value as the probability of being wrong, suggests that we should replace a proportion of samples equal to the probability of being wrong with samples from the sensors. The constant of proportionality between the average probability of a locale sample and the probability of being wrong is a parameter that can be tweaked to control how often the localization algorithm resets itself.

2.3.1 Summary of Sensor Resetting Localization

Movement update.
$P(l^{j+1}|m, l^j) = P(l^j)$ convolved $P(l'|m, l)$

Same as Monte Carlo Localization.

Sensor update.
$P(l^{j+1}|s, l^j) = P(l^j) * P(l|s)/\alpha$ where α is a constant.

1-10. Same as Monte Carlo Localization.
11. calculate number of new samples, ns $= (1 -$ avg sample prob/prob threshold$) *$ num samples
12. if(ns > 0) repeat ns times
    13. draw sample(s') from $P(l|s)$
    14. replace sample from $P(l^{j+1})$ with s'

2.3.2 Sensor Resetting Localization discussion

We modelled the robot movements and sensor readings in the same way as for Monte Carlo Localization. We used a sensor based resampling threshold equal to the expected result of 20% of the samples being distributed according to the sensor Gaussians and the other 80% having probability 0.

Sensor Resetting Localization is applicable in domains where it is possible to sample from the sensor readings $P(l|s)$. This is not a problem if landmarks are being used as the sensor readings as the sensor distributions are easy to sample from. If all possible locations of the robot are known, this sensor based sampling can be done by rejection sampling. However, rejection sampling increases the run time for resampling in proportion to the probability of having to reject a sample.

One of the advantages of SRL is that fewer samples can be used without sacrificing much accuracy. This is possible in part because it is more efficient when globally localizing. When the first marker is seen during global localization, the probability of almost all of the samples is very low. Thus the average probability of a sample is ridiculously small and SRL replaces almost all the locale samples with samples from the sensors. This results in all of the samples being distributed evenly around the circle determined by the marker. So, if we are using 400 samples, we have 400 samples instead of the 1-2 of MCL to represent the circle around the marker. Naturally, this reduces misleading errors during global localization. This also reduces the time required to converge to the correct localization since the correct answer has not been thrown out prematurely. After seeing another marker the circle collapses to a small area where the circles intersect. The average probability of the locale samples now is much higher than after seeing the first marker since more samples have been concentrated in the right place by the first sensor reading. Thus, if the threshold for sensor based resampling is set correctly, no new samples will be drawn due to the second sensor readings. As long as tracking is working, no new samples are generated from he sensors and the algorithm behaves exactly like MCL.

SRL can handle larger systematic errors in movement because once enough error has accumulated, SRL will replace the current estimate of the robot's location with one based on the sensor readings, effectively resetting the localization. Adaptive sample set sizing helps MCL, but MCL is still more sensitive to systematic errors in movement and unexpected/unmodelled robot movements. SRL is also easier to apply to real time domains since the running time per step is nearly constant and easy to bound.

SRL can handle larger unmodelled movements than MCL. The localization algorithm needs to handle extended collisions with other robots and the wall gracefully. SRL does this by resetting itself if its estimate of current robot position gets too far off from the sensor readings. SRL is able to handle large unmodelled movements such as movement by the referee easily. SRL does this by resetting itself the first time it gets a sensor reading that conflicts with its estimated position. MCL would take a long time to correct for long distance teleportation such as this since enough error in movement has to occur to move the mean of the samples to the new location.

3 Results

We tested Sensor Resetting Localization on the robots provided by Sony and in simulation. In simulation testing, we also compared Sensor Resetting Localization with Monte Carlo Localization with and without random noise samples added.

We tested SRL on the real robots using the parameters we used at RoboCup '99. We used 400 samples for all tests. In order to execute in real time, we were forced to ignore about 50% of the sensor readings. Due to inevitable changes in conditions between measuring model parameters and using them, the parameter for distance moved was off $\approx 25\%$, for angle of movement $\approx 10°$, and for amount of rotation $\approx .6°$/step. The deviations reported to the localization were 10% for movement and 15% for vision. We had the test robot run through a set trajectory of 156 steps while slowly turning it neck from side to side. The robot was instructed to stop after 7 different numbers of steps had been executed. The final position of the robot was measured by hand for each run. We did five runs at each of the 7 number of steps and averaged the results. We compared the actual location of the robot at each of the 35 data points with the location reported by the localization algorithm. We calculated the error in the mean position over time and the deviation the localization reported over time. We also calculated an interval in each dimension by taking the mean reported by the localization and adding/subtracting 2 standard deviations as reported by the localization. We then calculated the distance from this interval in each dimension which we refer to as interval error. We report both average interval error and root mean squared interval error. We feel that root mean squared interval is a more appropriate measure since it weights larger, more misleading errors more heavily. We also calculated the percentage of time that the actual location of the robot fell within the 3D box defined by the x,y, and θ intervals.

The table below shows the localization is accurate within about 10cm in x and y and 15° in θ despite the erroneous parameter values. The actual location of the robot is within the box most of the time and when it is outside the box, it is close to the box. The fact that the localization seldom gives misleading information is very important for making effective behaviors. Figure 1 show that the error in position quickly converges to a steady level. Figure 2 shows that the deviation reported by the algorithm quickly converges to a fairly steady level. The deviation tends to go up at the same time the error goes up which keeps the interval error

low and avoids misleading output. In competition, we observed that the localization algorithm quickly resets itself when unmodelled errors such as being picked up occur.

	x (mm)	y (mm)	theta (°)
average error	99.94	95.14	14.29
average interval error	15.18	4.91	2.07
rms interval error	34.92	13.94	3.82
in box percentage	74.29%	80.00%	57.14%

Figure 1

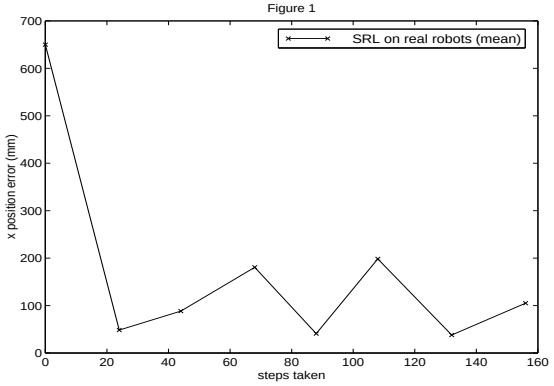
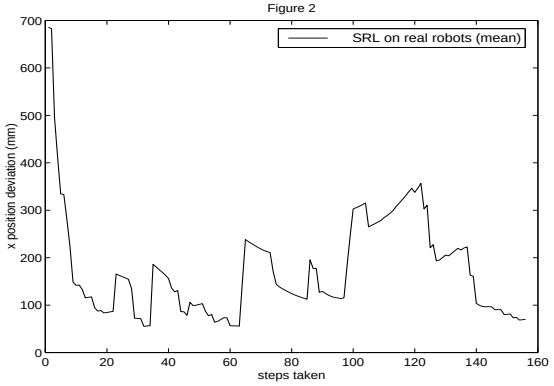


Figure 2



In simulation, we tested Sensor Resetting Localization against Monte Carlo Localization with and without random noise. Since the developers of Monte Carlo Localization suggested adding a few random samples to help the algorithm recover from errors, we tested Monte Carlo Localization adding no random noise samples and adding 5% random noise samples. Each test was run 30 times and the results averaged. All tests were run with 400 samples unless otherwise noted. The simulator models the movement of the robot with the same probability density used by the localization algorithm. Each step the probability density for motion is sampled once to generate the new position of the robot. The parameters for the movement model are the same parameters we used in RoboCup '99. The robot's vision system is modelled by assuming that the vision system correctly identifies everything within the robot's view cone and estimates the correct distance and angle to each marker. The reported deviation on the vision information as given to the localization algorithm is 15%. We modelled the movement of the robot's neck in the simulator by having it sweep through a set of degrees relative to the body at the same pace that the real robot used. The neck movement allowed the robot to see many different markers from almost all field positions.

Figure 3 shows that SRL has less error than MCL localization with small sample sets. Figure 3 and 4 show the mean error and 95% confidence interval error bands. Once the sample set has increased to about 5000 samples, the two algorithms give almost identical performance. Since performance of MCL with noise samples was slightly worse than MCL without noise samples, especially at small sample sizes, MCL with noise is not shown in these figures. Interestingly, the error of SRL stayed about constant above 100 samples was usable with only 10 samples. Figure 4 shows that SRL gives fewer misleading results than MCL. SRL gives fewer misleading results even at intermediate sample counts where the error between SRL and MCL has almost disappeared.

Figure 3

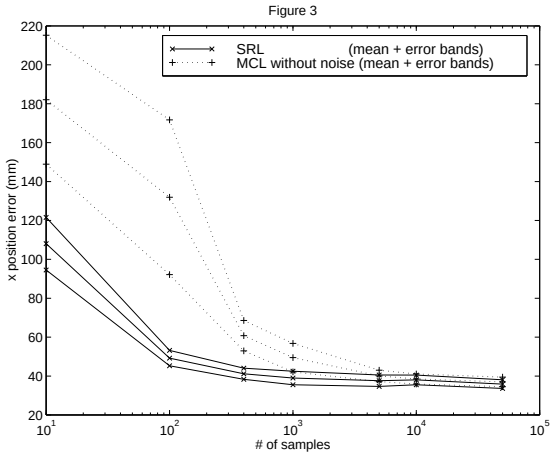
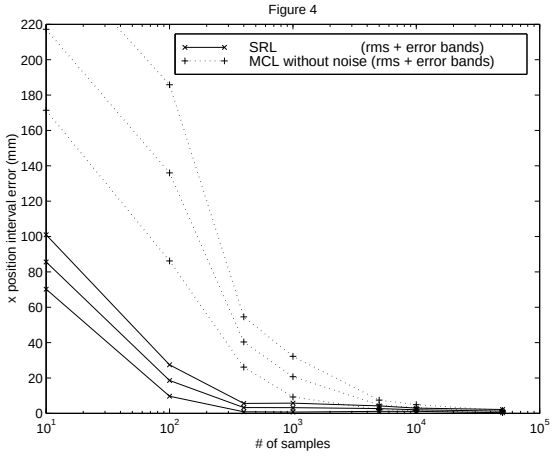


Figure 4



Figures 5 and 6 show that SRL is able to do global localization quicker than MCL using 400 samples. The figures show mean error and two standard deviations from the mean (the deviation lines are *not* error bars). SRL is able to globally localize in about 10 steps whereas MCL takes about 60 steps. In noisy domains with adversarial robots, we can't afford to spend 60 steps localizing. The global localization performed by SRL is more consistent than MCL making its output easier to reason with. Surprisingly, adding noise to MCL hurts global localization. We noticed in testing that MCL with noise pays a penalty in larger deviations than the other two algorithms. Figure 7 shows the average interval error over time for the 3 algorithms (see the experiment on real robots above for a description of interval error). Unlike SRL, MCL generates large interval errors during global localization which indicates that it is producing very misleading outputs.

Figure 5

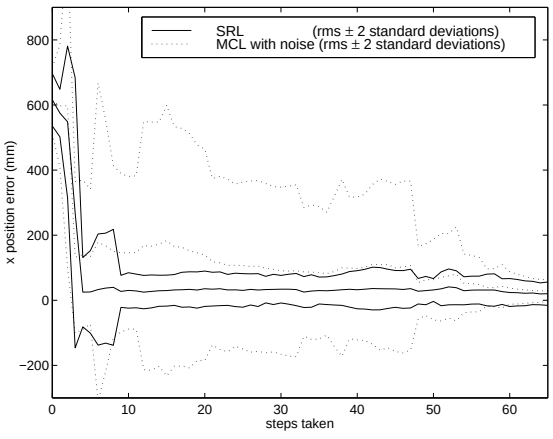


Figure 6

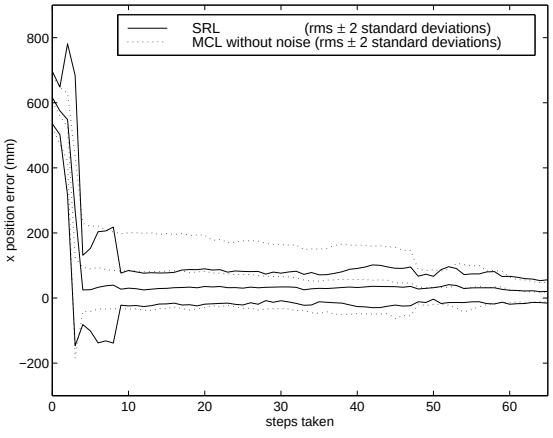
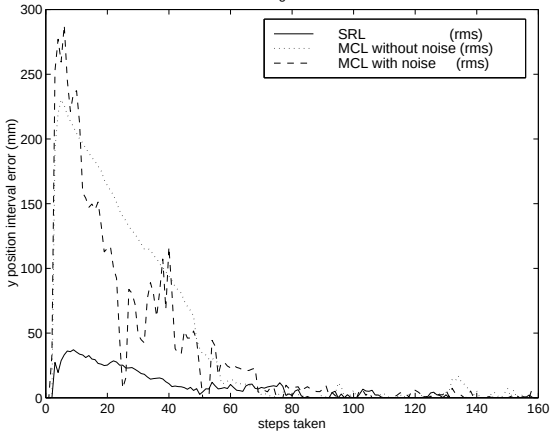


Figure 7



We also tested the response of SRL and MCL to systematic errors in movement and vision. We simulated systematic movement errors by multiplying the actual distance the robot moved and turned in the simulator by a constant factor between 0.1 and 1.9 without telling the localization algorithm. This shows the sensitivity of the algorithm to movement parameters which is important when movement parameters are expensive or difficult to measure. We also simulated systematic error in vision by multiplying all distance estimates passed to the localization by a constant factor between 0.7 and 1.75. Figures 8-11 show that SRL is more robust to modelling error than MCL especially in regards to systematic movement error. Shown in the figures is the mean and 95% confidence interval error bands.

Figure 8

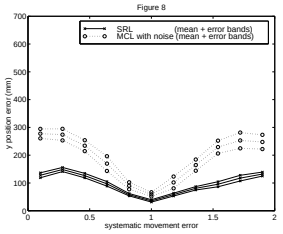


Figure 9

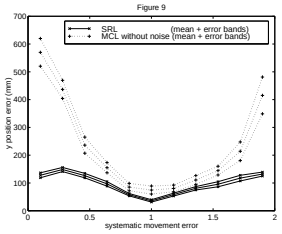


Figure 10

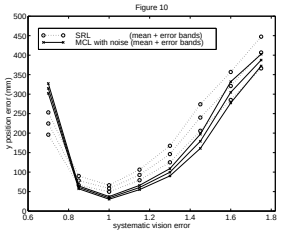
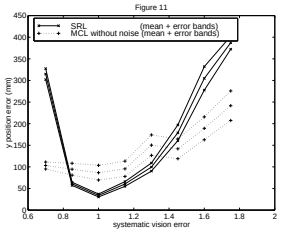


Figure 11



There is a tradeoff between robustness to systematic movement error and systematic vision error. SRL and MCL with noise favor robustness to movement errors while MCL without noise favors robustness to vision

errors. MCL without noise performs better than SRL for large systematic vision errors. SRL performs better than MCL with noise in all cases with similar response to vision errors and lower errors when presented with movement errors. SRL is much more robust to errors in movement than MCL without noise, especially when the movement error is larger than a few standard deviations.

4 Summary and Conclusions

Sensor Resetting Localization(SRL) provides an effective localization option for real time systems with limited computational resources. The technique is applicable in Markovian domains where locale samples can be drawn from the sensor readings, i.e. it must be possible to sample from $P(l|s)$. SRL requires a constant amount of processing power unlike Monte Carlo Localization(MCL) with adaptive sample set sizes which attempt to address some of the same issues as SRL. SRL achieves better results than MCL using small sample sets, which translates into smaller computational resource requirements. SRL is able to localize effectively using very small sample sets and very little computational power. Part of the reason SRL is able to accomplish this is SRL's ability to globally localize using fewer samples. SRL automatically resets itself before errors can accumulate too much allowing it to gracefully recover from errors in modelling such as collisions and teleportation. SRL is less sensitive than MCL to systematic errors in modelling except for large errors in vision when using MCL without random noise samples added. SRL is particularly robust with respect to systematic errors in movement. In addition to these benefits, the algorithm almost never returns misleading information. The algorithm correctly reports the reliability of its best guess of the location of the robot. Sensor Resetting Localization is an accurate, easy to use technique that is able to perform robust localization with very small computational resources.

Acknowledgments

We would like to thank Sony for providing us with wonderful robots and walking movements to work with. We would like to thank Jim Bruce for developing the vision system and Elly Winner for developing the behavior system for our robots. Thanks to Tucker Balch for suggesting ways to display the data collected.

This research is sponsored in part by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL) under agreement numbers F30602-97-2-0250 and F30602-98-2-0135. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing official policies or endorsements, either expressed or implied, of the Air Force or the United States Government.

References

[1] W. Burgard, D. Fox, D, Hennig, and T. Schmidt, "Estimating the absolute position of a mobile robot using position probability grids," *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, Vol. 2, pp. 896-901, 1996.

[2] F Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte Carlo Localization for mobile robots," *Proceedings of International Conference on Robotics and Automation 99*, 1999.

[3] D. Fox, W. Burgard, and S. Thrun, "Active Markov localization for mobile robots," *Robotic and Autonomous Systems,*, Vol. 25, pp. 193-207, 1998.

[4] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte Carlo Localization: Efficient Position Estimation for Mobile Robots," *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, 1999.

[5] M. Fujita, M. Veloso, W. Uther, M. Asada, H. Kitano, V. Hugel, P. Bonnin, J.-C. Bouramoue, and P. Blazevic, "Vision, strategy, and localization using the Sony legged robots at RoboCup-98," *AI Magazine*, 1999, to appear.

[6] J.-S. Gutmann, W. Burgard, D. Fox, and K. Konolige, "An experimental comparison of localization methods," *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-98)*, 1998.

[7] R. Simmons and S. Koenig, "Probabilistic robot navigation in partially observable environments," *Proceedings of International Conference on Machine Learning (ICML-95)*, 1995.

[8] M. Veloso and W. Uther, "The CMTrio-98 Sony legged robot team," In M. Asada and H. Kitano, eds, *RoboCup-98: Robot Soccer World Cup II*, Springer, 1999.