In *The IEEE International Conference on Robotics and Automation* (ICRA 05), Barcelona, Spain, April 2005.

Practical Vision-Based Monte Carlo Localization on a Legged Robot

Mohan Sridharan and Gregory Kuhlmann and Peter Stone
Department of Computer Sciences, The University of Texas at Austin
1 University Station C0500, Austin, Texas 78712-1188
{smohan,kuhlmann,pstone}@cs.utexas.edu

***Abstract*— Mobile robot localization, the ability of a robot to determine its global position and orientation, continues to be a major research focus in robotics. In most past cases, such localization has been studied on wheeled robots with range-finding sensors such as sonar or lasers. In this paper, we consider the more challenging scenario of a legged robot localizing with a limited field-of-view camera as its primary sensory input. We begin with a *baseline* implementation adapted from the literature that provides a reasonable level of competence, but that exhibits some weaknesses in real-world tests. We propose a series of practical enhancements designed to improve the robot's sensory and actuator models that enable our robots to achieve a 50% improvement in localization accuracy over the baseline implementation. We go on to demonstrate how the accuracy improvement is even more dramatic when the robot is subjected to large unmodeled movements. These enhancements are each individually straightforward, but together they provide a roadmap for avoiding potential pitfalls when implementing Monte Carlo Localization on vision-based and/or legged robots.**

I. Introduction

One of the most fundamental tasks for a mobile robot is the ability to determine its location in its environment from sensory input. A significant amount of work has been done on this so-called *localization* problem.

One common approach to this problem is *particle filtering* or Monte Carlo Localization (MCL) [10], [11]. MCL has been shown to be a robust solution for mobile robot localization, particularly in the face of collisions and large, unexpected movements (e.g. the "kidnapped robot" problem [5]). Although this method has a well-grounded theoretical foundation, and has been demonstrated to be effective in a number of real-world settings, there remain some practical challenges to deploying it in a new robotic setting.

This paper presents a case study demonstrating the practical steps needed to make particle filtering effective and reliable on a legged robot with only vision-based sensors. Most previous implementations have been on wheeled robots with sonar or laser sensors (e.g. [4], [5]). In comparison with our setting, these previous settings have the advantages of relatively accurate odometry models and 360^o sensory information. Although MCL has been applied to legged, vision-based robots in the past [6], [7], [9], our work contributes novel enhancements that make its implementation more practical.

We begin with a baseline implementation of Monte Carlo Localization adapted from recent literature [9] that achieves a reasonable level of competence. We then present a series of innovations and adjustments required to improve the robot's performance with respect to the following three desiderata:

1) When navigating to a point, the robot should be able to stabilize quickly close to the target destination.
2) The robot should be able to remain localized even when colliding with other objects in its environment.
3) The robot should adjust quickly and robustly to sudden large movements (the kidnapped robot problem).

All of these properties must be achieved within the limits of the robot's on-board processing capabilities.

In order to achieve these desiderata, we enhance our baseline implementation with the following three additions.

1) Maintaining a history of landmark sightings to produce more triangulation estimates.
2) Using an empirically-computed unbiased landmark distance model in addition to heading for estimate updates.
3) Tuning and extending the motion model for improved odometry calculation in a way that is particularly suited to improving localization.

We empirically evaluate the effectiveness of these general enhancements individually and collectively, both in simulation and on a Sony Aibo ERS-7 robot. In combination, the methods we present improve the robot's localization ability over the baseline method by 50%: the robot's average error in its location and heading estimates are reduced to half of that with the baseline implementation. The accuracy improvement is shown to be even more dramatic when the robot is subjected to large unmodeled movements.

II. Background

In Monte Carlo Localization, a robot estimates its position using a set of samples called *particles*. Each particle, $\langle\langle x, y, \theta\rangle, p\rangle$, represents a hypothesis about the robot's *pose*: its global location (x, y) and orientation (θ). The probability, p, expresses the robot's confidence in this hypothesis. The density of particle probabilities represents a probability distribution over the space of possible poses.

Each operating cycle, the robot updates its pose estimate by incorporating information from its action commands and sensors. Two different probabilistic models must be supplied to perform these updates. The Motion Model attempts to capture the robot's kinematics. Given the robot's previous pose, and an action command, such as "walk forward at 300 mm/s" or "turn at 0.6 rad/s", the model predicts the robot's new pose. Formally, it defines the probability distribution, $p(h'|h, a)$, where h is the old pose estimate, a is the last action command executed, and h' is the new pose estimate.

The Sensor Model is a model of the robot's perceptors and environment. It predicts what observations will be made by the robot's sensors, given its current pose. The probability distribution that it defines is $p(o|h)$, where o is an observation such as "landmark X is 1200mm away and 20 degrees to my right", and h is again the old pose estimate.

Given these two models, we seek to compute $p(h_T|o_T, a_{T-1}, o_{T-1}, a_{T-2}, \ldots, a_0)$, where T is the current cycle and h_t, o_t and a_t are the pose estimate, observation and action command, respectively, for time t.

A. Basic Monte Carlo Localization

The basic MCL algorithm proceeds as follows. At the beginning of each cycle, the motion model is used to update the position of each of the m particles, $\langle h^{(i)}, p^{(i)} \rangle$ based on the current action command. The new pose, $h_T^{(i)}$ is sampled from the distribution:

$$h_T \sim p(h_T | h_{T-1}^{(i)}, a_{T-1}) \tag{1}$$

Next, the probability of each particle is updated by the sensor model, based on the current sensory data. The sensor model computes the likelihood of the robot's observations given the particle's pose, and adjusts the particle's probability accordingly. To prevent occasional bad sensory data from having too drastic an effect, a particle's change in probability is typically limited by some filtering function, $F(p_{old}, p_{desired})$. The sensor model update is given by the following equation:

$$p_T^{(i)} := F(p_{T-1}^{(i)}, p(o_T | h_{T-1}^{(i)})) \tag{2}$$

Finally, particles are resampled in proportion to their probabilities. High probability particles are duplicated and replace particles with low probability. The expected number of resulting copies of particle $\langle h^{(i)}, p^{(i)} \rangle$ is:

$$m \cdot \frac{p^{(i)}}{\sum_{j=1}^{m} p^{(j)}} \tag{3}$$

This description of the basic MCL algorithm specifies how we maintain a probabilistic model of the robot's location over time, but it omits several details. For instance, how do we obtain the motion and sensor models? And how many particles do we need? Some previous answers to these questions are surveyed in the following section.

B. MCL for Vision-based Legged Robots

A large body of research has been performed on robot localization, mostly using wheeled robots with laser and sonar as the sensors [4], [5], [3]. Here, we focus on the few examples of MCL implemented on vision-based legged robots. In particular, our approach to localization is built upon previous research done in the RoboCup legged soccer domain [8].

Our baseline approach is drawn mainly from one particular system designed for this domain [9]. In this approach, the sensor model updates for each particle are performed based on the sensed locations of landmarks with known locations in the environment (landmarks include visual markers and line intersections in the problem domain - see Figure 1). Given the particle's pose, the robot calculates the expected bearing for each observed landmark, $\alpha_{exp}^{(l)}, \quad l \in L$, where L is the set of landmarks seen in the current frame. The posterior probability for a single observation is then estimated by the following equation representing the degree to which the observed landmark bearing $\alpha_{meas}^{(l)}$ matches $\alpha_{exp}^{(l)}$:

$$s(\alpha_{meas}^{(l)}, \alpha_{exp}^{(l)}) = \begin{cases} e^{-50\omega_l^2}, & \text{if } \omega_l < 1 \\ e^{-50(2-\omega_l)^2} & \text{otherwise} \end{cases} \tag{4}$$

where $\omega_l = \frac{|\alpha^{(l)}(_{meas} - \alpha_{exp}^{(l)}|}{\pi}$. The probability, p, of a particle is then the product of these similarities:

$$p = \prod_{l \in L} s(\alpha_{meas}^{(l)}, \alpha_{exp}^{(l)}) \tag{5}$$

Lastly, the following filtering function is applied to update the particle's probability [9]:

$$p_{new} = \begin{cases} p_{old} + 0.1 & \text{if } p > p_{old} + 0.1 \\ p_{old} - 0.05 & \text{if } p < p_{old} - 0.05 \\ p & \text{otherwise} \end{cases} \tag{6}$$

An important aspect of this sensor model is that distances to landmarks are completely ignored. The motivation for this restriction is that vision-based distance estimates are typically quite noisy and the distribution is not easy to model analytically. Worse yet, there can be a strong non-linear bias in the distance estimation process which makes the inclusion of distance estimates actively harmful, causing localization accuracy to degrade (see Section IV). In this paper, we show that the bias in distance estimates can be empirically modeled such that they can be incorporated to improve sensor model updates.

In the baseline approach, to address the frequently-occurring kidnapped robot problem,[1] a few of the particles with low probability are replaced by estimates obtained by triangulation from the landmarks seen in the current frame. This process, called *reseeding*, is based upon the idea of Sensor Resetting localization [7].

A shortcoming of previous reseeding approaches is that they require at least two or three landmarks to be seen in the same camera frame to enable triangulation. In this paper, we present a concrete mechanism that enables us to use reseeding even when two landmarks are never seen concurrently.

Another significant challenge to achieving MCL on legged robots is that of obtaining a proper motion model. In our initial implementation, we used a motion model that provided reasonably accurate velocity estimates when the robot was walking at near maximum speed. However, when the robot was close to its desired location, moving at full speed caused jerky motion. The resulting "noise" in the motion model caused erroneous action model updates (Equation 1). The robot assumed that it was moving at full speed, but before its motion was completed, it received a pose estimate beyond the target. This estimate generated another motion command, leading to oscillation around the target position. In this paper, we examine the effect on this oscillation of improving the motion model with an extension to the robot's behavior.[2]

[1] In the RoboCup legged league, when a robot commits a foul, it is picked up by the referee and replaced at a different point on the field.

[2] Note that the noisy motion model is not a property of MCL or any other algorithm, but rather our own baseline implementation.

III. Enhancements

In this section we detail the three enhancements that we made to our baseline implementation to obtain significant improvements in the robot's localization accuracy. These enhancements are each individually straightforward, and they do not change the basic particle filtering approach. But together they provide a roadmap for avoiding potential pitfalls when implementing it on vision-based and/or legged robots.

A. Landmark Histories

To triangulate one's pose from fixed landmarks, either two or three landmarks must be seen, depending on whether or not distance information is used. It is possible to reseed without seeing enough landmarks simultaneously by maintaining a *history* of previous observations. Observed distances and angles to landmarks are stored over successive frames. The stored distances and angles are adjusted each frame based on the robot's known motion. Successive observations of the same landmark are averaged, weighted by their confidence, then given as input for reseeding, as described in Section II.

More precisely, for each of the N landmarks, let M_i be the number of times the robot has recently observed landmark i. We represent the j^{th} observation of the i^{th} landmark as $Obs_{i,j} = (d_{i,j}, o_{i,j}, p_{i,j}, t_{i,j})$, where d and o are the relative distance and orientation of the landmark, t is the timestamp of this observation, and p is the probability of the observation according to a vision-based confidence measure [2]. Also, let $\overrightarrow{pos_{i,j}}$ be the two-dimensional Cartesian vector representation of the observation relative to the robot.

Given a 2-D velocity vector representing the robot's current motion, $\overrightarrow{v}$, the change in position of the robot is given by:

$$\overrightarrow{\delta pos} = \overrightarrow{v} * (t_c - t_{lu}) \tag{7}$$

where t_c and t_{lu} represent the current time and the time of the last update. Then, observations are *corrected* as:

$$\overrightarrow{pos_{i,j}}\Big|_{\substack{i\in[1,N]\\ j\in[1,M_i]}} = \overrightarrow{pos_{i,j}} - \overrightarrow{\delta pos} \tag{8}$$

Next, to merge the observations corresponding to any one landmark i, we obtain the distance, heading, and probability of the aggregate landmark observations as:

$$psum_i = \sum_j p_{i,j}, \quad p_i = \frac{psum_i}{M_i} \tag{9}$$

$$d_i = \frac{\sum_j p_{i,j} d_{i,j}}{psum_i}, \quad o_i = \sum_j p_{i,j} o_{i,j}$$

Because the motion model is not very accurate and the robot can be picked up and moved to a different location frequently, the history estimates are deleted if they are *older* than a threshold in time or if the robot has undergone significant movement (linear and/or rotational). That is, we remove observations from the history if any one of the following three conditions are true:

$$t_{i,j} \geq t_{th}, \quad d_{i,j} \geq d_{th}, \quad o_{i,j} \geq o_{th} \tag{10}$$

In our current implementation, we use $t_{th} = 3\text{sec}$, $d_{th} = 15.0\text{cm}$ and $o_{th} = 10.0^o$. These thresholds were found through limited experimentation, though the algorithm is not particularly sensitive to their exact values.

B. Distance-Based Updates

To use distances to landmarks effectively in localization we must first account for the non-linear bias in their estimation. Initially, the estimation was performed as follows:

- The landmarks in the visual frame are used to arrive at displacements (in pixels) with respect to the image center.
- Using similar triangles, along with knowledge of the camera parameters and the actual height of the landmark, these displacements are transformed into distance and angle measurements relative to the robot.
- Finally we transform these measurements, using the measured robot and camera (tilt, pan, roll) parameters to a frame of reference centered on the robot.

Using this analytic approach, we found that the distances were consistently underestimated. The bias was not constant with distance to the landmarks, and as the distance increased, the error increased to as much as 20%. This error actually made distance estimates harmful to localization.

To overcome this problem, we introduced an intermediate correction function. We collected data corresponding to the measured (by the robot) and *actual* (using a tape measure) distances to landmarks at different positions on the field. Using polynomial regression, we estimated the coefficients of a cubic function that when given a measured estimate, provided a corresponding *corrected* estimate. That is, given measured values X and actual values Y, we estimated the coefficients, a_i, of a polynomial of the form:

$$y_i\big|_{y_i\in Y} = a_0 + a_1 x_i + a_2 x_i^2 + a_3 x_i^3 |_{x_i \in X} \tag{11}$$

During normal operation, this polynomial was used to compute the *corrected* distance to landmarks. Once this correction was applied, the distance estimates proved to be much more reliable, with a maximum error of 5%. This increased accuracy allowed us to include distance estimates for both probability updates and reseeding.

C. Extended Motion Model

In our baseline implementation, an inaccurate motion model prevented the robot from being able to precisely navigate to a specific location on the field, a desirable goal. To overcome this obstacle, we modified the robot's behavior during localization. When navigating to a point, the robot moves at full speed when it is more than a threshold distance away. When it comes closer to the target position, it progressively slows down to a velocity almost $\frac{1}{10}$ the normal speed. The distance threshold was chosen to be $300mm$ based on experimentation to find a good trade-off between accuracy and stabilization time. The robot's performance is not very sensitive to this value.

Although this is a minor contribution to our overall set of enhancements, a properly calibrated motion model can lead to a considerable decrease in oscillation, which significantly improves the localization accuracy. Also, as we shall show, reduced oscillation leads to increased accuracy and smoother motion while not increasing the time to stabilize.

In the next section, we describe our experimental platform and the individual experiments we ran to measure the effect of these enhancements on localization performance.

IV. EXPERIMENTAL SETUP AND RESULTS

The RoboCup competition [8] is an annual robot soccer event that provides researchers with challenging practical and fundamental robotics problems. One of the competition's divisions, the Legged League, provides especially interesting localization research challenges. Here, teams of four four-legged robots, equipped with vision-based sensors, play soccer on a color-coded $2.9m \times 4.4m$ field. Figure 1 shows one of the robots along with an overhead view of the playing field. As seen in the diagram, there are two goals, one at each end of the field and there are four visually distinct beacons (markers), one at each corner of the field. These objects serve as the robot's primary visual landmarks for localization. The white field lines and borders provide additional visual cues that the robot can use to localize.

Fig. 1. An Image of the Aibo and the field. The robot has a field-of-view of 56.9^o (hor) and 45.2^o (ver), by which it can use the 2 goals and 4 visually distinct beacons at the field corners for the purposes of localization.

A. Test-bed Robot

In our experiments, we used the standard Legged League robot, the Sony Aibo ERS-7 [1]. It is roughly $280mm$ tall (head to toe) and $320mm$ long (nose to tail). It has 20 degrees of freedom: 3 in its head, 3 in each leg, and 5 more in its mouth, ears and tail. It is equipped with a CMOS color camera at the tip of its nose with a horizontal field-of-view of 56.9^o and a vertical field-of-view of 45.2^o. Images are captured at 30 frames per second in the YCbCr image format.

The robot also has a wireless LAN card that allows for communication with other robots or an off-board computer. An important point to note is that all processing is performed on-board the robot, using a 576MHz processor.

B. Simulator

Debugging code and tuning parameters are often cumbersome tasks to perform on a physical robot. Particle filtering implementations require many parameters to be tuned. In addition, the robustness of the algorithm often masks bugs, making them difficult to track down. To assist us in the development of our localization code, we constructed a simulator. Conveniently, the simulator has also proven to be useful for running experiments like the ones presented in this paper.

The simulator does not attempt to simulate the camera input and body physics of the actual Sony Aibo. Instead, it interacts directly with the localization level of abstraction. Observations are presented as distances and angles to landmarks relative to the robot. In return, the simulator expects high-level action commands to move the robot's head and body. The movement and sensor models both add Gaussian noise to reflect real-world conditions. Details can be found in the Austin Villa team's technical report [2].

C. Experimental Methodology

According to the desiderata presented in Section I, we set out to evaluate the robot's localization performance based on:

- Overall accuracy;
- Time taken to navigate to a series of points;
- Ability to stabilize at a target point;
- Ability to recover from collisions and "kidnappings."

We devised a group of experiments to measure the effects of our enhancements with regard to these metrics. Though we ran as many experiments as possible on the actual robot, we found it necessary to use the simulator for the recovery experiments because it allowed us to consistently reproduce collisions and kidnappings while accurately tracking the robot's true pose over time.

D. Test for Accuracy and Time

To test the effect of the incorporated enhancements on overall accuracy and time, we designed a task in which the robot was required to visit a sequence of 14 points on the field as depicted in Figure 2. The robot was allowed to repeatedly scan the environment.[3] For each run, we measured the time taken and the error in position and angle at each point.

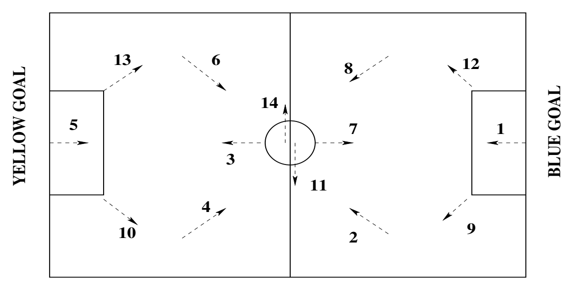

Fig. 2. Points, numbered in sequence, that the robot walks to during experimentation; arrows indicate the robot's target heading.

To measure the individual effects of the added enhancements, we performed this task over six different conditions:

1) Baseline Implementation (None).
2) Baseline + Landmark Histories (HST).
3) Baseline + Distance based probability updates (DST).
4) Baseline + Function approximation of distances (FA).
5) Baseline + Function approx. of distances and distance-based probability updates (DST+FA).
6) Baseline + All enhancements (All).

These conditions were chosen to test each enhancement in isolation and in combination. Further, it allows us to demonstrate that using landmark distances can be harmful

[3]In general, the robot is not able to scan the field constantly - we allow it to do so in these experiments for the purposes of consistency

if the distance estimates are not properly corrected. In all six conditions, we used the behavior-based motion model enhancement (Section III-C). The impact of the extended motion model was tested separately on a task requiring especially accurate positioning (See Section IV-F).

The results of these experiments, averaged across ten runs each, are shown in Tables I and II. The localization errors were computed as the distance between the robot's center and the target location when the robot indicated that it believed it had reached the target. Significance is established using a *Student's t-test*. The p-values measure the likelihood that each entry differs from the baseline algorithm (labeled *None*). We follow the convention of using a p-value of < 0.05 ($> 95\%$ confidence) to establish statistical significance.

Enhan.	Distance		Angle	
	Error (cm)	p-value	Error (deg)	p-value
None	19.75±12.0	–	17.75±11.48	–
HST	17.92±9.88	0.16	10.68±5.97	10^{-10}
DST	25.07±13.73	10^{-4}	9.14±5.46	10^{-13}
FA	15.19±8.59	10^{-4}	10.21±6.11	10^{-11}
DST+FA	13.72±8.07	10^{-6}	9.5±5.27	10^{-13}
All	9.65±7.69	10^{-15}	3.43±4.49	$< 10^{-15}$

TABLE I

ERRORS IN POSITION AND ORIENTATION

Enhan.	Time (sec)	p-value
None	161.25.75±3.43	–
HST	161.26±5.96	0.75
DST	196.18±12.18	10^{-6}
FA	171.85±15.19	0.04
DST + FA	151.28±48.06	0.56
All	162.54±4.38	0.43

TABLE II

AVERAGE TIME TAKEN PER RUN

From Table I it is evident that the error in pose is considerably reduced after the addition of all the enhancements. There is a 50% reduction in position error with respect to the baseline implementation. The improvement in orientation accuracy is even more dramatic.

Though the addition of observation histories alone (*HST*) does not significantly improve accuracy, when combined with the other enhancements, there is a significant improvement ($p - value$ between *DST+FA* and *All* $=< 10^{-15}$).

The general reluctance to utilize distance-based probability updates in localization is explained by the fact that when they are used without accounting for the non-linear bias through function approximation, the error is more than even the baseline implementation. By itself, *FA* produces a good reduction in error because the improved distance estimates lead to better reseed values, even if they are not used for probability updates. Using both, i.e. *DST+FA*, results in the biggest improvement (*p-value* $= 10^{-15}$ w.r.t DST) after *All*.

From Table II, we see that the addition of all of the enhancements does not significantly increase the time taken to reach the target locations. However, using *DST* without incorporating *FA* or *HST* does lead to worse time performance because the robot has trouble settling at a point. Since the oscillation happens even with the extended motion model in place, it can be attributed solely to bad probability updates. Though the average time taken is the lowest for *DST+FA*, this was not significant (p-value between *DST+FA* and *All* is 0.51).

E. Test for Stability

In addition to being able to localize precisely, once the robot has arrived at the target location, it should stay localized, a property we refer to as *stability*. To test stability, we developed the following experiment. The robot is placed at each one of the points shown in Figure 2 and is given ten seconds to localize. Subsequently, for twenty seconds, the robot remains stationary and records its pose estimates, at the end of which period the robot calculates the deviation in its pose estimates. Since the robot does not actually move during this period, changing pose estimates reflect erroneous oscillation.

Table III summarizes the results. The values shown in the table are the average deviation in pose estimates over ten runs at each point. These 140 data points reflect the average deviation obtained from roughly 600 samples each.

Enhan.	Dist Error (cm)	p-value	Ang Error (deg)	p-value
None	2.63	–	0.678	–
HST	1.97	10^{-5}	0.345	$< 10^{-15}$
DST	9.26	$< 10^{-15}$	3.05	$< 10^{-15}$
FA	1.46	10^{-10}	0.338	$< 10^{-15}$
DST+FA	4.07	10^{-8}	1.30	$< 10^{-15}$
All	1.32	10^{-12}	0.332	$< 10^{-15}$

TABLE III

AVERAGE DEVIATION IN POSITION AND ORIENTATION

Based on these results, we can conclude that the addition of all enhancements provides a significant improvement in stability. The use of *DST* without the other two enhancements (*HST*, *FA*) once again performs the worst. It is surprising that *FA* does as well as *All* ($p_{dist} = 0.413, p_{ang} = 0.743$). Then again, because the distances are being estimated well, the robot gets better reseed estimates. This confirms our hypothesis that landmark distances can be useful enhancements.

F. Extended Motion Model

We performed an additional experiment to determine the importance of a well-calibrated and extended motion model. In the robot soccer domain, one of the robots typically plays the role of keeper and must stay well-localized within the goal box (points 1 and 5 in Figure 2). Doing so requires precise and controlled movements. We performed an experiment in which the robot, playing the role of a keeper, tries to stay localized at its base position. We repeatedly moved the robot out of its position to the center of the field and let it move back to its base position. We then measured the error in the pose estimate over ten runs of this test both with and without an extended motion model. The results are shown in Table IV.

Enhan.	Error		
	Dist (cm)	Ang (deg)	Time (sec)
None	12.89±2.48	15.0.±9.72	17.21±1.25
Extended MM	7.496±1.96	5.5±4.97	18.14±2.25

TABLE IV

ERRORS WITH AND WITHOUT THE EXTENDED MOTION MODEL

From the results in Table IV, we see that incorporating an extended motion model significantly boosts the localization performance. The corresponding increase in time is not large enough to be disruptive.

G. Recovery

Finally, we performed a set of experiments using our simulator to test the effect of our enhancements on the robot's ability to recover from unmodeled movements. We tested our localization algorithm against two types of interference that are commonly encountered in RoboCup soccer: collisions and kidnappings. In both cases, we disrupt the robot once every 30 (simulated) seconds while it attempts to walk a figure-8 path around the field and scan with its head. Collisions are simulated by preventing the robot's body from moving (unbeknownst to it so it continues to perform motion updates) for 5 seconds. The robot's head is permitted to continue moving. In the kidnapping experiments, the robot is instantly transported to a random spot on the field, 1200mm from its previous location and given a random orientation. We test our enhancements by comparing these scenarios to the case in which the robot simply walks a figure-8 pattern undisturbed. Twice per second, we record the absolute error in robot's pose estimate. We compare average errors for 2 hours of simulated time, corresponding to roughly 50 laps around the field. The results are summarized in Tables V and VI.

Enhan.	Distance Error (cm)		
	Undisturbed	Colliding	Kidnapped
None	8.03 ± 4.92	27.7 ± 22.4	74.3 ± 55.2
HST	17.6 ± 16.2	25.3 ± 21.5	27.3 ± 36.4
DST+FA	7.83 ± 5.35	16.2 ± 16.9	31.5 ± 41.6
All	8.67 ± 9.68	14.4 ± 15.6	13.5 ± 23.4

TABLE V

ERRORS FOR VARIOUS PERTURBATIONS.

Enhan.	Angle Error (deg)		
	Undisturbed	Colliding	Kidnapped
None	2.74 ± 2.20	7.15 ± 9.33	15.3 ± 22.7
HST	3.69 ± 3.15	10.7 ± 21.4	6.66 ± 15.3
DST+FA	2.91 ± 2.36	7.07 ± 11.0	9.81 ± 21.2
All	2.38 ± 2.31	5.57 ± 11.1	4.38 ± 13.5

TABLE VI

ERRORS FOR VARIOUS PERTURBATIONS.

From the distance error table, we can see that for every test condition, the collision and kidnapping disturbances caused an increase in average error when compared to the undisturbed scenario, as we would expect. However, the increase in error is much smaller when our enhancements were used. For instance, with all enhancements turned off, the kidnapped robot problem causes almost a 10-fold increase in error. But when we include our enhancements, the error is only increased by 56%. There is a corresponding improvement in angle accuracy.

When we look at the individual contributions of the enhancements, we see that for both disturbance scenarios, the individual enhancements are better than no enhancements. Also, in both cases, they have a larger effect when combined than when they are used individually. This implies that both enhancements contribute to the improved recovery. Only in the case of *DST+FA*, when comparing angle accuracy to the baseline for collisions, do we not see a significant improvement. This is most likely explained by the limited effect of distance updates on orientation accuracy in general.

Finally, looking at the "Undisturbed" column, we notice that the enhancements, if anything, have a somewhat negative effect in the absence of interference. It would seem that this finding is incompatible with the accuracy results reported in Table I. However, the experimental conditions in the two scenarios were significantly different. In particular, in the recovery experiments, we did not wait for the robot's estimate to stabilize before measuring the error. This metric not only measures how well the robot localizes when it has good information, but also how poorly it does when it is lost, which is why this metric is appropriate for testing recovery.

Although the distance error for *HST* is quite high, in the *All* case we do only slightly worse than originally. Thus, although the algorithm has been tuned to work well in game situations where collisions and kidnappings are common, we don't give up much accuracy when those disturbances do not occur.

V. CONCLUSION

In this paper, we addressed the task of performing accurate localization on a legged robot with vision as the primary sensor. This task presents new challenges in terms of sensor and motion modeling in comparison with previous approaches on wheeled robots with 360^o range sensors. Starting with a *baseline* implementation adapted from the literature, we presented several novel enhancements. We presented empirical results, both on a physical robot and in simulation that demonstrate that these enhancements improve the robot's overall accuracy, its stability when stationary, and its recovery from disturbances. Our main contribution is a detailed cataloging of some potential pitfalls and their solutions when implementing particle filtering on legged robots with limited-range of vision.

ACKNOWLEDGMENTS

The authors would like to thank the UT Austin Villa team for their efforts in developing the software used as a basis for the work reported in this paper. Special thanks to Daniel Stronger for his specific contributions to the landmark-distance calibration process. This research is supported in part by NSF CAREER award IIS-0237699, ONR YIP award N00014-04-1-0545, and DARPA grant HR0011-04-1-0035.

REFERENCES

[1] The Sony Aibo robots. `http://www.us.aibo.com`.

[2] P. Stone et al. UT Austin Villa 2004: Coming of Age, AI Technical Report 04-313. Technical report, Department of Computer Sciences, University of Texas at Austin, October 2004.

[3] S. Thrun et al. Probabilistic algorithms and the interactive museum tour-guide robot minerva. *International Journal of Robotics Research*, 19(11):972–999, 2000.

[4] D. Fox. Adapting the sample size in particle filters through kld-sampling. *International Journal of Robotics Research*, 2003.

[5] D. Fox, W. Burgard, H. Kruppa, and S. Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence*, 11, 1999.

[6] C. Kwok and D. Fox. Map-based multiple model tracking of a moving object. In *The International RoboCup Symposium*, Lisbon, 2004.

[7] Scott Lenser and Manuela Veloso. Sensor resetting localization for poorly modelled mobile robots. In *The International Conference on Robotics and Automation*, April 2000.

[8] Daniel Polani, Brett Browning, Andrea Bonarini, and Kazuo Yoshida, editors. *RoboCup-2003: Robot Soccer World Cup VII*. Springer Verlag, Berlin, 2004.

[9] T. Rofer and M. Jungel. Vision-based fast and reactive monte-carlo localization. In *The IEEE International Conference on Robotics and Automation*, pages 856–861, Taipei, Taiwan, 2003.

[10] S. Thrun. Particle filters in robotics. In *The 17th Annual Conference on Uncertainty in AI (UAI)*, 2002.

[11] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust monte carlo localization for mobile robots. *Journal of Artificial Intelligence*, 2001.