Practical Extensions to Vision-Based Monte Carlo Localization Methods for Robot Soccer Domain

Kemal Kaplan[1], Buluç Çelik[1], Tekin Meriçli[2], Çetin Meriçli[1] and H. Levent Akın[1]

[1]Boğaziçi University
Department of Computer Engineering
34342 Bebek, Ístanbul, TURKEY
{kaplanke, buluc.celik,cetin.mericli, akin}@boun.edu.tr

[2]Marmara University
Department of Computer Engineering
Göztepe, Ístanbul, TURKEY
tmericli@eng.marmara.edu.tr

Abstract. This paper proposes a set of practical extensions to the vision-based Monte Carlo localization for RoboCup Sony AIBO legged robot soccer domain. The main disadvantage of AIBO robots is that they have a narrow field of view so the number of landmarks seen in one frame is usually not enough for geometric calculation. MCL methods have been shown to be accurate and robust in legged robot soccer domain but there are some practical issues that should be handled in order to maintain stability/elasticity ratio in a reasonable level. In other words, the fast convergence ability is required in case of kidnapping. But on the other hand, fast converge can be vulnerable when an occasional bad sensor reading is received. In this work, we presented four practical extensions in which two of them are novel approaches and the remaining ones are different from the previous implementations.

Keywords: Monte Carlo localization, Vision based navigation, mobile robotics, robot soccer

1 Introduction

The question ”Where Am I?” is one of the non-trivial and challenging problems of the mobile robotics and known as the self localization problem.

Monte Carlo Localization (MCL) [1], [2] or *particle filtering* is one of the common approaches to this problem. This approach has been shown to be a robust solution for mobile robot localization; especially for unexpected movements such as ”kidnapping” [3].

The practical steps needed to make MCL reliable and effective on legged robots using only vision-based sensors, which have a narrow field of view, are presented in this paper. Most previous implementations have been on the wheeled robots using sonar

or laser sensors [3], [4] which have the advantages of 360 sensory information and relatively accurate odometry models. Although it has been applied to legged robots using vision-based sensors in the past [5], [6], [7], the works described in this paper contribute novel enhancements that make the implementation of particle filtering more practical.

This work is done as a part of the Cerberus Project of Bogazici University [8]. Cerberus made its debut in 2001 and competed in 2001, 2002, and 2003 RoboCup events. In this work, we have used Sony AIBO ERS-210 robots with 200 MHz processor as our testbed.

Organization of the rest of the paper is as follows: Brief information about Basic Monte Carlo Localization (MCL) and vision-based MCL is given in Section 2. In Section 3, information about our application platform is given. Proposed approach is explained in Section 4 in detail. Section 5 contains the results and we conclude with the Section 6.

2 Background

2.1 Basic Monte Carlo Localization

MCL is a version of Sampling / Importance Re-sampling algorithm and the variations of these techniques are also known as Particle Filters. Markov localization suffers from computation burden since it needs to express $Bel(l)$ over all the possible positions in the environment. This is usually done by dividing the environment into grids. If an accurate localization is needed, it means that the grid size should be sufficiently small. As the grids become smaller the number of grids increase and the required computations for belief propagation for entire environment becomes higher. The idea behind MCL is to represent the probability distribution function for posterior belief about the position $Bel(l)$ as a set of samples drawn from the distribution. The samples are in the format $(x, y, \Theta), p$ where (x, y, Θ) is the position and orientation of the sample and p is the discrete probability associated with the sample denoting the likelihood of being at that position of the sample. Since $Bel(l)$ is a probability distribution, sum of all p_i should be equal to 1. Belief propagation is done in terms of two types of updates. When a motion update is performed, the new samples, based on both the old ones and the provided motion estimation, are generated to reflect the change in robot's position. In the motion update, the samples are displaced according to the relative displacement fed by odometry while taking the odometric error into account. For the observation update, the p values of each particle (l, p) is multiplied by $P(s|l)$ which is the probability of receiving sensor reading s assuming that the robot is located at l. Then, updated p values are normalized to maintaining $\sum p_i = 1$.

The second phase of MCL is the so-called resampling phase. In this phase, a new sample set is generated by applying fitness-proportionate selection or the survival of the fittest rule. The number of instances of a particle in the next generation is determined by the formula

$$K = \frac{N.p_i}{\sum_j p_j} \tag{1}$$

The particles with higher p values are more likely to be selected and copied to the new sample set. As a result of this, the particles will eventually move to the locations where the robot is more likely to be located at. In the second step of resampling, particles in the new sample set are moved according to their probabilities. The amount of movement for a particle instance is determined with the formula

$$x_i^{T+1} = x_i^T.(1 - p_i^T).Rnd(-1, 1).\Delta_{trans} \quad (2)$$

$$y_i^{T+1} = y_i^T.(1 - p_i^T).Rnd(-1, 1).\Delta_{trans} \quad (3)$$

$$\Theta_i^{T+1} = \Theta_i^T.(1 - p_i^T).Rnd(-1, 1).\Delta_{rot} \quad (4)$$

where, Rnd returns a uniform random number in the range $[-1, 1]$ and Δ_{trans} and Δ_{rot} are translational and rotational constants, respectively. The amount of movement is inversely proportional to the probability so the particles with higher probabilities will have a smaller move than particles with small probabilities.

2.2 Vision-Based Monte Carlo Localization

Many variations of MCL exist in the literature but most of these are implemented on wheeled robots equipped with distance sensors such as laser range finders or sonar sensors. In RoboCup Four-Legged League, the only sensor that the robot can use for localization is the color camera mounted on its nose. There are uniquely bi-colored landmarks placed around the field so the robot can use the information about the distances and orientations to these landmarks to localize itself. Using vision data obtained from a single camera is more challenging in some aspects. The main disadvantage of vision data is its narrowness compared to distance sensor arrays in which most of them have a panoramic field of view of 360 degrees. On the other hand, odometry is more noisy in legged robots than those in wheeled robots. Narrowness of field of view prevents the robot of seeing a high number of landmarks at a time, so it is hard to extract useful information from single visionary sensor reading. A few implementations in which the special circumstances mentioned above were taken into account have been proposed for the legged robots. The motion update phase is same as the generic approach. Each time a motion update is received, the particles are moved according to the relative displacement information received. In the observation update phase, we need an observation model for obtaining the $P(s, l)$. Visual feedback obtained from the sensors consist of a set of visual percepts seen at that time. Relative distance and bearing of the percept from the robot's egocentric origin is provided and by using this information, $P(s, l)$ can be computed as follows

Let v be a visual percept and $V = (v_0, v_1, ., v_{n-1})$ be the set of visual percepts seen in a time step. Then the probability of having V in location l which is $P(V, l)$ is

$$P(V, l) = \prod_{i=1}^{N} F(v_i, e_i) \quad (5)$$

where v_i is the observed bearing and distance for percept i and e_i is the expected distance and bearing of object seen in the percept i assuming that the robot is located at l. In

some aspects, F works as a similarity metric that maps the difference between expected and observed properties of an object of interest to a value in the range $[0, 1]$. In our implementation, F consists of two parts: $anglesimilarity$ and $distancesimilarity$. For each processed camera frame, our vision system provides us a collection of seen percepts with their relative bearing in radians and relative distance in millimeters from the robot's egocentric origin.

$$F(v_i, e_i) = F_{angle}(v_i, e_i).F_{dist}(v_i, e_i) \tag{6}$$

$$F_{angle}(v_i, e_i) = \frac{1}{1 + e^{(-40.\Delta_{bearing}.0.875)}} \tag{7}$$

$$\Delta_{bearing} = \frac{|\Theta^i_{observed} - \Theta^i_{expected}|}{\pi} \tag{8}$$

$$F_{dist}(v_i, e_i) = \frac{1}{1 + e^{(-20.\Delta_{dist}.0.75)}} \tag{9}$$

$$\Delta_{dist} = \frac{|min\{Dist^i_{observed}, Dist^i_{expected}\}|}{|max\{Dist^i_{observed}, Dist^i_{expected}\}|} \tag{10}$$

The idea behind using a sigmoid function in similarity calculation is favoring poses with small differences between observed and expected values and punishing the poses with high deviation from the expected values.

3 Robot Soccer Domain

The RoboCup organization is an international initiative which aims to build a soccer team of fully autonomous humanoid robots beating the last human world soccer champion by the year 2050 [9].Sony four-legged league is one of the subdivisions of the RoboCup in which two teams each consisting of four Sony AIBO robotic dogs compete against each other. Game area is 6m by 4m and four unique bi-colored beacons are placed in order to provide information for localization. Robots are fully autonomous so any human intervention other than placing robots on the field, any off-board computation and providing external data such as image from overhead camera is prohibited. Field setup can be seen in the Figure 1.

4 Proposed Extensions

4.1 Considering Number of Percepts Seen

The number of landmarks used in the localization process has an important role in determining the actual position and orientation of the robot accurately. The accuracy of the estimated position and orientation of the robot increases with the increasing number of landmarks seen in a single frame.

When calculating the confidence on each particle, each landmark contributes to the confidence by its angle and distance similarity. However, this approach results in an

Fig. 1. Field setup for RoboCup Legged League

undesired output as the number of landmarks increases. For example, seeing a single landmark having a probability of 0.75 seems to provide a better estimation than four landmarks each having 0.9 probability which results in 0.9 x 0.9 x 0.9 x 0.9 = 0.6561 confidence. In order to avoid this misinterpretation, confidence is calculated in such a way that increasing number of landmarks increases the confidence. The formula used for calculating the confidence is

$$confidence = p^{(5-N_{percepts})} \tag{11}$$

where, $N_{percepts}$ is the number of the percepts seen

Since the maximum number of landmarks that can be seen in a single frame is 4, $p^{5-4} = p$ assigns the current value of the probability to the confidence which is the highest possible value.

4.2 Using Inter-percept Distance

Using the landmark distance and orientation information in vision update phase works fine under normal conditions. However, all these calculations depend on the assumption of noiseless information, which is nearly impossible for applications with real robots. Introducing noise to the calculations with two static objects will conclude to a geometric space of possible solutions and multiple configurations instead of a single position and a single configuration in the environment. The number of possible solutions increases as

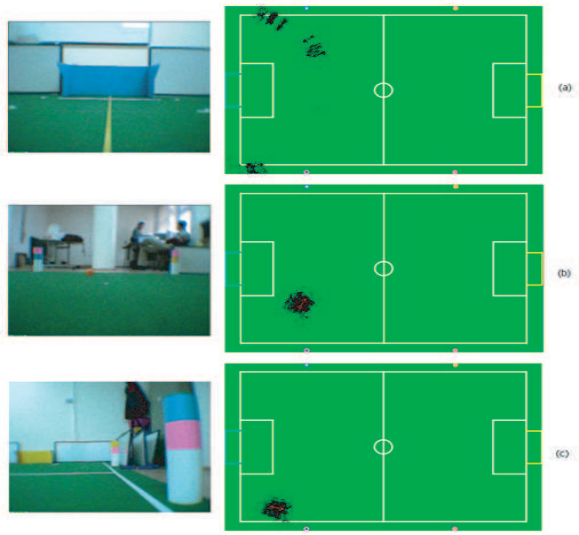

Fig. 2. Possible positions and orientations of the robot for (a) one landmark, (b) two landmarks, and (c) three landmarks

the confidence decreases, i.e. the gray area in Figure 3 (b) grows. At one extreme point, when the confidence is complete, the gray area converges to a single point as shown in Figure 3 (a). In the opposite case, when the confidence is zero, the robot is lost and gray area covers the environment. In case of seeing only one landmark, the normal MCL update rules are applied. But, it is not so rare to see more than one landmark (two beacons or one goal and one beacon) at a time. In such case, using inter-percept distance provides more accurate estimation than using percept information individually.

When the robot perceives two static objects, it uses the widths and/or heights of these objects to calculate the relative distances.

In the Figure 4 w_1 and w_2 are the perceived width (in pixels) of the static objects. Similarly h_1 and h_2 are the heights of the static objects. The distances d_1 and d_2 can be calculated from these values by using a special function. This special function is calculated by fitting one polynomial or partial polynomials on the top of the experimental data. When the distances are known, the calculation of the orientation is relatively simple. The angles α and β can be used to find the orientation of the robot. Under ideal conditions, where there is no noise and the vision is perfect, d_1, d_2, α and β values are enough to find the current configuration of the robot. However, there is

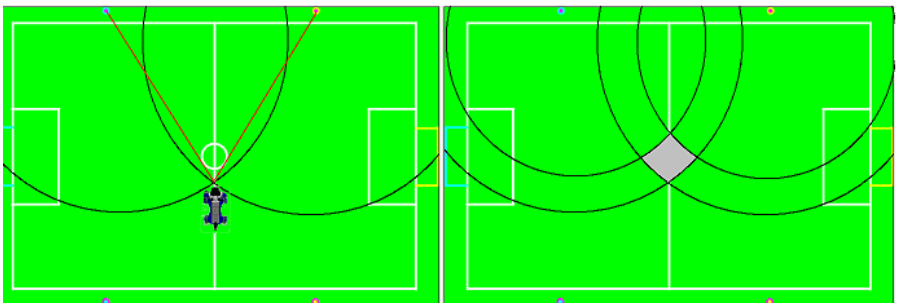

Fig. 3. Localization (a) without noise, (b) with noise

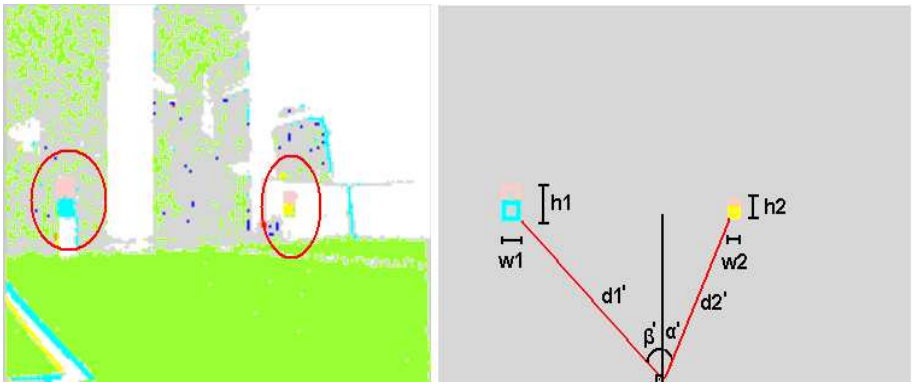

Fig. 4. Relative distances and orientations. (a) Classified image. (b) Recognized static objects

noise and it affects the distance calculations dramatically. In our case, where the resolution is 176x144, two-pixel error in the width of the static objects causes less than one-centimeter distance error for near objects. But, for far objects, two-pixel error may cause up to 50 cm distance error.

At this point, we used another measure, which is the distance between the perceived static objects. This measure both reduces the sensitivity to noise and provides additional information for localization.

In Monte Carlo Localization (MCL), a population of candidate configurations is used to converge to the actual position of the robot. During the iteration process, a probability value for each configuration is calculated. The configurations with higher probabilities are selected more frequently for the next iteration. The calculation of these probabilites are given in Section 2.2. However, to use the distance between the static objects, more complicated equations are required. Because of perspective, the difference of estimated and expected distances cannot be used directly.

As shown in Figure 5 the estimated distance should be compared with $a + b$, but not d_3. As an example, suppose that the static objects are at (s_x^1, s_y^1) and (s_x^2, s_y^2). For a

given robot configuration (x, y, Θ), the following equations are used to calculate a and b.

$$d_1 = \sqrt{(sx_1 - x)^2 + (sy_1 - y)^2} \tag{12}$$

$$d_2 = \sqrt{(sx_2 - x)^2 + (sy_2 - y)^2} \tag{13}$$

$$\alpha = arcTan(\frac{sy_1 - y}{sx_1 - x}) - \theta \tag{14}$$

$$\beta = arcTan(\frac{sy_2 - y}{sx_2 - x}) - \theta \tag{15}$$

$$s_d = cos(\beta).d_2, d_1 > d_2; cos(\alpha).d_1, o/w \tag{16}$$

$$a = tan(\alpha).s_d \tag{17}$$

$$b = tan(\beta).s_d \tag{18}$$

Where s_d is the scene distance, which is the projection line (or plane in 3D) distance to the robot. Furthermore, $(a + b)$ is the distance between the two static objects on the projection plane at s_d.

After those calculations, we have two values to compare. The first one is the distance between two static objects, calculated from the captured image, which is in pixels. The other one is $(a + b)$ which is in mm. In our work, we used the ratio of each distance to the width of the image, instead of, converting the units. The first ratio is trivial.

$$visionRatio = \frac{d_s}{w_{img}} \tag{19}$$

where, d_s is the distance between static objects and, w_{img} is the captured image width

However the second ratio depends again on s_d. In addition, horizontal field of view (FoV) of the camera is also used to calculate the width of the projection line or plane. For AIBO ERS-210's horizontal FoV is 57.6 degrees.

$$expectedRatio = \frac{(a + b)}{tan(FoV/2).2.s_d} \tag{20}$$

And finally the affect of the distance between the static objects to the overall configuration probability is calculated as follows,

$$p_d = \frac{1}{1 + e^{(-40.\Delta_{ratio}.0.875)}} \tag{21}$$

$$\Delta_{ratio} = \frac{|min\{visionRatio, expectedRatio\}|}{|max\{visionRatio, expectedRatio\}|} \tag{22}$$

At the end of each iteration of MCL, for each configuration these p_d values, which are calculated from distances, orientations and other possible sources, are multiplied to find the final confidence of the configuration.

In our work, we use a two-dimensional representation of the environment instead of a three-dimensional one. Since the effects of rotation of the camera and the orientation of the robot are handled in the object recognition subsystem, this representation works well. However, we assume that the heights of the static objects and the robot are equal. In addition, we only consider the horizontal difference while estimating the distance between static objects.

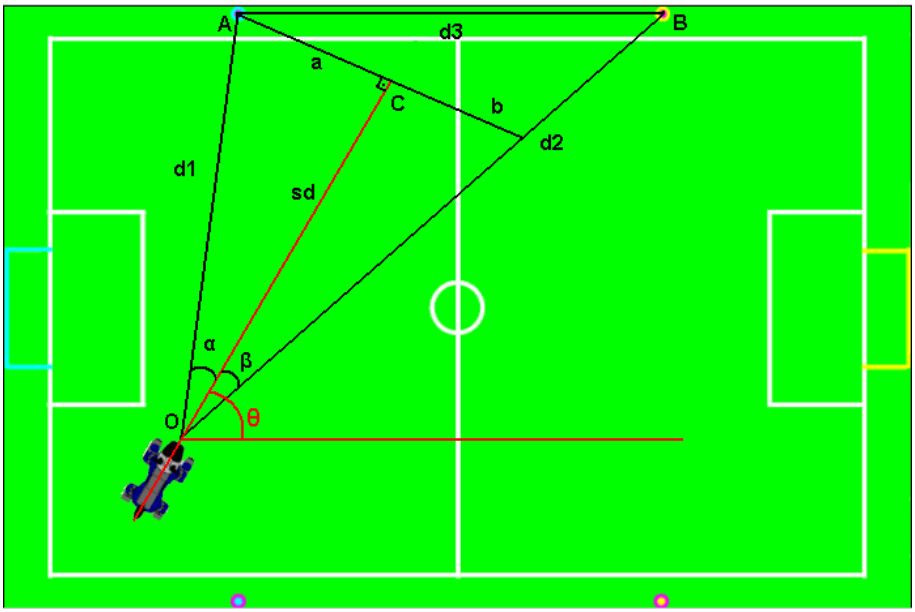

Fig. 5. Calculation of Distance Projection

4.3 Variable-Size Number of Particles

In MCL the number of particles, which are candidate configurations for current position, is one of the most important parameters. If the unnecessarily large amount of particles used, the process slows down dramatically.

On the other hand, if the number of particles is too small, the system converges to a configuration very slowly, or cannot converge at all. One possible solution is to fix the number of particles to a constant for which the processing speed is moderate and the localization converges in a reasonable amount. But in this scenario, when the localization starts to converge, which means the candidate configurations are similar to each other, most of the processing is unnecessary.

In our localization system, the number of particles is assigned to the maximum number of particles allowed. This maximum number is the lowest particle count for

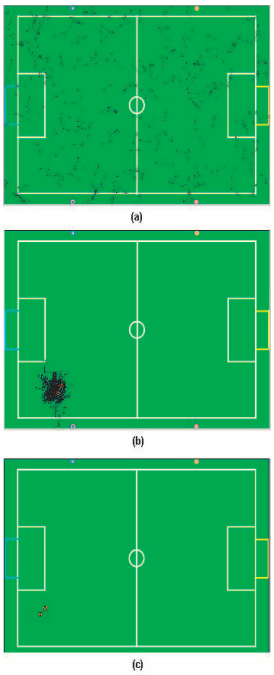

Fig. 6. Number of particles changes while (a) searching, (b) converging and (c) converged

which the localization system converges in a reasonable time, for nearly all cases. But still it is a large number. During the execution of the localization system, the number of particles is reduced if the confidence about the position of the robot increases. Which means, the system searches fewer configurations if it is certain about its position, as shown in Figure 6.

Similarly, when the confidence about the current pose of the robot decreases, the number of particles increases, which means the search for a better configuration speeds up. This oscillation continues if the confidence for the current pose is above a specific constant. Otherwise, which means the robot is lost, the confidence is set to zero and the entire process is restarted.

The overall process can be modeled by the following equation,

$$N_{par} = K.p, p > T; K, o/w \tag{23}$$

Where N_{par} is the number of particles, which are candidate configurations, K is the maximum number of particles and p is the confidence of the current pose. Finally, T is the threshold value for reseting the system to the lost state.

4.4 Dynamic Window Size Based on Pose Confidence for Resampling

In the earlier approaches, when the confidence on the current position decreases below a threshold value, the positions and orientations of each particle are reset and the particles are distributed over the entire field randomly. However, each reset operation requires processing a large number of particles over a large area. In order to solve this problem, a window, in which the particles will be distributed, is constructed around the representative particle. The size of this window is inversely proportional with the confidence value of the representative particle, and the number of particles that will be distributed in this window is directly proportional to the size of the window. That is, when the confidence on the representative is high, the window constructed around the representative and the number of particles that will be used is small. If a reset operation fails to include particles having significant probability values, the size and the number of particles that will be used in the next reset operation are gradually increased. This approach provides a faster convergence since a smaller number of particles in a smaller area are processed. Figure 7 illustrates the situation in which the robot is kidnapped from the blue goal into the yellow goal. Convergence is seen after 5 frames, and after 11 frames the robot finds its position and orientation.

5 Results

To test the efficiency of the extensions, we have performed two different experiments. In both experiments, extended MCL is compared with the original implementation. To test both the converge time and the estimation error, the robot is placed to the upper left corner of the field in order to provide enough number of visual percepts. Localization process is terminated if 95 percent confidence is achieved or the process iterates 200 times. The iteration counts for converging to a point and distance error of that point to the actual position are given in Table 1.

According to the results, extended MCL reduces the iteration count to converge a configuration and the error of this configuration nearly by 50 percent. Since the standard MCL fails to converge for some iteration, the iteration count is high than extended MCL in average.

In the second experiment, we tested the convergence speed of the original and extended implementations in case of kidnapping. The robot is moved from where the localization system is converged to a point farther away and the re-convergence time is logged. The results can be seen in Table 2.

6 Conclusions

Autonomous self localization is one of the key problems in mobile robotics research and have been addressed many times with proposed many different approaches. Robot soccer is a good test bed for many aspects of the mobile robotics research such as multi-agent systems, computer vision, self localization and effective locomotion with its highly dynamic and partially observable nature.

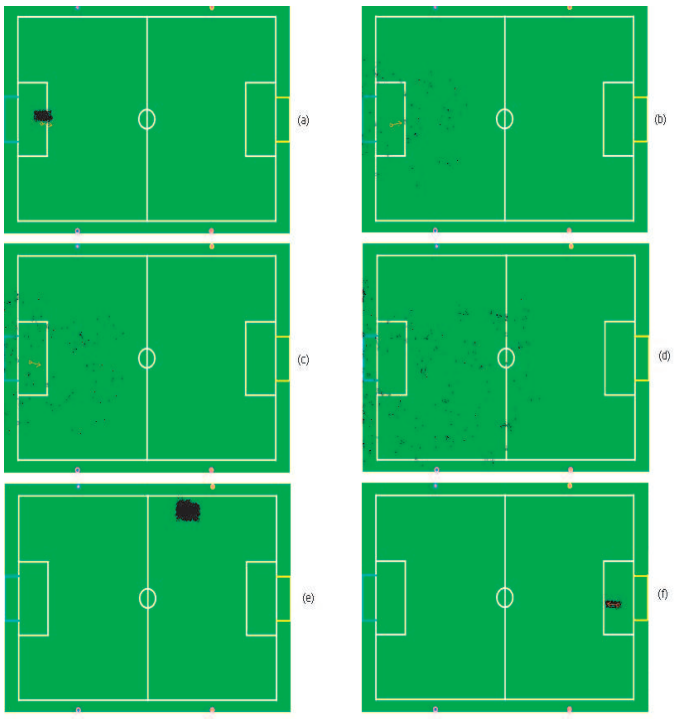

Fig. 7. (a) Just after being kidnapped, (b) first frame after kidnapping, (c) after 2 frames, (d) after 3 frames, (e) after 4 frames, and (f) after 11 frames the robot finds its position

In this paper, we proposed several novel extensions to the vision based Monte Carlo Localization for legged robots playing soccer. MCL is a sample based statistical method for estimating the robot's current pose in the environment by representing the probability distribution of being at a certain pose as a set of samples drawn from that distribution. The probabilities are updated through a motion model which models the movement of the robot in terms of relative displacements and observation model which models the likelihood of receiving a certain observation from a certain pose. Although the idea of both MCL and our extensions are straightforward, there should be some extensions to the core idea in order to handle limited and noisy sensory information, unreliable motion models and some extreme conditions such as the so-called kidnapping problem.

In this work, we have proposed four practical extensions to the vision-based MCL for legged robots. Using variable number of particles is not a new approach, but our implementation has no extra computational requirement as the other implementations (i.e. determining the number of particles proportional to the variance of confidences in sample set). Using inter-percept distance in addition to distances and bearings to the percepts is a novel approach and the results are quite satisfactory. Also, considering the number of percepts seen while calculating the pose confidence is a novel approach

Table 1. Convergence time and error ratios for Extended MCL vs. Standart MCL

Extended MCL		**Standart MCL**	
Iter. Count	Distance Error (cm)	Iter. Count	Distance Error (cm)
12	6.96	10	17.72
121	6.06	200	28.98
52	21.28	10	7.43
11	12.54	10	13.54
11	2.83	16	5.39
13	7.70	17	23.58
50	18.96	15	20.70
16	18.28	200	33.84
16	8.39	200	35.11
77	5.28	20	18.44
Averages			
37.9	10.83	69.8	20.47

Table 2. Re-convergence time in case of kidnapping for Extended MCL vs. Standart MCL

Extended MCL		**Standart MCL**	
Kidnapping Distance (mm)	Number of Frames	Kidnapping Distance (mm)	Number of Frames
$\tilde{3}550$	26	$\tilde{3}550$	35
$\tilde{2}280$	9	$\tilde{2}280$	15
$\tilde{1}760$	11	$\tilde{1}760$	39

and allows the observations with high number of percepts have higher effect on the confidence, in other words, the more number of percepts seen, the more reliable the observation is. Again, using a subset of the state space for resampling when our belief about the current pose is over a threshold is not a new idea but the our way of window size and position determination for resampling is novel.

Using proposed extensions allowed the system to converge rapidly without affecting from an occasional bad sensor reading while maintaining enough elasticity that allows a fast convergence in case of kidnapping.

7 Acknowledgments

This project is supported by Boğaziçi University Research Fund project 03A101D.

References

1. S. Thrun, D. Fox, W. Burgard, and F. Dellaert, ”Robust Monte Carlo Localization for Mobile Robots”, Journal of Artificial Intelligence, 2001.

2. S. Thrun, ”Particle Filters in Robotics”, In The 17th Annual Conference on Uncertainty in AI (UAI), 2002.
3. D. Fox, W. Burgard, H. Kruppa, and S. Thrun, ”Markov Localization for Mobile Robots in Dynamic Environments” Journal of Artificial Intelligence, 11, 1999.
4. D. Fox, ”Adapting the Sample Size in Particle Filters Through KLD-Sampling” International Journal of Robotics Research, 2003.
5. C. Kwok and D. Fox, ”Map-Based Multiple Model Tracking of a Moving Object” In The International RoboCup Symposium, Lisbon, 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. 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.
8. H. L. Akin, et al., ”Cerberus 2003” Robocup 2003: Robot Soccer World Cup VII, The 2003 International Robocup Symposium Pre-Proceedings, June 24-25, 2003, Padova, pp.448.
9. Robocup Organization ”http://www.robocup.org”