In *IEEE International Conference on Robotics and Automation* (ICRA 08),
Pasadena, CA, May 2008.

Negative Information and Line Observations for Monte Carlo Localization

Todd Hester and Peter Stone
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712
{todd,pstone}@cs.utexas.edu

***Abstract*— Localization is a very important problem in robotics and is critical to many tasks performed on a mobile robot. In order to localize well in environments with few landmarks, a robot must make full use of all the information provided to it. This paper moves towards this goal by studying the effects of incorporating line observations and negative information into the localization algorithm. We extend the general Monte Carlo localization algorithm to utilize observations of lines such as carpet edges. We also make use of the information available when the robot expects to see a landmark but does not, by incorporating negative information into the algorithm. We compare our implementations of these ideas to previous similar approaches and demonstrate the effectiveness of these improvements through localization experiments performed both on a Sony AIBO ERS-7 robot and in simulation.**

I. Introduction

Knowledge about one's location in the world is a prerequisite for many common tasks. For example, a tourist with a map of Paris who is trying to meet a friend at the Eiffel tower may try to plan a route, but can only do so after locating himself on the map. Observing and noting one's relative location to another landmark on the map, such as Notre Dame, is a particularly useful clue. From two observations of this kind, the tourist can figure out his position unambiguously. However, in the absence of such clear landmarks, more ambiguous information can also be useful. For example, observing a river indicates that one's location is somewhere along the river bank without identifying exactly where. Similarly *not* observing Notre Dame constrains one's possible locations.

Robot localization is the challenge of enabling a mobile robot to determine its location on a map of its environment based on its observations and knowledge of its past actions. Many approaches to this problem only use observations of distinct landmarks located in a specific location [1], [2], [3].

Generally, localization algorithms use observations of the world to update their belief of the pose of the robot in the environment. This pose is then adjusted by odometry estimates based on the robot's actions. Two of the most popular approaches to this problem are Kalman filtering [4], and particle filtering, or Monte Carlo localization (MCL) [5], [6]. In this paper, we focus on the MCL approach because of its ease of implementation and ability to represent multimodal pose hypotheses. In this approach, the robot's pose is represented by a set of particles, which are updated and moved at each time step based on the robot's observations and movements.

This paper presents two main enhancements to the general MCL algorithm, as applied to vision-based legged robots. First, we extend Hoffmann et al.'s method [2], [7] for using negative information by making it more robust to errors where visible landmarks go undetected. Second, we incorporate line observations into the probability updates of the algorithm. Line observations have previously been incorporated into MCL by Röfer et al. [8], [9]. Our approach differs from theirs by using lines as atomic entities instead of using individual line pixels. We also make some small but significant enhancements to the re-seeding and re-sampling portions of the algorithm. We show that the combination of these components improves the localization accuracy of the robot.

II. Background

In this section we introduce our implementation of the general MCL algorithm [5], [6] that has been applied successfully to vision-based legged robots in the past [1], [2], [3], [8], [9]. In MCL, the robot's belief of its current pose is represented by a set of particles, each of which has a hypothesis of a possible pose of the robot. Each particle is represented by $\langle h, p \rangle$ where $h = (x, y, \theta)$ is the particle's pose and p represents the probability that the particle's pose is the pose of the robot. The weighted distribution of the particle poses represents the overall belief of the robot's pose.

Our implementation of MCL is shown in Algorithm 1, which takes as parameters the set of particles P, the odometry update m, the set of observations O, and the set of known landmarks L. At each time step, the particles are updated based on the robot's actions and perceptions. The pose of each particle is moved according to odometry estimates of how far the robot has moved since the last update. The odometry updates take the form of $m = (x', y', \theta')$, where x' and y' are the distances the robot moved in the x and y directions in its own frame of reference and θ' is the angle that the robot has turned since the last time step. The odometry update to the particles is shown on line 2 of Algorithm 1.

Algorithm 1 MCL(P, m, O, L)

1: **for all** $i \in$ P **do**
2: $\quad h_i \leftarrow h_i + m$
3: $\quad p_i \leftarrow 1$
4: $\quad$ **for all** $o \in$ O **do**
5: $\quad\quad l \leftarrow$ from o
6: $\quad\quad$ **if** l is a distinct point landmark **then**
7: $\quad\quad\quad p_i \leftarrow p_i \cdot$ DISTINCT-LM-PROB(i, o, L)
8: $\quad\quad$ **else if** l is an ambiguous point landmark **then**
9: $\quad\quad\quad p_i \leftarrow p_i \cdot$ AMBIGUOUS-LM-PROB(i, o, L)
10: $\quad\quad$ **else if** l is a distinct line landmark **then**
11: $\quad\quad\quad p_i \leftarrow p_i \cdot$ DISTINCT-LINE-PROB(i, o, L)
12: $\quad\quad$ **else** $\{l$ is an ambiguous line landmark$\}$
13: $\quad\quad\quad p_i \leftarrow p_i \cdot$ AMBIGUOUS-LINE-PROB(i, o, L)
14: $\quad\quad$ **end if**
15: $\quad$ **end for**
16: **end for**
17: $P \leftarrow$RESAMPLE-PARTICLES(P) (Eq. 6)
18: $P \leftarrow$RESEED-PARTICLES(P, O)
19: $P \leftarrow$RANDOM-WALK(P)
20: $(pose, conf) \leftarrow$ WEIGHTED-AVERAGE(P)
21: Return $(pose, conf)$

For the use of this algorithm, we classify the types of landmarks that can be observed in two dimensions. The first dimension classifies landmarks as either point or line landmarks. Point landmarks only exist at one point, while line landmarks can include carpet edges or the edge of a sidewalk, where it can be difficult to determine where along the line the robot is. The second dimension describes whether landmarks are distinct or ambiguous. Distinct landmarks are uniquely identified, while ambiguous landmarks can be part of a set of similar looking landmarks, such as identical looking doors in a hallway. This taxonomy provides us with four different types of landmarks: distinct points, ambiguous points, distinct lines, and ambiguous lines.

After the odometry update, the probability of each particle is updated using the robot's perceptions. The probability of the particle is set to be $p(O|h)$, which is the likelihood of the robot obtaining the observations that it did if it were in the pose represented by that particle. The robot's observations at each time step are defined as a set O of observations $o = (l, d, \theta)$ to different landmarks, where l is the landmark that was seen, and d and θ are the the observed distance and angle to the landmark. For each observation o that the robot makes, the likelihood of the observation based on the particle's pose is calculated based on its similarity to the expected observation $\hat{o} = (\hat{l}, \hat{d}, \hat{\theta})$, where $\hat{d}$ and $\hat{\theta}$ are the the expected distance and angle to the landmark based on the particle's pose. The likelihood $p(O|h)$ is calculated as the product of the similarities of the observed and expected measurements using the following equations:

$$r_d = d - \hat{d} \tag{1}$$

$$s_d = e^{-r_d^2/\sigma_d^2} \tag{2}$$

$$r_\theta = \theta - \hat{\theta} \tag{3}$$

$$s_\theta = e^{-r_\theta^2/\sigma_\theta^2} \tag{4}$$

$$p(O|h) = s_d \cdot s_\theta \tag{5}$$

Here s_d is the similarity of the measured and observed distances and s_θ is the similarity of the measured and observed angles. The likelihood $p(O|h)$ is defined as the product of s_d and s_θ. Measurements are assumed to have Gaussian error and σ^2 represents the standard deviation of the measurement. The measurement variance affects how similar the observed and expected measurement must be to produce a high likelihood. For example, σ^2 is higher for distance measurements than angle measurements when using vision-based observations, which results in angles needing to be more similar than distances to achieve a similar likelihood. The measurement variance also differs depending on the type of landmark observed.

The calculation of the likelihood values for each type of landmark observed is done by the calls to the functions DISTINCT-LM-PROB, AMBIGUOUS-LM-PROB, DISTINCT-LINE-PROB, and AMBIGOUOUS-LINE-PROB in Algorithm 1. These functions differ in the way they determine the expected observations $\hat{o}$ to use in equations 1 to 5. DISTINCT-LINE-PROB and AMBIGUOUS-LINE-PROB calculate the probability update for an observation of a line and are explained in section III-B. DISTINCT-LM-PROB uses equations 1 through 5 to calculate the probability update for an observation of a distinct point landmark, which is the product of the likelihoods of the measured distance and angle. AMBIGUOUS-LM-PROB is explained below. The product of the likelihood values of all the observations is assigned as the new probability of the particle.

For observations of ambiguous landmarks, the specific landmark being seen must be determined to calculate the expected observation $\hat{o} = (\hat{l}, \hat{d}, \hat{\theta})$ for its likelihood calculations. With a set of ambiguous landmarks, the likelihood of each possible landmark is calculated and the landmark with the highest likelihood is assumed to be the seen landmark. The particle probability is then updated using this assumption. This probability is calculated by the call to AMBIGUOUS-LM-PROB on line 9 of Algorithm 1.

Next the algorithm re-samples the particles on line 17 of Algorithm 1. Re-sampling replaces lower probability particles with copies of particles with higher probabilities. The expected number of copies that a particle i will have after re-sampling is

$$n \times \frac{p_i}{\sum_{j=1}^{n} p_j} \tag{6}$$

where n is the number of particles and p_i is the probability of particle i. This step changes the distribution of the particles to increase the number of particles at the likely pose of the robot.

After re-sampling, new particles are injected into the algorithm through the use of re-seeding on line 18 of Algorithm 1. Histories of landmark observations are kept and averaged over the last three seconds. When two or more landmarks observations exist in the history, likely poses of the robot are calculated using triangulation. Lower probability particles are replaced by new particles that are created with these poses [10].

The pose of each particle is then updated using a random walk where the magnitude of the particle's adjustment is inversely proportional to its probability (line 19). Each particle's pose h is updated by adding $w = (i, j, k)$ where (i, j, k) are defined as:

$$i = \text{MAX-DISTANCE} \cdot (1 - p) \cdot random(1) \tag{7}$$

$$j = \text{MAX-DISTANCE} \cdot (1 - p) \cdot random(1) \tag{8}$$

$$k = \text{MAX-ANGLE} \cdot (1 - p) \cdot random(1) \tag{9}$$

The MAX-DISTANCE and MAX-ANGLE are parameters that are used to set the maximum distance and angle that the particle can be moved during a random walk and $random(1)$ is a random real number between 0 and 1. This process provides another way for particles to converge to the correct pose without re-sampling.

Finally, the localization algorithm returns an estimate of the pose of the robot based on an average of the particle poses weighted by their probability (line 20). The algorithm also returns the standard deviation of the particle poses. The robot may take actions to improve its localization estimate when the standard deviation of the particle poses is high.

III. Enhancements

The focus of this paper is on specific improvements we made to the observation model for negative information and line observations. These improvements greatly improve the localization ability of the robots. Our implementation of negative information is based on the work of Hoffmann et al. [2], [7]. We make some modifications to their algorithm to make it more suitable for use on robots in situations where the robot may frequently miss an observation of a landmark that it should have seen. We have also extended the MCL algorithm to utilize line observations in its updates. Our use of line observations is simpler than previous algorithms [8], [9], but very effective at localizing the robot. In addition to adding information from observations of these landmarks, we have also made some modifications to the re-sampling and re-seeding in the algorithm.

In our implementation of the MCL algorithm, re-sampling is only performed every n action-perception cycles[1]. This infrequent re-sampling prevents the particles from converging quickly to an incorrect pose for the robot [6]. The probability of the particles based on the most recent observations is used for the purpose of re-sampling. Informal experiments showed it to be more effective to use the probabilities based on only the most recent observations rather than all the observations since the last re-sampling was performed.

Our previous implementation of re-seeding [1] only used distinct point landmarks for triangulation. Our new algorithm improves upon this by using ambiguous landmarks for re-seeding as well. When one of the observations used for triangulation is an observation of an ambiguous landmark, triangulation is performed for all the possible landmarks and the pose that matches the observations the best is selected. Due to the extra calculation required to calculate poses for ambiguous landmarks, re-seeding is only performed every m action-perception cycles[2].

A. Negative Information

Negative information is used when an observation is expected but does not occur. If the robot is not seeing something that it expects to, then it is likely not where it thinks it is. When starting from a situation where particles are scattered widely, many particles can be eliminated even when no observations are seen because they expect to see a landmark. For each observation that is expected but not seen, the particle is updated based on the probability of not seeing a landmark that is within the robot's field of view. It is important to note that the robot can also miss observations that are within its view for reasons such as image blurring or occlusions, and these situations need to be considered when updating a particle's probability based on negative information.

Algorithm 2 shows the calculation of the probability update for a particle based on negative information, which is based on the work of Hoffmann et al. [2], [7]. The first step is to calculate which landmarks are expected to be seen from each particle pose. For every combination of particle and possible observation, we can calculate if the landmark should be within the field of view of the robot if the robot were at that particle's pose. Using the pose of the particle, we can calculate $\hat{o} = (\hat{l}, \hat{d}, \hat{\theta})$, which is the expected distance and angle to landmark $\hat{l}$. These coordinates can be transformed into coordinates in the robot's camera image using a transformation matrix (line 6 of Algorithm 2). If the landmark should exist within the camera's image, then the robot expects to see that landmark from that particle's pose (line 7).

In Hoffmann et al.'s implementation of this algorithm [2], [7], the probability of not seeing the expected observation is calculated and that value is used as the update for the particle in the observation model. In practice, however, this method can be risky. A single observation of a landmark that is within view can be missed very easily for many different reasons. For example, it is quite easy for observations to be missed due to motion distortion of the images.

To make our method more robust to missed observations of landmarks that really are within the robot's field of view, we keep a count for each particle of how many consecutive frames the robot expected to see a landmark (line 8) and the number of consecutive frames that the landmark was not seen (line 4). We define u to be the probability of missing one observation of a landmark that is within the robot's view. The particle is updated with negative information only when the robot had expected and not seen the landmark for t consecutive frames (line 15-17 of Algorithm 2). In this case, the particle's probability is updated with u^t. The experiments

[1]All the experiments in this paper were run with $n = 20$.

[2]All the experiments in this paper were run with $m = 15$.

Algorithm 2 NEGINFO-PROB(i, O, L)

$p \leftarrow 1$
for all $l \in$ L **do**
  if $l \notin O$ **then**
    Increment $notseen_l$
    Calculate $\hat{o} = (\hat{l}, \hat{d}, \hat{\theta})$
    $(x, y) \leftarrow$ Transformation of $(\hat{d}, \hat{\theta})$
    if (x, y) within camera image dimensions **then**
      Increment $expected_l$
    else
      $expected_l = 0$
    end if
  else
    $notseen_l = 0$
  end if
  if $notseen_l >= t$ and $expected_l >= t$ **then**
    $p \leftarrow p \cdot u^t$
  end if
end for
Return p

in this paper use $t = 5$, which was determined through experimentation. Lower values make the algorithm respond more quickly to missed observations, but can also result in poor localization if the robot misses many observations of landmarks that are within view. This method can be reduced to Hoffmann et al.'s method by setting $t = 1$.

B. Line Observations

Observations of lines can provide very good information to the localization algorithm. Although we cannot determine where along the line the robot is, the observation provides information about the robot's orientation and distance to the line. This information can be very valuable to the robot, especially when other observations are infrequent.

In previous work, Röfer et al. [8], [9] integrated lines into their observation model by using individual line pixels. They discretized the world into 2.5 centimeter grids and pre-calculated the angle to the closest line pixel for each grid cell. When the robot sees a line, some line pixels are randomly selected and the probability update is made based on the difference between the angles to the selected line pixels and the expected angle to the closest line pixel provided by the lookup table.

Our implementation updates particles based on observations of the lines as atomic entities instead of using individual line pixels. This approach allows us make better use of the information provided by the orientation of the lines, as well as freeing us from having to pre-calculate tables of expected line pixel observations for every location in the world. We fit lines to the line pixels in the image and update our model using the observations of these lines.

Algorithm 3 shows the DISTINCT-LINE-PROB method called on line 11 of Algorithm 1 that calculates the probability update for a particle i based on line observations. For each observed line, we calculate $o = (l, d, \theta)$ to the

Algorithm 3 DISTINCT-LINE-PROB(i, o, L)

1: $p \leftarrow 1$
2: $l \leftarrow$ from o
3: $pt \leftarrow$ closest point on l
4: $(d, \theta) \leftarrow$distance and angle to pt
5: $\hat{pt} \leftarrow$ closest point on $\hat{l}$
6: $(\hat{d}, \hat{\theta}) \leftarrow$distance and angle to $\hat{pt}$
7: $r_d \leftarrow |d - \hat{d}|$ (Eq. 1)
8: $s_d \leftarrow e^{-r_d^2/\sigma_d^2}$ (Eq. 2)
9: $r_\theta \leftarrow |\theta - \hat{\theta}|$ (Eq. 3)
10: $s_\theta \leftarrow e^{-r_\theta^2/\sigma_\theta^2}$ (Eq. 4)
11: $p \leftarrow s_d \cdot s_\theta$ (Eq. 5)
12: Return p

closest point on the observed line (lines 3-4). We calculate the expected observation $\hat{o} = (\hat{l}, \hat{d}, \hat{\theta})$ to the closest point on the line from the particle's pose (lines 5-6). Next we determine the likelihood of seeing this line on lines 7-11 of the algorithm. We update the probability of the particle with this likelihood (line 12). Thus the lines are used in the same way as point observations, by comparing expected and observed distances and angles to a point. AMBIGUOUS-LINE-PROB is a simple extension to DISTINCT-LINE-PROB that uses the likelihood of the most likely line similar to AMBIGUOUS-LM-PROB.

IV. Experimental Results

Our experiments will show the benefits of adding negative information and line observations to the localization algorithm. Localization accuracy improves when these components are used as compared to our baseline algorithm. In addition, we will show that they are particularly useful when the robot is kidnapped, or moved from one location to another.

A. Experimental Setup

Experiments were performed using the setup from the RoboCup four legged league [11]. Robocup is a useful testbed for this research because the field that the robots play on includes distinct and ambiguous point landmarks as well as ambiguous line landmarks. Games in the four legged league are played between two teams of four Sony AIBO robots on a 3.6 by 5.4 meter field. The field has two uniquely colored beacons, one on either side of the field, as well as different colored goals at each end of the field. There are also lines on the field, which provide both ambiguous point landmarks (line intersections) as well as ambiguous line landmarks. The four legged league has previously served as a useful testbed for localization research [1], [3], [7], [8].

The experiments were run on a Sony AIBO ERS-7 robot [12] which is roughly 280 mm tall and 320 mm long. The AIBO has 20 degrees of freedom. Images are captured at 30 frames per second from the CMOS color camera in the robot's head. All computation is performed on the robot.

Further experiments were performed in simulation. The UT Austin Simulator [13] works at the localization level of abstraction and does not attempt to simulate real world physics or vision. At each time step, the simulator expects high level movement commands, which it uses to update the robot's pose. In return, it provides a set of observations O to the robot by providing simulated distances and angles to landmarks. The simulator calculates which landmarks should be within the field of view of the robot and returns simulated observations to these landmarks with Gaussian noise added. The simulator will also randomly "miss" observations of landmarks some percentage of the time.

B. Localization Accuracy Experiments

Our first set of experiments took place on the actual robot and involved measuring the accuracy of the robot's localization by having it go to a target pose. We ran experiments with four different versions of the described localization algorithm. We tested the algorithm with all combinations of negative information and line observations being on and off. The purpose of these experiments was to isolate the effects of the different components of the localization algorithm and see their effects on the localization accuracy of the robot.

In these experiments, the robot self-localized and went to 14 different poses on the field, shown in Figure 1. The robot moved towards its target pose and stopped after it believed it had been within 10 centimeters and 10 degrees of the target pose for at least half of a second. Once the robot got into position, we measured its distance and angle from the target pose. We performed ten runs of the experiment for each algorithm.

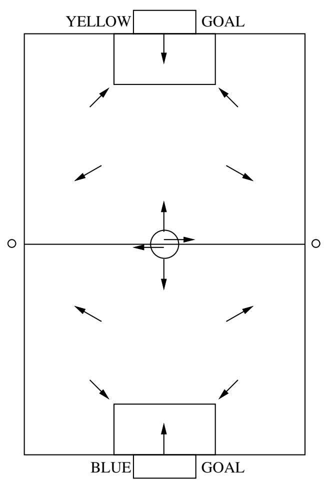

Fig. 1. Experiment Target Poses

To account for the fact that the robot was not attempting to get to the exact pose, we compared the robot's actual pose with its belief of its pose. Once the robot stopped in place, it did one head scan and output its belief of its true pose. Table I shows the error between the robot's belief and its actual pose along with p-values comparing each method to the baseline method using a one-tailed Student's t test. Results that are significantly better ($p < 0.05$) are in bold. The robot localized best when both negative information and line observations were used, performing significantly better than the baseline algorithm in distance errors. The algorithm with only line observations on was also significantly better than the baseline algorithm.

Algorithm		Belief Error			
Negative Information	Lines	Distance (cm)	p-value	Angle (deg)	p-value
Off	Off	17.67	–	5.39	–
On	Off	15.57	0.103	5.16	0.358
Off	On	**13.62**	0.010	5.08	0.381
On	On	**13.38**	0.014	4.40	0.104

TABLE I

REAL ROBOT ERROR IN POSE BELIEF

Algorithm		Belief Error					
Neg. Info.	Lines	Distance (cm)	p-value	Angle (deg)	p-value	Recovery Time (sec)	p-value
Off	Off	46.04	–	15.8	–	7.76	–
On	Off	**42.29**	$< 10^{-6}$	**14.0**	$< 10^{-7}$	7.53	0.35
Off	On	**41.19**	$< 10^{-8}$	15.3	0.11	**6.35**	0.01
On	On	**36.74**	$< 10^{-29}$	**14.2**	$< 10^{-5}$	**5.86**	$< 10^{-3}$

TABLE II

SIMULATED KIDNAPPING EXPERIMENT

C. Robot Kidnapping Experiments

Negative information can be particularly useful in situations where the robot is kidnapped. A robot kidnapping is anytime the robot is moved from one location to another without its knowledge. This situation occurs whenever a robot is picked up and moved, or when it collides with something. Since the robot is no longer in the location that it thinks it is, it should expect to see landmarks that are no longer in its field of view. We ran an experiment in simulation in which the robot followed a figure eight path around the field for 2 hours. Every 30 seconds, the robot was kidnapped and placed in the center of the field at a random orientation.

Table II shows the average distance and angular error in the robot's localization estimate over the course of the two hour experiment along with p-values calculated using a one-tailed Student's t test. Results that are significantly better than the baseline method ($p < 0.05$) are shown in bold. Once again, the method using both negative information and line observations performed the best and was significantly better than the baseline method. It was also significantly better than both intermediate methods in distance errors.

In addition to the average distance and angular error, Table II shows the average time it took the robot to recover from being kidnapped. The kidnap recovery time is defined as the time it takes the robot to achieve a localization error of less than 20 cm and 20 degrees after being kidnapped. The algorithm using both negative information and line observations performed the best. Both this method and the method using only line observations were significantly better than the baseline method.

By changing the threshold of missed observations t from 5 to 1, we were also able to run Hoffmann et al.'s version of negative information on the kidnapped robot experiment in simulation. Table III shows the results of the kidnapped robot experiment for our fully implemented algorithm with $t = 5$ and using Hoffmann et al.'s version with $t = 1$. Our method performed significantly better in terms of both distance and angular errors.

V. DISCUSSION

Our experiments showed the benefits of utilizing negative information and line observations in a Monte Carlo localization algorithm. In all of our experiments, the version of the algorithm with both components turned on performed the

Algorithm	Belief Error			
	Distance (cm)	p-value	Angle (deg)	p-value
Our Method ($t = 5$)	36.74	$-$	14.2	$-$
Hoffmann ($t = 1$)	40.11	$< 10^{-5}$	15.5	$< 10^{-3}$

TABLE III

SIMULATED COMPARISON TO HOFFMANN ET AL.'S METHOD

best. Adding these components to the algorithm allows it to make full use of all of the information in the environment. Every observation is used to update the robot's belief of its pose, and missed observations are also used to update the robot's pose estimates.

Negative information and line observations proved to be especially useful in situations where the robot has been kidnapped. Both intermediate methods were significantly better than the baseline method in distance error. The method combining the two approaches gained the benefits of each, with improvement close to double that of either component alone. It is interesting to note that using negative information caused more improvement in angular errors than using line observations. We hypothesize that this is caused by negative information being used whenever a landmark goes out of view while the robot is scanning its head. Particles with incorrect pose orientations might still expect the landmark to be within view and would be updated with negative information. Another interesting aspect of these results is that the method with negative information does not perform as well in recovery time as the one using line observations. Negative information informs the robot when it has been kidnapped, but is not sufficient for the robot to re-localize itself after the kidnapping.

The comparison to Hoffmann et al.'s algorithm showed that our method was significantly better in distance and angular errors. Hoffmann et al.'s method got some benefit from negative information as its errors were less than that of using only line observations, however their method did not gain as much from negative information as ours. Their method was not as robust as ours to missed observations, as it would update particles with negative information after a single missed observation, which could be caused by any number of factors such as motion distortion. These faster negative updates to particles can create poorer localization estimates when missed observations are frequent.

The localization algorithm presented in this paper was used on the UT Austin Villa robot soccer team during the four legged league competition at RoboCup 2007. In addition to the experiments performed in this paper, the success of this algorithm on the robots during actual game situations show its effectiveness. Our team made the quarterfinals in the competition, and based on informal observations, accurate localization was an important aspect of that success. In particular, the localization algorithm's ability to keep the goal keeper in place in front of the goal showed its accuracy.

VI. CONCLUSION

In this paper, we presented two main improvements to the Monte Carlo localization algorithm. Each improvement in isolation improves localization moderately, and taken together, they significantly improve localization accuracy when compared to a baseline MCL implementation. First, we extended the algorithm that previously used only point landmarks to use information from line observations as well. Second, we used negative information for localization pose updates, extending Hoffmann et al.'s method [2], [7] by adding a threshold of the number of frames an observation must be missed before the particles are updated. In addition to these two changes, we made some small but significant improvements to the re-seeding and re-sampling portions of the algorithm. Our contributions are fully implemented both on a Sony ERS-7 robot and in simulation. Detailed empirical results show that these improvements are beneficial to mobile robot localization.

ACKNOWLEDGEMENTS

The authors would like to thank the UT Austin Villa robot soccer team for developing the code base used in this work. We would especially like to thank Mohan Sridharan for the vision code and Gregory Kuhlmann for the simulator. Thanks as well to Daniel Stronger, Nate Kohl, and Michael Quinlan for their advice on this research. This research is supported in part by NSF CAREER award IIS-0237699 and ONR YIP award N00014-04-1-0545.

REFERENCES

[1] M. Sridharan, G. Kuhlmann, and P. Stone, "Practical vision-based monte carlo localization on a legged robot," in *IEEE International Conference on Robotics and Automation*, April 2005.

[2] J. Hoffmann, M. Spranger, D. Göhring, and M. Jüngel, "Exploiting the unexpected: Negative evidence modeling and proprioceptive motion modeling for improved markov localization." in *RoboCup*, 2005, pp. 24–35.

[3] D. Stronger and P. Stone, "A comparison of two approaches for vision and self-localization on a mobile robot," in *IEEE International Conference on Robotics and Automation*, April 2007.

[4] J. Borenstein, H. R. Everett, and L. Feng, *Navigating Mobile Robots: Systems and Techniques*. Natick, MA, USA: A. K. Peters, Ltd., 1996.

[5] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *IEEE International Conference on Robotics and Automation (ICRA99)*, May 1999.

[6] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT Press, 2005.

[7] J. Hoffmann, M. Spranger, D. Göhring, and M. Jüngel, "Making use of what you don't see: Negative information in markov localization," in *IEEE/RSJ International Conference of Intelligent Robots and Systems*, 2005.

[8] T. Röfer, T. Laue, and D. Thomas, "Particle-filter-based self-localization using landmarks and directed lines." in *RoboCup*, ser. Lecture Notes in Computer Science, A. Bredenfeld, A. Jacoff, I. Noda, and Y. Takahashi, Eds., vol. 4020. Springer, 2005, pp. 608–615.

[9] T. Röfer and M. Jüngel, "Fast and robust edge-based localization in the sony four-legged robot league," in *7th International Workshop on RoboCup*, 2003.

[10] S. Lenser and M. Veloso, "Sensor resetting localization for poorly modelled mobile robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2000.

[11] "Robocup 4 legged league." [Online]. Available: http://www.tzi.de/4legged/bin/view/Website/WebHome

[12] "Sony aibo ers-7." [Online]. Available: http://esupport.sony.com/US/perl/model-home.pl?mdl=ERS7&LOC=3

[13] P. Stone, K. Dresner, P. Fidelman, N. K. Jong, N. Kohl, G. Kuhlmann, M. Sridharan, and D. Stronger, "The UT Austin Villa 2004 RoboCup four-legged team: Coming of age," The University of Texas at Austin, Department of Computer Sciences, AI Laboratory, Tech. Rep. UT-AI-TR-04-313, October 2004.