

Comparison of Localization Methods for a Robot Soccer Team

Hatice Kose; Buluc Celik & H. Levent Akin

Department of Computer Engineering,
Bogazici University,
34342 Bebek, Istanbul, Turkey
kose@boun.edu.tr

Abstract: In this work, several localization algorithms that are designed and implemented for Cerberus'05 Robot Soccer Team are analyzed and compared. These algorithms are used for global localization of autonomous mobile agents in the robotic soccer domain, to overcome the uncertainty in the sensors, environment and the motion model. The algorithms are Reverse Monte Carlo Localization (R-MCL), Simple Localization (S-Loc) and Sensor Resetting Localization (SRL). R-MCL is a hybrid method based on both Markov Localization (ML) and Monte Carlo Localization (MCL) where the ML module finds the region where the robot should be and MCL predicts the geometrical location with high precision by selecting samples in this region. S-Loc is another localization method where just one sample per percept is drawn, for global localization. Within this method another novel method My Environment (ME) is designed to hold the history and overcome the lack of information due to the drastically decrease in the number of samples in S-Loc. ME together with S-Loc is used in the Technical Challenges in Robocup 2005 and play an important role in ranking the First Place in the Challenges. In this work, these methods together with SRL, which is a widely used successful localization algorithm, are tested with both offline and real-time tests. First they are tested on a challenging data set that is used by many researches and compared in terms of error rate against different levels of noise, and sparsity. Besides time required recovering from kidnapping and speed of the methods are tested and compared. Then their performances are tested with real-time tests with scenarios like the ones in the Technical Challenges in ROBOCUP. The main aim is to find the best method which is very robust and fast and requires less computational power and memory compared to similar approaches and is accurate enough for high level decision making which is vital for robot soccer.

Keywords: MCL, ML, robot soccer, self-localization

1. Introduction

The global localization problem is the estimation of the position of a robot relative to the environment, using its actions and sensor readings. In challenging real-time test beds like Robot Soccer with four legged robots, the sensors and the environment are uncertain, and the results are typically erroneous and inaccurate. Many solutions to the localization problem including geometric calculations which do not consider uncertainty and statistical solutions which cope with uncertainty by applying sophisticated models have been proposed (Burgard, W., et al, 1996; Thrun, S., et al, 2001; Gutmann, J.S., & Fox D., 2002). The nature and complexity of typical environments have a profound effect on the results proposed by these algorithms. Generally, solutions producing precise are rather slow and have a high memory usage. Fast solutions, however, typically produce only coarse results. Although some approaches like the Kalman filter produce precise local results, they fail to find the global position. For this reason, localization is still a nontrivial and challenging problem. In robot soccer, there are typically distinguishable unique landmarks in the field which can be used by a robot to find its own location. This information can be used next

to find the location of the ball and goal. In such real-time application with robots limited by on board computational resources, fast solutions with less memory and computational resources are especially demanded.

This work is a part of the Cerberus Team Robot soccer project (Cerberus, 2005), and aims to localize the legged robots in the soccer field globally, while solving the above mentioned problems. We compare an improved version of the previously developed hybrid approach called *Reverse Monte Carlo Localization* (R-MCL) (Kose, H., & Akin, H. L., 2004; Kose, H., & Akin, H. L., 2005a; Kose, H., & Akin, H. L., 2005b) combining the ML and MCL methods with a new localization method. This method consists of a module called *My Environment* (ME) which stores the perceptual data and provides a filtered and more robust input to the interested modules of the robot, and a new localization technique called *Simple Localization* (S-Loc), which is a sample based localization technique where only one sample is used for each perception data (Celik, B., 2005).

These methods that are specially designed and implemented to be used in Robot Soccer Games and Technical Challenges by Cerberus Team, are compared against first in an offline manner with SRL on the data set

in (Gutmann, J.S., & Fox D., 2002) then online on the new field.

The organization of the paper is as follows: In the second section, a brief survey of localization methods is presented. In the third section detailed information about the proposed algorithms R-MCL, S-Loc and ME environment can be found. The results of the application of the test set to the proposed approaches are given in section four. In the fifth section, conclusions and suggestions for future work are given.

2. Localization Methods

Triangulation is the simplest localization method that depends on the range and bearing data and uses geometry to compute a single point that is closest to the current location. But in real world applications a robot can never know where it is exactly because of the uncertainty in its sensors, and the environment. Consequently, several different approaches that estimate the position of robot probabilistically were introduced to integrate this uncertainty into the solutions.

Kalman filter (Kalman-Bucy filter) is a well-known approach for this problem. This filter integrates uncertainty into computations by making the assumption of Gaussian distributions to represent all densities including positions, odometric and sensory measurements. Since only one pose hypothesis can be represented, the method is unable to make global localization, and cannot recover from total localization failures (Gutmann, J.S., & Fox D., 2002; Stroupe, A.W., et al, 2002).

Many works consider Markov localization (ML) (Burgard, W., et al, 1996; Fox, D., et al, 1999). ML is similar to the Kalman filter approach, but it does not make a Gaussian distribution assumption and allows any kind of distribution to be used. Although this feature makes this approach flexible, it adds a computational overhead.

Monte Carlo Localization (MCL) is a version of Markov localization that relies on sample-based representation and the sampling/importance re-sampling algorithm for belief propagation (Schulz, D., & Burgard, W., 2001; Thrun, S., et al, 2001). Odometric and sensory updates are similar to ML. Most of the MCL based works suffer from the kidnapping problem, since this approach collapses when the current estimate does not fit observations. There are several extensions to MCL that solve this problem by adding random samples at each iteration. Some of these methods are Sensor Resetting Localization (SRL), Mixture MCL (Mix-MCL), and Adaptive MCL (A-MCL). In SRL, when the likelihood of the current observation is below a threshold, a small fraction of uniformly distributed random samples is added (Lenser, S., & Veloso, M., 2000). Mix-MCL additionally weights these samples with current probability density. This method has been developed for extremely accurate sensor information (Gutmann, J.S., & Fox D., 2002). Adaptive

MCL only adds samples when the difference between short-term estimate (slow changing noise level in the environment and the sensors) and the long-term estimate (rapid changes in the likelihood due to a position failure) is above a threshold (Gutmann, J.S., & Fox D., 2002). The MHL method discussed in (Kristensen S., & Jensfelt P., 2003) aims to avoid caused by using a single Gaussian, by using a mixture of Gaussians enabling the representation of any given probability distribution of the robot pose.

ML-EKF method is a hybrid method aiming to make use of the advantages of both methods, taking into consideration the fact that ML is more robust and EKF is more accurate (Gutmann, J.S., & Fox D., 2002).

Although there have been only a few fuzzy logic based approaches, they appear to be promising (Buschka, P., et al, 2000; Kose et al, 2003; Kose, H., & Akin, H. L., 2005b). In these approaches, the uncertainty in sensor readings (distance and heading to beacons) is represented by fuzzy sets.

3. R-MCL Method

As mentioned in section 2, ML is robust and converges fast, but is coarse and computationally complex. On the other hand, sample based MCL is not as computationally complex as ML, and gives accurate results. However, it cannot converge to a position as fast as ML, especially in the case of an external impact on the position of the robot (such as kidnapping). In addition, the number of samples to be used is generally kept very high to cover all the space and converge to the right position. Several extensions have been made for adaptive sample size usage, but these still do not solve the slow coverage problem.

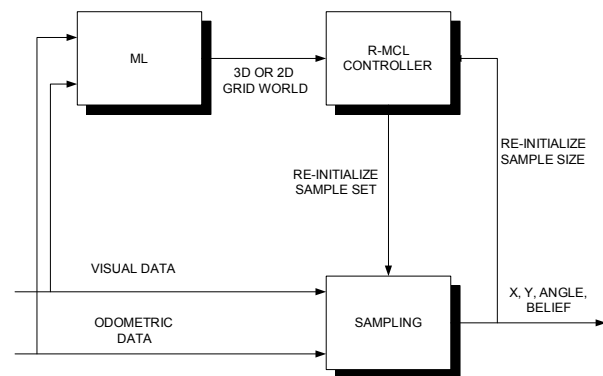


Fig. 1. The R-MCL working schema

The Reverse Monte Carlo Localization algorithm (Kose, H., & Akin, H. L., 2005a; Kose, H., & Akin, H. L., 2005b) was developed to benefit from the advantages of these two methods while avoiding their disadvantages. The idea is to converge to several cells by ML or another grid based method, and then produces a limited number of samples inside this bulk of grids to find the final position. The average of these samples would give the final

position and the standard deviation might give the uncertainty of the final position as in the MCL based methods. In the original MCL, the number of samples is increased to decrease the bias in the result. In R-MCL since we converge by selecting the cells with maximum probability, so the bias is already decreased. The R-MCL algorithm is given in Fig. 1.

In this work, some improvements were done on the R-MCL method and in particular the ML module. In the modified ML, not only the distance but also the bearing information is used to find the best grids, so the number of chosen grids decreases and confidence increases. When the samples are drawn, also the best samples are selected using distance and the bearing from these cells, and their average is returned as the current pose. Note that samples are taken into consideration only when the position is above a certainty level, in other words the number of chosen cells are below a limit (e. g. 50), and there is at least one cell which is below or equal to the minimum error in both distance and bearing limitations. Also if there are no samples which satisfy the minimum bearing and distance error condition then the results of ML are used instead. The bearing of the new pose is found by the ML module inside the R-MCL because it is more accurate and robust.

Algorithm 1 The R-MCL Algorithm

```

1: if bool_ML==TRUE then
2:   ML_update
3:   if ML_number_of_grid_cells_in_max_grid_array < ThML then
4:     MCL_init(ML_samples)
5:     bool_ML=FALSE
6:   end if
7: else
8:   MCL_update
9:   MCL_init(ML_samples)
10:  if MCL_lost()==TRUE then
11:    ML_reset()
12:    bool_ML=TRUE
13:  end if
14: end if

```

The R-MCL algorithm is given in Algorithm 1. Here, *bool_ML* is a *Boolean* variable indicating whether to call ML or not. If it is *TRUE*, ML is active otherwise MCL module is active. *ML_reset*() function initializes all grid cells with equal probability that adds up to one. ML update works like the normal ML sensory and action update as stated in the previous sections.

After each step, the cells with non-zero probability are put into a data structure called *max_grid_array*. If the number of cells in this array is smaller than a threshold called *th_{ML}*, this means that the cells have converged to a coarse location in the field. Then *MCL_init*() function is called with the samples generated by ML as input. Instead of generating random samples, we produce smaller-sized grid cells from the chosen non-zero weighted grid cells¹. Then the weights for each sub cell, which is a member of the sample set of MCL now, are calculated using the sensory and action readings, and the

¹ The size of the sub cells can also be made adaptive.

normal MCL updates are done as stated in the previous section. Since *bool_ML* becomes *FALSE*, ML is not called anymore, and MCL update function is used. If the uncertainty in MCL is below a threshold *Th_{MCL}*, this means either the cumulative error increased so much that the robot feels totally lost, or it is kidnapped. Then *MCL_lost*() function returns *TRUE*, so *ML_reset*() function which initializes the variables of ML module, such as grid cell weights and *max_grid_array* is called. In addition, *bool_ML* becomes *TRUE*, so in the next step ML module would be active.

4. My Environment

My Environment (ME) is a buffering module which provides pose estimates using perception and odometry data. It stands between the perception module and the other modules that use the output of the perception module.

In robotic applications, the perceptions are typically imprecise, inadequate and even it is possible to have false identification of objects. ME keeps track of the positions of the static and dynamic objects in the environment relative to the robot. After each observation, these data are updated, filtered, and the effects of noise and false identifications are eliminated as much as possible. So it keeps a history of the perceptions. Because the movement of a dynamic object distorts its position history, the more recent perceptions are weighted more than that of static objects.

Buffering the poses of the static objects is very valuable for localization in two ways. First, the processed static object poses would be more robust and lead to more accurate agent pose estimations. Secondly, this buffering will make it possible to process more static objects than that are seen at any moment, as long as the buffered relative poses could give a proper estimate for the current pose of the static object.

The pose estimations and updates may seem costly, but this cost is overwhelmed by the benefits of the system easily, since ME module provides more data than the current perception which are also more stable and robust than the raw perception. Besides ME is a stand alone module that is not just designed to be used with a specific algorithm. It can be used with any localization algorithm, after some minor arrangements.

5. Simple Localization

S-Loc is a new localization technique which is based on the weighted average of temporary pose samples (Celik, B., 2005). In this method, for each perceived landmark, a sample pose is calculated according to that perception and the previous pose estimate of the agent as shown in Fig. 2, where d_1 is the distance that the robot should have to the processed landmark if it was at the processed pose, θ_2 is the angle that is calculated assuming that the robot is at the position indicated by the processed pose and the landmark is at the orientation that is perceived, and d_2 is

the perceived distance to the processed landmark, and the arrow shows the calculated pose estimate position.

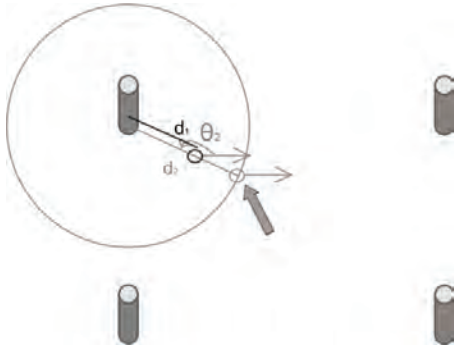


Fig. 2. S-Loc pose calculation

Algorithm 2 The S-Loc Algorithm

```

1: if no landmark perceived then
2:   use the old pose estimate reducing its confidence by a constant factor
3: end if
4: Set the sample poses
5: Add first sample pose for the old pose estimate
6: Take the position of the old pose estimate
7: Calculate the Euclidian differences of the perceived landmarks' positions
   and the position of the processed sample pose.
8: Using a fusion formula, the differences found in 7, and the confidence of
   the processed landmark; calculate likelihood for the first sample pose.
9: for each perception do
10:  Using the old pose estimate, calculate the angle of the new pose sample
     assuming the robot is at the old pose estimate position and the processed
     landmark's perceived relative orientation is correct.
11:  Assuming that the angle of the new pose calculated in 10 and the pro-
     cessed landmark's perceived distance is correct; the position of the new
     pose sample is calculated.
12:  Calculate the Euclidian differences of the perceived landmarks' positions
     and the position of the processed sample pose.
13:  Using a fusion formula, the differences found in 12, and the confidence
     of the processed landmark; calculate likelihood for the new sample pose.
14: end for
15: Calculate the weights for sample poses
16: for each sample pose do
17:   The weight of the processed sample is directly a function of its likelihood
18: end for
19: Calculate the new pose estimate
20: Take the weighted average of the positions of the sample poses.
21: Take the weighted average of the result in 20 and the old pose estimate
     to provide memory.
22: Calculate the orientation of the new pose estimate
23: for each perceived landmark do
24:   Calculate the orientation that satisfies the perception
25: end for
26: Take the weighted average of the calculated orientations in a radial man-
     ner, where the weights are the confidences of the corresponding percep-
     tions.

```

The detailed description of S-Loc is given in Algorithm 2. Assuming that the robot has the previous pose position, the orientation of the sample pose is calculated so that the perception would be on the correct direction. Then, the position of the sample pose is calculated on a line through the old position and along with the same direction assuming that the perception was exact. These temporary pose samples are calculated for each perceived landmark. In addition to them, the previous pose is also used as an additional sample.

For each sample pose, likelihood is calculated. Assuming that the agent's actual pose is the sample pose being

processed, the Euclidian difference of the perceived landmarks' positions and their actual positions is calculated. Together with the confidence of the perception from which that sample pose is calculated, these differences are used in the calculation of the likelihood of that sample pose. For the sample that is copied from the previous pose, instead of the confidence of the perception, confidence of the previous pose is used. The likelihoods of the sample poses are used for calculating their weights, and a new pose position is calculated as the weighted average of these sample poses' positions. That weighted average pose position is then used together with the previous pose estimate's position to calculate the current pose estimate's position. The purpose of not using the weighted average pose directly is to provide memory in order to prevent fluctuations of the pose estimate and make it more stable. After the current position of the agent is estimated, the current orientation of the agent is calculated using the current position estimation and the perceptions.

In the case of having no perception at a certain time, decreasing the confidence of the previous pose estimate, the current pose estimate could be obtained.

The odometry update process is as simple as updating the pose estimation with the odometry data. As only the pose estimation is used from the previous perception update, no more update or calculation is necessary.

In a way, S-Loc works similar to the MCL as the sample poses are used in the same way they are used in MCL. The main difference is the selection of these sample poses. In MCL, there is a large number of pose samples, and they are populated according to their confidences, and randomly mutated for small changes. In S-Loc, new pose samples are calculated at every estimation, and for each perceived landmark a pose sample is calculated. This way S-Loc becomes a much lower cost, i.e. much faster, localization method with an accurate pose estimation capability.

The memory used in the S-Loc increases the robustness of the system even further and the big jumps of the pose estimate are prevented.

6. Test Environment

In this study, we first compared the methods R-MCL, S-Loc, and SRL which were implemented to be used by our soccer team, by using a well-known data set in an offline manner. This enables us to compare the performance of these methods with the other outstanding methods in the same domain, tested by the same data set (Gutmann, J.S., & Fox D., 2002; Kristensen and Jensfelt, 2003). The results of the tests that belong to the previous versions of the R-MCL could be found in (Kose, H., & Akin, H. L., 2005; Kose, H., & Akin, H. L., 2006).

Then we tested the methods on the real robot, based on a challenging scenario, under varying light conditions. These real-time tests were implemented using the current field conditions to test the success and robustness of the

method in the real environment, since simulation results may underestimate the problems and bugs related to real life conditions.

6.1. The Test Field

This work is a part of the Cerberus Team Robot soccer project (Cerberus'05, 2005), and aims to localize the legged robots in the soccer field globally within the rules of Sony Legged Robot League. The Sony Legged Robot League is an international robot competition that has been launched within the RoboCup initiative (Robocup Organization, 2005; Sony Four-Legged Robot League, 2005). In robot soccer, teams of robots, that are capable of seeing and moving, play matches against each other for fixed time periods, and the team with the highest goal score win the match like in real soccer. In order to do this, the player robots must detect their location, the goals, the ball, the members of their team and the opponent team members (optional for high level planning), and place the ball in the opponent team's goal to score a goal. A robot is typically expected to find its own location using the artificial landmarks called *beacons* in the field, and then use this information to find the location of the ball and goal. So localization is a vital problem for robot soccer.

There are several limitations and assumptions related to the rules of the Robocup. The locations of the beacons are given as prior information to the robots. The robots could identify the beacons and estimate their relative distance to these beacons using their cameras with some noise due to the imperfect vision system, motion of the robot during perception, and dynamic nature of the world. With a well-developed locomotion module, also odometric data is available to the robot. Consequently, the robot could calculate to where and how much it has moved during a specific time interval. When this information is not available, a motion model may be used for estimation.

The testing environment is the soccer field which has standards specified by the Robocup legged robot league technical committee. The data used in the tests were collected using the field specifications of 2001. The background is green with white strips showing the half line and penalty line, and the covering walls are white. There are six artificial landmarks called beacons which are uniquely colored and have predefined position, size and colors to aid the robot to localize itself. There are two goals. One of them is blue, and the other yellow.

Recently, the size of the field was enlarged, which allows working with larger number of robots, but adds more uncertainty to the observations since now the landmarks are further away from the robots and it is harder to observe the distance to them accurately as the distance increases. In addition, two of the beacons and the sidewalls are removed, to make the field look like the real soccer fields. These facts make the problem more challenging.

6.2. Sony AIBOS



Fig. 3. SONY AIBOs: ERS 7 and ERS-210

The algorithms are tested on the ERS 210 type robot dog AIBO that is a commercial product of SONY (Sony Four-Legged Robot League, 2005). These are quadruped legged robots which have an onboard CMOS camera with angle of view 57.6 degrees in horizontal, and 47.8 degrees in vertical axis. The resolution of the camera is 176 x 144 pixels. The robots also have an IR sensor which works in the range 10-90 cm, and wireless LAN cards that allow wireless communication between the robots. Robots are dark gray with blue and red uniforms, which allow the other robots to detect them and decide whether they are opponents or teammates. Unfortunately detecting the ID of robots is not trivial, therefore can not be implemented and is not currently used in localization.

Since 2004, the new product ERS 7 which has higher computational and physical power is used by most of the soccer teams (Fig. 3. (Celik, B., 2005)).

7. Tests and Results

7.1. Offline experiments

In the testing phase, a standard set of data is used. These data are based on the records of the test runs of Sony's ERS 210 quadruped robots (AIBO) on the Robocup soccer field, used in (Gutmann, J.S., & Fox D., 2002) for the comparison of several well-known localization methods in literature. These are produced by running the robot on the field as shown in Fig. 4 on a figure of eight like path for almost an hour, stopping the robot on several predefined points called markers, and recording the observations and odometry readings during this run. The tests aim to analyze accuracy and robustness in case of noisy and sparse data, and the ability to handle kidnapping problem.

- **Noisy Data test:** In the noisy data tests, the percentage of noise is increased from 0 to 80 by increments of 10 points in every data set. The number of noisy samples is increased to evaluate the robustness and accuracy of observed methods in case of high noise levels.
- **Sparse Data Test:** In sparse data tests, samples are deleted from the tests in a predefined sequence. The sparsity increases from 1/1 to 1/256, by 1/2ⁿ sparsity

increases in every data set. As the frequency increases, the behavior of the selected methods is observed.

- **Kidnapping Test:** The kidnapping problem is tested, by changing the position of the robot (beyond the robots awareness), and the time for recovery is recorded.

Lastly, the time required for processing one frame is measured.

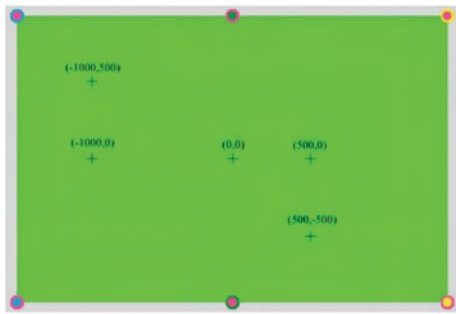


Fig. 4. The soccer field of the test environment

The parameter set were chosen after detailed tests and comparisons. In R-MCL, the cell size is chosen as 5 cm in the referenced works, whereas it is taken as 30 cm in the current work, to increase the speed. Triangulation method which is used in case of observing two or more landmarks is also not used in our implementations. Although smaller cell size and use of triangulation is expected to decrease the error rate considerably, the current approach is taken since it is more realistic and similar to the real world case.

In Fig. 5 and Fig. 6, the results for sparse and noisy data are presented, respectively. The error rates of the tests are calculated based on the expected location of robot when it reaches a marker, and its exact location. Note that, there are also unavoidable errors in the exact locations of robot due to experimental problems reported by the data providers. When the results of the tests are compared to the similar tests in (Gutmann, J.S., & Fox D., 2002; Kristensen and Jensfelt, 2003), the methods show similar performance in the sparse and noisy data tests. In the noisy data tests, the R-MCL outperforms the other methods. The ML module inside R-MCL keeps the method robust even under high levels of noise.

In the sparse data tests, S-Loc outperforms the others with the help of its ME module which provides extra filtered information. The performance of the other methods against sparsity could be increased by using ME within them.

During the design and implementation of the S-Loc algorithm, some observations are made on the actual robots and the data provided to the localization module in the actual application domain, which is Cerberus'05. It was noticed that the perception noise in the relative angle of the landmarks is much less than the noise in the distance. The main problem with the perception was the

noise in the distance estimation. Occasionally, false perceptions were also present.

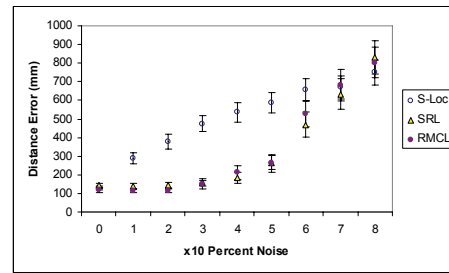


Fig. 5. Results of the noisy data tests

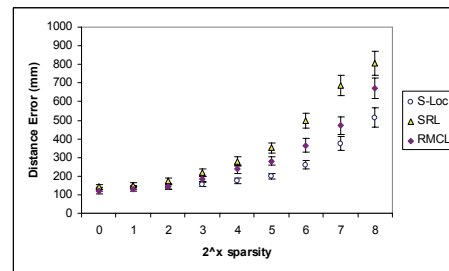


Fig. 6. Results of the sparsity tests

It was also observed that by using the ME module, the effect of infrequent false perceptions could be decreased. They are generally not a big problem for the ME module, since not only the last perception but a window of the last perceptions are used to calculate the output.

Since the false perceptions are relatively rare, and having the relative angle perceptions the most reliable source of information about the environment, the S-Loc algorithm is optimized on the actual robots for depending on the relative angle data more than others.

In the way noisy data sets are prepared for the experimental study, only the false perception of the beacons is modeled, which constitutes only a small part of the noise problem and occurs infrequently (Gutmann, J.S., & Fox D., 2002). Noise in the perception of the distance and the relative angle of the landmarks are not modeled. This way, only the robustness of the algorithms with respect to the false perceptions is evaluated.

In the experiment 3, the ability of the methods to solve the kidnapped robot problem is analyzed. To do so, the average time the methods needed for re-localizing the robot after it has been manually displaced is computed over 22 kidnapping tests. The results are shown in Fig. 7, where the results are in seconds. According to the experimental results, R-MCL recovers the earliest from kidnapping. It is followed by SRL and S-Loc.

As S-Loc is a geometric method, where a large number of samples are not distributed over the field, the average time for recovering from kidnapping is longer than others, but it is still in an acceptable interval for robotic soccer domain.

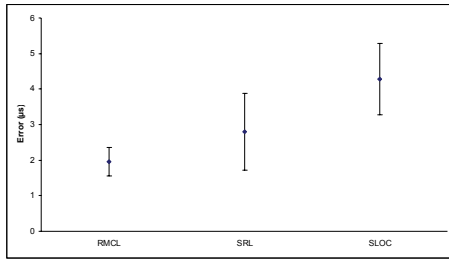


Fig. 7. Results of the experiment 3

In the experiment 4, which is carried on an AMD Athlon XP 1800+ 1.54 GHz computer with 1.00 GB of RAM, average processing time of the methods are tested. The average processing times and the number of processed frames per second are calculated for each method over the complete raw data set that contains 51523 frames ten times. The vision and motion data come from data set, so only the processing times of localization algorithms are measured, and the processing time required for motion, vision and other modules is excluded from the tests. For SRL and R-MCL, the processing time interval is the average of each frame's time intervals for the motion and vision updates for localization module; whereas for S-Loc, the time intervals include motion and vision updates for the ME module as well. As is shown in Table 1, where results are in microseconds, the total processing time for S-Loc and ME is less than R-MCL, and both are much less than SRL. The high speed of the proposed system is a very important characteristic especially for real-time systems.

	S-Loc	SRL	R-MCL
Processing Time	149 µs	930 µs	383 µs

Table 1. Results of the experiment 4

7.2. Real-time experiments

Besides the offline tests, to observe the performance of the methods while running on the robot, in real environment, the methods were tested with several real-time tests. The field used in these tests is the new field which has four beacons, and enlarged by almost 1.5 times the old field used in the offline tests. Also the white walls around the field were removed, so that the robot could detect any object in the lab around the field and might be confused. Besides the color table of the robot is an older one which was trained under different lighting conditions, and the lighting conditions of the test field varied due to day light which could not be avoided.

Unlike the offline tests, besides the beacons, also the goal areas, and white field lines were perceived to be used by the localization methods. The field lines are not used by our methods yet. But the goal areas are used for correcting orientation.

The scenario of the tests was taken from a localization challenge of Robocup games. There are six markers on the field, four of them placed near by the beacons and

two on the mid field line, and the robot should visit all of these markers in a predefined sequence.



Fig. 8. Real Time Test field with markers

As seen in Fig. 8, robot follows a rectangular path while visiting markers shown by '+' sign on the field. At each time, the robot is started from the center of the field. Whenever its distance to the current marker in the visiting list is below a threshold (10 cm in the current tests), and it is confident enough, the robot stops and wags its tail. During the tests, while the robot is following its path, it becomes kidnapped many times when it goes out of the field and is placed on the nearest position in the field. Also it has been kidnapped and placed on the opposite side of the field many times on purpose to measure its robustness against kidnapping. These facts besides problems due to the manual measurement and data collection increased the overall error unavoidably. Each test is performed ten times and the average and standard deviation of the error in distance is calculated. Since the robot stops for every marker and is started manually again, it was not suitable to measure the time. So, only the error in distance is measured in these tests.

We ran the four algorithms S-Loc, SRL and two versions of R-MCL, according to this scenario. All of the parameters of R-MCL1 and R-MCL2 are the same with the offline tests, except *th_{ML}* which is 15 in the offline tests and R-MCL1, but 30 in R-MCL2. Since the field is larger, the field is represented by more cells now, so it is logical to increase the *th_{ML}* to improve the success. This also increases the number of samples used in the MCL part, but still keeps them in the acceptable range. In the tests, SRL could not converge to the points in the limited time duration so its results are not taken into consideration. In these tests, R-MCL outperforms S-Loc and another version of itself, as seen in Fig. 9.

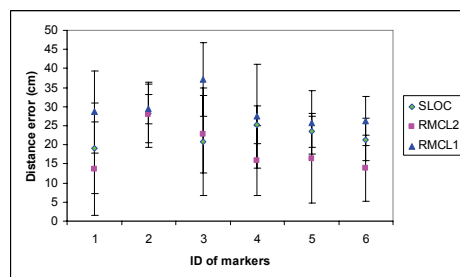


Fig. 9. Results of the real time tests with six markers

8. Conclusions

Localization in a totally unknown area is a very hard problem for autonomous mobile robots. The solution should be successful in environments like the Robocup Games and the Technical Challenges which require very high accuracy and speed, should be robust against noise, and insufficient data, and should recover fast from kidnapping. For this reason previously a hybrid method called R-MCL method was developed. In this work, a new novel system consisting of a localization method S-Loc and environment ME which holds and filters the perceptions for S-Loc are introduced. Both methods are very robust and fast and require less computational power and memory compared to similar approaches and they are accurate enough for high level decision making which is vital for robot soccer. They both perform well in the tests and keep in the acceptable range when compared with the results of other methods in (Gutmann, J.S., & Fox D., 2002; Kristensen and Jensfelt, 2003). They outperform each other in different cases. R-MCL outperforms S-Loc in the noisy environments, due to the robustness of its ML module. S-Loc outperforms R-MCL in case of high sparsity with the help of its history based module ME. S-Loc requires less processing power since it uses a negligible amount of samples compared to the particle filter based methods, and it overcomes the lack of information problem with its "smart" history module ME. It is also very successful in case of high sparsity although it uses a small numbers of samples. But since the window size of the ME is high it detects and recovers from kidnapping slower than R-MCL. It can be improved by using an adaptive sized history in ME. R-MCL is robust and fast in case of kidnapping but it uses fixed parameters, so it is outperformed by adaptive methods in case of high sparsity and noise. Since it does not keep a history due to big cell sizes in ML module, its performance decreases in very high sparsity levels. These could be improved by using a history based module and adaptive parameters due to changing levels of noise and sparsity.

Acknowledgements

This work was supported by Bogazici University Research Fund project 03A101D. We thank Stephen Gutmann and Dieter Fox for kindly providing the test data, and their endless help when needed.

8. References

Burgard, W., Fox, D., Hennig, D. & Schmidt, T., (1996) "Estimating the Absolute Position of a Mobile Robot Using Position Probability Grids", Institut für Informatik III., Universität Bonn, Proc. of the Fourteenth National Conference on Artificial Intelligence (AAAI-96), Vol. 2, pp. 896-901

Buschka, P., Saffiotti, A., & Wasik, Z. (2000) "Fuzzy Landmark-Based Localization for a Legged Robot"

Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS) Takamatsu, Japan, pp. 1205-1210.

Celik, B., (2005) "S-LOC and MY ENVIRONMENT: A New Localization System For Autonomous Robots", Master Thesis, Bogazici University.

Cerberus'05 Team Web Site (2005), Available from: <http://robot.cmpe.boun.edu.tr/aibo/home.php3>.

Fox, D., Burgard, W., & Thrun, S., (1999) "Markov Localization for Mobile Robots in Dynamic Environments", Journal of Artificial Intelligence Research 11, pp. 391-427.

Gutmann, J.S., & Fox D., (2002) "An Experimental Comparison of Localization Methods Continued", In Proc. of the 2002 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'02), pp. 454-459.

Kose, H., Bayhan S. & Akin, H. L., (2003) "A Fuzzy Approach to Global Localization in the Robot Soccer Domain", IJCI Proceedings of International XII Turkish Symposium on Artificial Intelligence and Neural Networks TAINN 2003 ISSN 1304-2386 Vol:1, No:1 pp.1-7.

Kose, H., & Akin, H. L., (2004) "Experimental Analysis and Comparison of Reverse-Monte Carlo Self-Localization Method", Proceedings of CLAWAR/EURON Workshop on Robots in Entertainment, Leisure and Hobby, Vienna, Austria, December 2 - 4, pp. 85-90

Kose, H., & Akin, H. L., (2005) "Robots from nowhere", RoboCup 2004: Robot Soccer World Cup VIII, D. Nardi, M. Riedmiller, C. Sammut, (Eds.), LNCS, Vol. 3276, pp.594-601.

Kose, H., & Akin, H. L., (2006) "A Fuzzy Touch to R-MCL Localization Algorithm", RoboCup 2005: Robot Soccer World Cup IX, A. Bredendfeld, A. Jacoff, I. Noda, Y. Takahashi (Eds.) LNCS Vol. 4020, pp. 420 - 427.

Kristensen S., & Jensfelt P., (2003) "An Experimental Comparison of Localisation Methods, the MHL Sessions", Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'03), pp.992-997.

Lenser, S., & Veloso, M., (2000) "Sensor Resetting Localization for Poorly Modeled Mobile Robots" Proceedings. ICRA '00. vol.2, pp. 1225-1232.

Robocup Organization, (2005), Available from: <http://www.robocup.org/>.

Schulz, D., & Burgard, W., (2001) "Probabilistic State Estimation of Dynamic Objects with a Moving Mobile Robot", Robotics and Autonomous Systems 34, pp. 107-115.

Sony Four-Legged Robot League, (2005), Available from: <http://www.tzi.de/4legged/>.

Stroupe, A.W., Sikorski, K., & Balch, T., (2002) "Constraint-Based Landmark Localization." Proceedings of 2002 RoboCup Symposium.

Thrun, S., Fox, D., Burgard, W., & Dellaert, F., (2001) "Robust Monte Carlo Localization for Mobile Robots", Artificial Intelligence, 128, pp. 99-141.