

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

<http://www.cs.utexas.edu/~{smohan, kuhlmann, pstone}>

Abstract—Mobile robot localization, the ability of a robot to determine its position and orientation in a global frame of reference, 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 limited-field-of-view vision as the 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, and even more dramatic improvements when the robot is subjected to large unexpected movements. These enhancements are each individually straightforward, and they do not change the basic particle filtering approach. But together they provide a practical guide for avoiding potential pitfalls when implementing it on vision-based and/or legged robots. Our complete localization system is fully implemented on the Sony ERS-7 robot platform. We present extensive empirical results, both in simulation and on the physical robots, isolating the impacts of our contributions.

Keywords: Localization and Mapping, Legged Robots, Monte Carlo Localization, Particle Filtering.

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], [12]. 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 [4]). 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. [3], [4]). In comparison with our setting, these previous settings have the advantages of relatively accu-

rate odometry models and 360° sensory information. Although MCL has been applied to legged, vision-based robots in the past [5], [6], [8], our work contributes novel enhancements that make its implementation more practical and effective.

We begin with a baseline implementation of Monte Carlo Localization adapted from recent literature [8] 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 using only a vision-based sensor, and within the limits of the robot’s on-board processing capabilities.

In order to achieve these desiderata, we enhance our baseline particle filtering 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 robot heading for estimate updates.
- 3) Tuning and extending the robot behavior and associated 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 unexpected movements.

The remainder of this paper is organized as follows. We first describe, in Section II, the baseline localization algorithm on which our approach is built. In Section III we present our enhancements to this algorithm. Section IV describes the

experimental setup and discusses the experimental results. Finally, we provide our conclusions in section V.

II. BACKGROUND

In Monte Carlo Localization, a robot estimates its position using a set of samples called *particles*. Each particle, $\langle (x, y, \theta), p \rangle$, 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. First, the Movement Model attempts to capture the robot's kinematics. Given the robot's previous pose, and an action command, such as "walk straight ahead at 300 mm/s" or "turn clockwise at 0.6 rad/s", the model predicts the robot's new pose. More 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 $p(h_T|o_T, a_{T-1}, o_{T-1}, a_{T-2}, \dots, a_0)$, 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 movement 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:

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

Next, the probabilities of each particle are 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 summarized by the following equation:

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

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:

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

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 movement 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 [3], [4], [11]. In this section, we focus on the few examples of MCL implemented on vision-based legged robots. In particular, our approach to localization is built upon previous MCL research done in the RoboCup legged soccer domain [7].

Our baseline approach is drawn mainly from one particular system designed for this domain [8]. 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, $\alpha_{exp}^{(l)}$, $l \in L$, 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 $\alpha_{exp}^{(l)}$:

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

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

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

Lastly, the following filtering function is applied to update the particle's probability [8]:

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

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 used 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 [6]. Reseeding estimates are created as new particles, one for each combination of two (if we are using distance estimates) or three (if we are not) landmarks for which we have sensory data. The new particle’s pose is computed using triangulation, and its probability is set according to the average quality of the observations involved in its computation. All reseed particles with sufficiently high probability (> 0.5) are combined with the original particles for resampling. If we have m particles, and n reseeding estimates, the expected number of copies of each particle, $\langle h^{(i)}, p^{(i)} \rangle$ is determined to be:

$$m \cdot \frac{p^{(i)}}{\sum_{j=1}^{m+n} p^{(j)}} \quad (7)$$

This equation replaces the resampling step described in Equation 3. Note that it keeps the number of particles constant (m) across successive iterations.

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. In particular, the discrete nature of a “step” makes it difficult to move small distances with the same motion that enables high speeds. 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 overshooting generated another motion command, leading to oscillation around the target position. In this paper, we examine the impact of improving the motion model on this oscillation by an extension in the robot’s behavior.²

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

¹According to the rules of 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.

²Note that the noisy motion model is not a property of any algorithm in the literature, but rather our own baseline implementation developed prior to considering the subtleties of MCL.

not change the basic particle filtering approach. But together they provide a practical guide for avoiding potential pitfalls when implementing it on vision-based and/or legged robots.

A. Landmark Histories

In order 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. Reseeding without seeing enough landmarks simultaneously is made possible by maintaining a *history* of previous observations. Doing so requires the storage of the observed distances and angles to landmarks over successive frames. Furthermore, these distances and angles must be adjusted based on the robot’s known motion. Successive observations of the same landmark are averaged, weighted by their probabilities. The probabilities are estimated using vision-based confidence measures, dependent on the distance of the landmark and how closely the camera image of the landmark matches the actual one. The merged estimate is then used as the input for the reseeding as described in Section II.

More formally, let there be N landmarks, and assume that $\forall i \in [1, N]$, the robot has observed landmark i M_i times. We represent the j th observation of the i th landmark as $Obs_{i,j} = (d_{i,j}, \vec{o}_{i,j}, p_{i,j}, t_{i,j})$, where d and \vec{o} represent the relative distance and orientation of the landmark, t represents the timestamp of this observation, and $p_{i,j}$ is the probability of this observation estimated from the vision-based confidence measures. Also, let $\vec{pos}_{i,j}$ 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, \vec{v} , the change in position of the robot is calculated as:

$$\vec{\delta pos} = \vec{v} * (t_c - t_{lu}) \quad (8)$$

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

$$\vec{pos}_{i,j} |_{\substack{i \in [1, N] \\ j \in [1, M_i]}} = \vec{pos}_{i,j} - \vec{\delta pos} \quad (9)$$

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

$$psum_i = \sum_j p_{i,j}, \quad p_i = \frac{psum_i}{M_i} \quad (10)$$

$$d_i = \frac{\sum_j p_{i,j} d_{i,j}}{psum_i}, \quad \vec{o}_i = \sum_j p_{i,j} \vec{o}_{i,j}$$

Considering that the motion model is not very accurate and that the robot can frequently be picked up and moved to a different location, 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 of the following three conditions are true:

$$t_{i,j} \geq t_{th}, \quad d_{i,j} \geq d_{th}, \quad \vec{o}_{i,j} \geq \vec{o}_{th} \quad (11)$$

In our current implementation, we use $t_{th} = 3\text{sec}$, $d_{th} = 15.0\text{cm}$ and $\vec{o}_{th} = 10.0^\circ$. These values were determined based on limited experimentation in the problem domain, though the algorithm is not particularly sensitive to their exact values.

This process modifies the observations being used as inputs to the reseeding (Equation 7). In particular, it can now be performed for several frames after a landmark is actually seen, which was not possible in the baseline implementation.

B. Distance-Based Updates

Using the distances to landmarks effectively in localization requires that we first account for the non-linear bias in the estimates of landmark distances. The distance computation is performed initially as follows:

- The landmarks in the visual frame, are used to arrive at displacements (in pixel coordinates) with respect to the image center.
- These displacements are transformed, using basic similar triangles and knowledge of the camera parameters and the actual height of the landmark in the environment, into an estimate of distance and angle 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 there was lag and noise in sensor measurements. Also, the farther the robot was from the landmark, the larger the effect of minor defects in visual recognition. As a result, the distances were consistently underestimated. The bias was not constant, and as the distances to the landmarks increased, the error increased to as much as 20%, rendering distance estimates actually harmful to localization.

To overcome this drawback, we introduced an intermediate training phase of *function approximation*. 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 distance 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:³

$$y_i|_{y_i \in Y} = a_0 + a_1 x_i + a_2 x_i^2 + a_3 x_i^3 |_{x_i \in X} \quad (12)$$

Then, 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%. At that error rate, they could be incorporated in the localization algorithm, both for the probability updates and for reseeding, which can be then be done with observations corresponding to just two landmarks.

³This operation can be performed easily using MATLAB [1].

C. Extended Behavior and Motion Model

A motion model designed based on the assumption that the robot always moves at full speed prevented us from using the baseline implementation for navigating precisely to a specific location, a capability which is desirable in many practical problem domains. To overcome this drawback, we incorporated an effective modification in the behavior of the robot during localization. When navigating towards a point, the robot moved at full speed when it was more than a threshold distance (300mm in our implementation) away from its target location. When it reached closer to the target position, it progressively slowed down to a velocity almost $\frac{1}{10}$ the normal velocity by shortening its strides. The change in stride-length is necessary due to the discrete nature of “steps” taken by legged robots. The threshold distance affects a trade-off between the accuracy achieved and the associated increase in time. We chose the threshold based on limited experimentation, and found that the robot’s behavior is not very sensitive to its exact value. Though this is a minor contribution to our overall set of enhancements, a properly calibrated motion model allowing for accurate slow movements can lead to a considerable decrease in oscillation, which significantly improves the localization accuracy. Also, as we shall show, as a result of the reduction in the oscillation about the target position when using the baseline approach, this enhancement did not cause localization to take longer than the baseline approach, but achieved greater accuracy and smoother motion.

Next, we shall describe the experimental platform and the individual experiments that we ran to measure the effect that each of the enhancements had on the localization performance.

IV. EXPERIMENTAL SETUP AND RESULTS

For the past several years, RoboCup soccer [7] has provided researchers with challenging practical and fundamental robotics problems. The RoboCup competition is an annual robot soccer event that draws international participation. One of the competition’s divisions, the Legged League, provides especially interesting localization research challenges. In this league, teams of four four-legged robots, equipped with vision-based sensors, play soccer on a color-coded $2.9\text{m} \times 4.4\text{m}$ field. Figure 1 shows one of the robots along with an overhead view of the playing field. As seen in the figure, 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 markers 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.

Our team, UT Austin Villa, has participated in the past two RoboCup legged league competitions. In 2003, our team was a newcomer to the field. We participated successfully, but were not one of the better teams [9]. In 2004, our team finished in 3rd place (out of 8) at the U.S. Open competition, and made it to the quarterfinals (from a field of 24) at the international event. The localization enhancements presented in this paper were one of the main contributing factors to our dramatic improvement in performance. In this paper, we isolate the

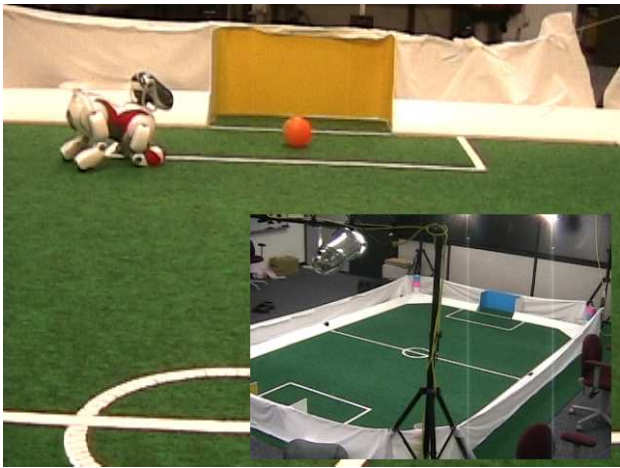


Fig. 1. An Image of the Aibo and the field. The robot has a field-of-view of 56.9° (hor) and 45.2° (ver), by which it can use the 2 goals and 4 visually distinct beacons at the field corners for the purposes of localization.

localization aspect of the robots’ overall capability to quantify the effects of some of these enhancements.

A. Test-bed Robot

In our experiments, we used the standard Legged League robot, the Sony Aibo ERS-7 [2]. The ERS-7 robot 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 that captures images at 30 frames per second in the YCbCr native image format. It has a horizontal field-of-view of 56.9° and a vertical field-of-view of 45.2° . Other sensors include touch sensors, accelerometers and distance sensors. The robot also has a wireless LAN card that allows for communication with other robots or an off-board computer. All of the robot’s processing, including vision, is performed on-board, 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 works to mask bugs, making them difficult to track down. For these reasons, we constructed a simulator that allows our localization code to interact with a simulated environment through abstracted low-level sensors and actuators. We have found that this simulator is also 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 within the robot’s viewing angle. 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

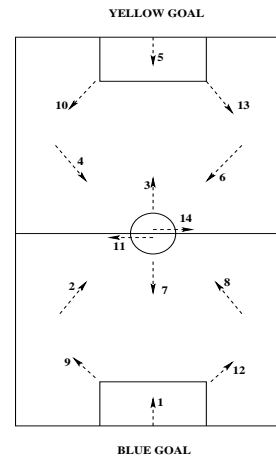


Fig. 2. Points that the robot walks to during experimentation. The arrows indicate the robot’s target heading. The points are numbered in the sequence they are to be traversed.

reflect real-world conditions. Details can be found in the UT Austin Villa team’s technical report [9].

C. Experimental Methodology

Based on the desiderata presented in Section I, we decided to evaluate the robot’s localization performance according to the following metrics:

- 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. We ran as many experiments as possible on the actual robot. However, we found it necessary to use the simulator for the recovery and confidence measure experiments. It is much easier to control collisions and kidnappings in the simulator. In addition, the simulator allows us to track the ground truth of the robot’s pose over a prolonged period of time.

D. Test for Accuracy and Time

For testing the effect of the incorporated enhancements on overall accuracy and time taken, 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. We chose the points so as to reflect a variety of levels of difficulty. For example, from some of the points (e.g. 9–14), the robot may not be able to see any landmarks, and some points (e.g. 3 and 7) are tricky to reach due to symmetry in the environment. During our experiments, the robot was allowed to repeatedly scan the environment.⁴ During each such run, we measured the time taken and the error in position and angle at each point.

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

⁴In 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.

- 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).

The goal was to be able to test each enhancement in isolation (and in combination) with respect to the metrics. Further, we wanted to demonstrate that using landmark distances can be harmful to localization unless the distance estimates are properly corrected. In all the conditions 1–6, we incorporated the behavior-based motion model compensation in which the robot took controlled steps when it was close enough to the target position (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 the robot performing this task ten times in each condition are shown in Tables I and II. The average values and standard deviations of the localization errors measured when the robot indicated that it has arrived at each target point are shown. 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*). By convention, we use a p-value of < 0.05 to indicate statistical significance ($> 95\%$ confidence) on all experiments.

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

We can draw a few conclusions from the results of Table I:

- The error in pose (position and orientation) is considerably reduced after the addition of all the enhancements. In fact, there is a 50% reduction in position error with respect to the baseline implementation. The improvement in orientation accuracy is even more dramatic.
- By itself, the addition of a history of observations (row labeled *HST*) does not significantly improve accuracy, but when used in conjunction with the other enhancements, there is a significant improvement (p-value between DST+FA and All $< 10^{-15}$).
- Addition of distance estimated for probability updates, without incorporating the function approximation to account for the non-linear bias, does indeed prove harmful to localization. The error is significantly more than even the baseline implementation. This result explains the reluctance of others to use distance-based updates.
- Function approximation of the distances, by itself, does produce a good reduction in error. This is because the

improved distance estimates lead to better reseed values, even if they are not used for probability updates.

- Using both the distance-based probability updates and function approximation to determine the distances does indeed result in significant improvement (p-value = 10^{-15} w.r.t DST); it provides the best performance other than the case in which all enhancements are incorporated.
- From the table, we notice that the confidence intervals overlap. However, the table also reports the p-values that support the significance of our claims. The enhancements complement each other such that their combination provides the best possible performance.

Enhan.	Time (sec)	p-value
None	161.25±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

Similarly, based on the values in Table II, we arrived at the following conclusions:

- The addition of all the enhancements does not significantly increase the time taken to perform the task (of walking to 14 points across the field) as compared to the baseline implementation.
- Performing the distance-based probability updates in isolation, i.e., without incorporating the function approximation and/or histories does lead to worse performance in terms of time because the robot has trouble settling at a point and spends a lot of time walking around the target position. Note that this oscillation is happening even with the extended motion model in place. Therefore, it can be attributed solely to bad probability updates that result in constantly varying pose estimates.
- For (DST+FA), the average time taken is the lowest. However, this difference was determined not to be significant (p-value between DST+FA and All is 0.51).

E. Test for Stability

In addition to being able to localize precisely, we would like to enable a robot to stay localized at the desired position once it has arrived, a property we refer to as *stability*. To test stability, we developed the following experiment. The robot was placed at each one of the points shown in Figure 2. At each point it was allowed a period of ten seconds to localize itself. Subsequently, for a period of twenty seconds, the robot remained stationary and recorded its pose estimates. At the end of the recording period, the robot calculated the deviation in its pose estimates (both position and angle). Given that the robot does not actually move during this period, changing position estimates reflect erroneous oscillation in the robot’s localization.

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 location estimates at each point.

Based on the tabulated results, we can conclude the following:

- The addition of all enhancements provides a significant improvement in stability.
- The use of distance-based probability updates without the other two enhancements (HST, FA) once again performs the worst. But when used with at least the function approximation (DST+FA), it provides good performance. This confirms our hypothesis that landmark distances can be useful enhancements, if used appropriately.
- It is somewhat surprising that HST does as well as all enhancements combined. Then again, because the robot is not moving, the landmark history is able to gather many observations. The average landmark angles are quite accurate, and thereby provide for better reseed estimates when using three-angle triangulation as opposed to the distance-based reseeding in the subsequent cases.

Enhan.	Dist Error (cm)	p-value	Ang Error (deg)	p-value
None	7.02	—	0.785	—
HST	3.58	10^{-6}	0.495	0.36
DST	11.85	$< 10^{-15}$	3.29	$< 10^{-15}$
FA	6.74	0.33	0.79	0.65
DST+FA	5.54	$< 10^{-15}$	1.47	$< 10^{-15}$
All	3.59	10^{-13}	0.776	$< 10^{-15}$

TABLE III

AVERAGE DEVIATION IN POSITION AND ORIENTATION

F. Extended Behavior and Motion Model

Next we performed an experiment to determine the importance of a well-calibrated and extended motion model allowing for slow movement near the target destination. In the robot soccer domain, we have one of the robots playing the role of a keeper whose task, much like a keeper in real soccer, involves staying well-localized within the goal box (points 1 and 5 in Figure 2) and preventing the opposition from scoring goals. Doing so requires precise and controlled movements. We performed an experiment in which the robot, playing the role of a keeper, tried 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. We performed ten runs of this test both with and without an extended motion model. The results are shown in Table IV.

From the results in Table IV, we infer:

- Incorporating an extended motion model in the *behavior* significantly boosts the localization performance.
- Also, the increase in time due to this slower motion is not significant enough to cause a problem.

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

G. Recovery

Finally, we performed a group 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 faced by robots in the RoboCup soccer domain: 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, while continually scanning with its head. Collisions are simulated by preventing the robot's body from moving (unbeknownst to it so that it continues to perform motion-based 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 unrestricted case in which the robot simply walks a figure-8 pattern undisturbed. Twice per second, we record the absolute error in the robot's pose estimate. We compare average errors for 2 hours of simulated time, corresponding to roughly 50 laps around the field.

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.

The results of the recovery experiments are summarized in Tables V and VI. 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 of these cases, they have a larger effect when combined than when they are used individually. This implies that both of the enhancements contribute to the improved recovery. Only in the case of DST+FA, when comparing

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.

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. Although the error in the DST+FA case is quite high, for “All” 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. These results do not quite correspond with our findings on the physical robot. However that is not too surprising given that the experimental conditions in the two cases were quite different. In particular, in the recovery experiments, we do not wait for the robot’s estimate to stabilize before measuring the error.

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° range sensors. Starting with a *baseline* implementation adapted from the literature, we presented several enhancements that were shown to cause a significant improvement in localization accuracy. In particular, we added the ability to store observations over several frames that helps generate estimates even when several landmarks are not seen concurrently. We also presented a method to account for the non-linear bias in distance estimation, thus allowing us to use these distance estimates for probability updates of particles to the benefit of localization performance. Finally, we have shown that making some intuitive modifications to the behavior to extend the motion model helps us position the robot precisely. 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. Though all of the experiments are run on a single robot platform, all of the enhancements are fully general and should be applicable on any robot with discrete movement cycles and/or limited field-of-view sensing capabilities. The main contribution of this paper is a detailed accounting of some potential pitfalls and ways to avoid them when implementing particle filtering on such robots.

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 specific contributions to the landmark-distance calibration process. This research is supported in part by NSF CAREER award IIS-0237699 and ONR YIP award N00014-04-1-0545.

REFERENCES

- [1] The Mathworks Website. <http://www.mathworks.com/products/matlab/>.
- [2] The Sony Aibo robots. <http://www.us.aibo.com>.
- [3] D. Fox. Adapting the sample size in particle filters through kld-sampling. *International Journal of Robotics Research*, 2003.
- [4] D. Fox, W. Burgard, H. Kruppa, and S. Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence*, 11, 1999.
- [5] C. Kwok and D. Fox. Map-based multiple model tracking of a moving object. In *The International RoboCup Symposium*, Lisbon, Portugal, 2004.
- [6] Scott Lenser and Manuela Veloso. Sensor resetting localization for poorly modelled mobile robots. In *The International Conference on Robotics and Automation*, April 2000.
- [7] Daniel Polani, Brett Browning, Andrea Bonarini, and Kazuo Yoshida, editors. *RoboCup-2001: Robot Soccer World Cup VII*. Springer Verlag, Berlin, 2004.
- [8] 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.
- [9] P. Stone, K. Dresner, P. Fiedelman, N. K. Jong, N. Kohl, G. Kuhlmann, E. Lin, M. Sridharan, and D. Stronger. Ut austin villa 2004: Coming of age. Technical report, Department of Computer Sciences, University of Texas at Austin, October 2004.
- [10] S. Thrun. Particle filters in robotics. In *The 17th Annual Conference on Uncertainty in AI (UAI)*, 2002.
- [11] S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A.B. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. Probabilistic algorithms and the interactive museum tour-guide robot minerva. *International Journal of Robotics Research*, 19(11):972–999, 2000.
- [12] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust monte carlo localization for mobile robots. *Journal of Artificial Intelligence*, 2001.