Bearing-Only Localization for Mobile Robots

Matthias Jüngel
Humboldt-Universität zu Berlin
AI Lab, Unter den Linden 6
D-10099 Berlin, Germany
juengel@informatik.hu-berlin.de

***Abstract*— In this paper we describe a new localization-method. It is a bearing-only approach which only uses horizontal bearings to landmarks and incorporates odometry. The approach is perfectly suited for mobile robots equipped with a camera because bearings can be extracted from images with high accuracy. The method itself does not need any internal representation of the robot's position which is updated by alternating motion and sensor updates. The position is calculated directly using a short term memory of the observations. Our approach is also able to generate template positions for Monte-Carlo localization. The distribution of the template positions reflects the accuracy of the position calculation which depends on the configuration of the selected landmarks. In this paper we give a detailed description of the method and show the results of experiments conducted on a Sony Aibo robot demonstrating the precision.**

I. Introduction

Localization is one of the most important challenges for a mobile robot. There are a lot of researchers developing new methods each year. In the last years the Monte-Carlo Localization has been the standard approach to the localization problem. A lot of improvements have been suggested to overcome limitations in the processing power and to address the limited angle of view of robots that are not equipped with omni-vision.

There are a lot of suggested improvements to the sensor model. Sensor-resetting reseeds new position templates obtained from observations [5] and there are improvements that build a short-time history of observations to create more accurate position templates [9]. Other approaches try to incorporate negative information [3]. A lot of improvements has also been suggested for the motion model - for example using the detection of collisions.

This work was motivated by the experiences we collected with the localization method that we use in RoboCup for our Aibo robots. We use a standard Monte-Carlo Localization as described in [2], [1], [10]. This method performs very well on robots equipped with infra-red distance sensors or laser range finders. Things become harder when the main sensor is a camera. Distance measurements can be inaccurate due to partial occlusions of landmarks. Additionally size-based distance measurements have a large error when the objects are too far away or the resolution of the camera is low.

In this paper we provide a bearing-only method for localization that incorporates odometry and can be used as a template generator for Monte-Carlo Localization. Section II describes the method in detail. Section III describes the experiments we performed with our Aibo robots.

II. Bearing-only Localization Using Odometry

In this section we show a method that allows a robot to localize based on two inputs. The first input are observations. The vector

$$\vec{\alpha} = \begin{pmatrix} \alpha_{l_1} \\ \alpha_{l_2} \\ \dots \\ \alpha_{l_n} \end{pmatrix}$$

contains the measured bearings to the landmarks $l_1, l_2, ..., l_n$. These angles were measured at different times $t_1, t_2, ..., t_n$. The second input is the knowledge about the motion of the robot. The vector

$$\vec{u} = \begin{pmatrix} u_1 \\ u_2 \\ \dots \\ u_n \end{pmatrix}$$

contains the robot's odometry at times $t_1, t_2, ..., t_n$.

A robot can obtain these vectors $\vec{\alpha}$ and $\vec{u}$ by storing its observations and the according odometry in a buffer. Figure 1 shows a visualization of such a buffer.

In this section we define a function $F(x, y, \vec{\alpha}, \vec{u})$ which describes the likelihood for the robot of being at position (x,y) on the field. This function can be used to calculate a robot position (the maximum of the function) or to generate templates for Monte-Carlo Localization.

A. Localization with three simultaneously seen horizontal bearings

In this subsection we show two methods to determine the position of the robot when the robot is not moving. The first one uses well-known simple geometry, the second one is a constraint-based approach.

1) Using simple geometry: When a robot perceives three landmarks without moving between the observations, the calculation of the position is straightforward. With the known position of the landmarks a circle can be constructed for each pair of bearings. The radius of the circle is determined by the difference of the angles and the distance between the landmarks. The intersection point of the circles is the only possible position for the robot. Figure 2 shows an example.

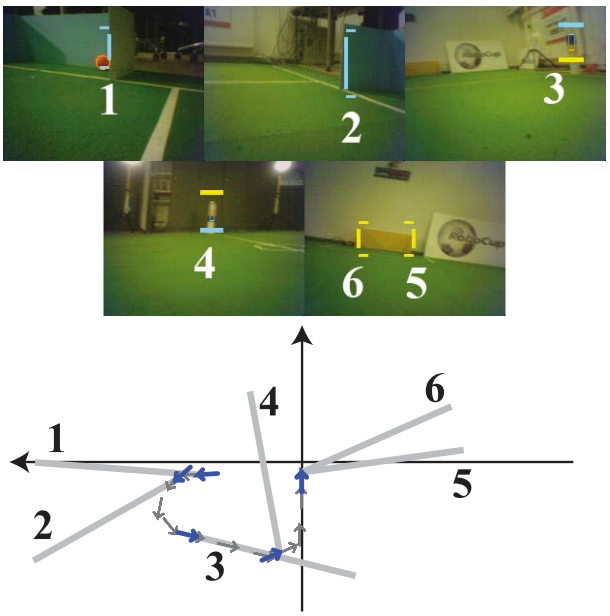

Fig. 1. Odometry and horizontal bearings. Top: Five images with six horizontal bearings (1: right goal post, 2: left goal post, 3 and 4: center landmarks, 5 and 6 goal posts) Bottom: Gray arrows show the robot's odometry at different times, bold arrows show the odometry associated with the horizontal bearings.

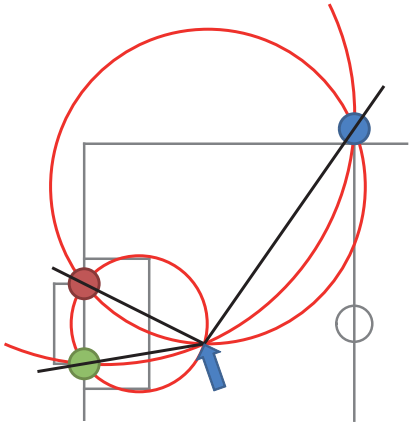

Fig. 2. Determining the position of the robot by three horizontal bearings to known landmarks. Black lines: horizontal bearings. Large circles: circles given by two bearings and two landmarks.

2) Pose estimation using angular constraints: When the position is determined by intersecting circles, there is nothing known about the influence of errors in the measurement of the bearings. This influence can be determined using a constraint-based approach. A single observation of a landmark l at a certain relative angle constrains the angle ϑ_l the robot can have at a certain position (x, y) on the field. This angle is given by

$$\vartheta_l(x, y, \alpha_l, x_l, y_l) = \arctan\left(\frac{y_l - y}{x_l - x}\right) - \alpha_l$$

where (x_l, y_l) is the position of the landmark on the field and α_l is the relative angle to the landmark. Figure 3 a) shows this function. When two bearings to two landmarks are given, the function

$$D_{l_1,l_2}(x, y) = (\vartheta_{l_1}(x, y) - \vartheta_{l_2}(x, y))^2$$

describes the likelihood for being at position (x, y). Figures 3 b,c,d) show function D for several examples. The shape of the function represents how good a certain pair of landmarks is suited to constrain the position on the field. For example a plateau in this function (like in figure 3 b) means that a small error in an observation leads to a large error in the resulting position.

The function $D_{l_1,l_2}(x, y)$ introduced above describes for each position (x, y) how good the angles ϑ_{l_1} and ϑ_{l_2} obtained from two different horizontal bearings match. To use more than two observations $\alpha_{l_1}, \alpha_{l_2}, ..., \alpha_{l_n}$, we can calculate the average angle of all resulting $\vartheta_{l_1}, \vartheta_{l_2}, ..., \vartheta_{l_n}$ for each position (x, y) using this formula

$$\vartheta_{average}(x, y) = \arctan\left(\frac{\sum_{i=1}^{n} \sin(\vartheta_{l_i}(x, y))}{\sum_{i=1}^{n} \cos(\vartheta_{l_i}(x, y))}\right)$$

Figure 4 a) shows function $\vartheta_l(x, y)$ for three different landmarks and the resulting average angle. Using $\vartheta_{average}(x, y)$ we can define the function

$$G(x, y) = \sum_{i=1}^{n} (\vartheta_{average}(x, y) - \vartheta_{l_i}(x, y))^2$$

which describes how similar the angles ϑ_l are. This function has its maximum at the position (x,y) that best fits with all observations $\alpha_{l_1}, \alpha_{l_2}, ..., \alpha_{l_n}$. Furthermore the function provides an estimation of the position error for known errors in the observation. Figure 4 b) shows this function for three observations.

B. Incorporating odometry

To incorporate odometry we define a function $\upsilon_l(x, y, \alpha_l, \Delta_{odometry_l}, x_l, y_l)$ which determines the angle of the robot at position (x, y) when the landmark l was seen at angle α_l and the robot moved $\Delta_{odometry}(\Delta_x, \Delta_y, \Delta_\phi)$ since the observation. Figure 5a) illustrates these parameters and the resulting angle υ_l. To determine υ_l we define a triangle with its corners at the position (x_l, y_l) of the landmark l (angle β), at the position (x, y) (angle γ) and at the position (x_0, y_0) where the observation was taken (angle δ). Figure 5b) shows this triangle. Note that in this triangle (x_l, y_l) and (x, y) are fixed. The position of (x_0, y_0) can be calculated using the angle ω from (x, y) to (x_l, y_l) and the distance Δ_d the robot walked:

$$\begin{aligned} x_0 &= x + \cos(\omega + \gamma) \cdot \Delta_d \\ y_0 &= y + \sin(\omega + \gamma) \cdot \Delta_d \end{aligned}$$

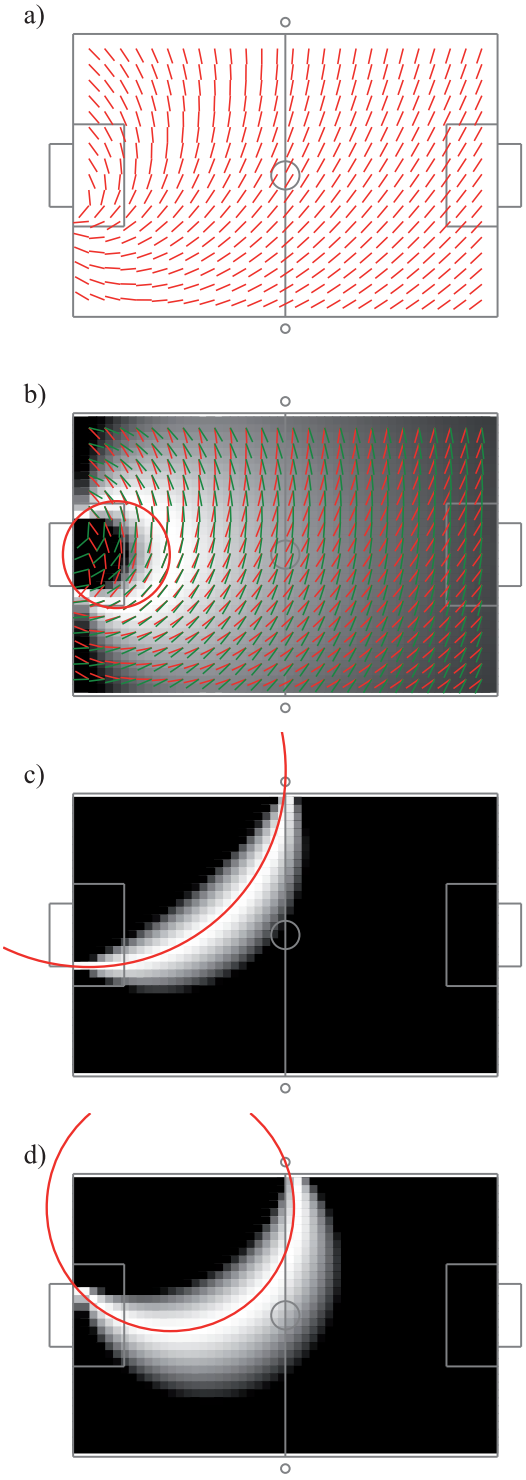

Fig. 3. Angular constraints for location estimation. a) The red lines show the angle ϑ_{l_1} on the field for each position. This angle results from a given horizontal bearing to the left goal post. b) The green lines show the angle ϑ_{l_2} on the field that results from a bearing to the right goal post. The gray-scale grid in the background shows the square of the difference $(\vartheta_{l_1} - \vartheta_{l_2})^2$ between the angles. This function has a peak near the positions that are covered by the circle obtained using simple geometry. c) The difference $(\vartheta_{l_1} - \vartheta_{l_3})^2$ between the angles resulting from the bearings to the left goal post and the right center flag. d) The difference $(\vartheta_{l_2} - \vartheta_{l_3})^2$ between the angles resulting from the bearings to the right goal post and the right center flag.

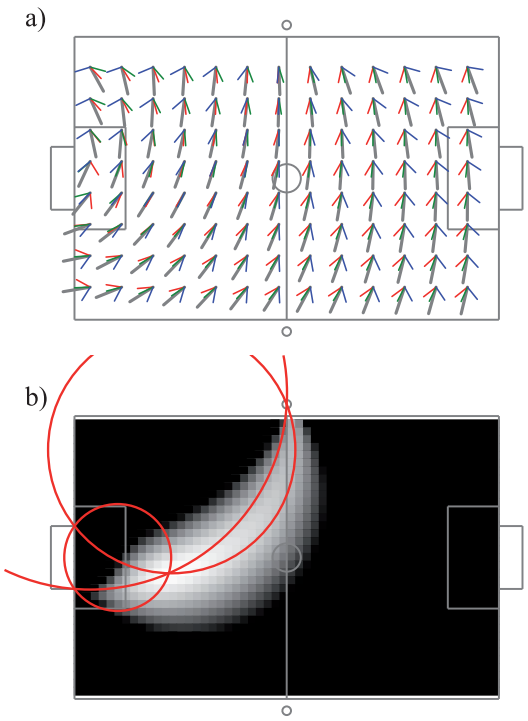

Fig. 4. Similarity of angles. a) The thin lines show the angles $\vartheta_l(x, y)$ for three different observations. The bold line shows the average angle. The robot's position is constrained to the positions where the angles are similar. b) Function $G(x, y)$ displayed as height map. White: small difference between the angles, black: large difference between the angles. The red circles are obtained from the method using simple geometry described above.

where γ follows using sine rule:

$$\begin{aligned}\gamma &= \pi - \delta - \beta \\ &= \pi - \alpha_l - \arctan\left(\frac{\Delta_y}{\Delta_x}\right) - \arcsin\left(\frac{\Delta_d \cdot \sin(\delta)}{d_l}\right).\end{aligned}$$ [1]

With the known position (x_0, y_0) follows

$$\upsilon_l(x, y, \alpha_l, \Delta_{odometry_l}, x_l, y_l) = \vartheta(x_0, y_0) + \Delta_\phi.$$

When the robot is at position (x, y), has seen the landmark l at angle α_l some time ago, and has moved by $\Delta_{odmetry_l}$ since that observation, the function υ_l gives the angle the robot must have. Similar to the function G from section II-A we define a function

$$F(x, y, \vec{\alpha}, \vec{u}) = \sum_{i=1}^{n} \left(\upsilon_{average}(x, y) - \upsilon_{l_i}(x, y)\right)^2$$

which describes the likelihood of the robot for being at position (x, y). This function can incorporate an arbitrary number of observations from the past and does not need any internal representation of the position that is updated by alternating sensor and motion updates. The selected sensor information $\vec{\alpha}$ and the according motion information $\vec{u}$ are processed at once.

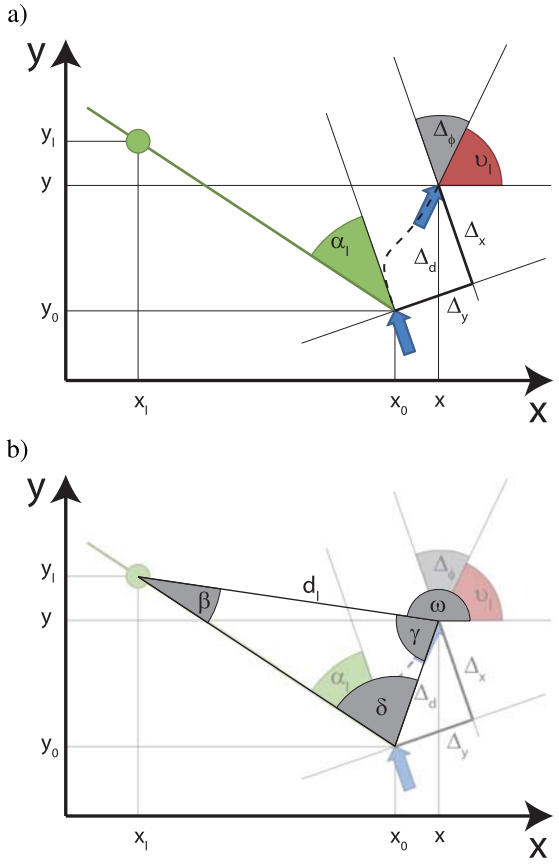

Fig. 5. Geometrical considerations to determine the angle υ_l of the robot.

C. Calculating the robot pose

The maximum of function F given in the last section is the position of the robot. The rotation of the robot can immediately be calculated using $\upsilon_{average}$ or the angle υ_{l_0} that is defined by the last observation. When a fast and rough estimation of the robot pose is wanted, the maximum can be determined by an iteration through the domain of the function. When a more accurate estimation is wanted it can be obtained by means of standard methods as Gradient Descent with only a few iterations. Note that such methods usually find only local maxima of the function.

D. Generating templates for Monte-Carlo localization

Often there is more information than the horizontal bearings to unique landmarks to determine the pose of the robot. Especially when there is ambiguous information like distances to walls or field lines a localization method that is able to track multiple hypotheses might be preferred. In such a case the function F described in section II-B can be used to create template poses for sensor resetting. Which is in particular useful when only a small number of particles can be used due to computational limitations. To obtain a fixed number of samples you can normalize F such that all values are between 0 and 1:

$$F' := \frac{1}{(1+F^2)}$$

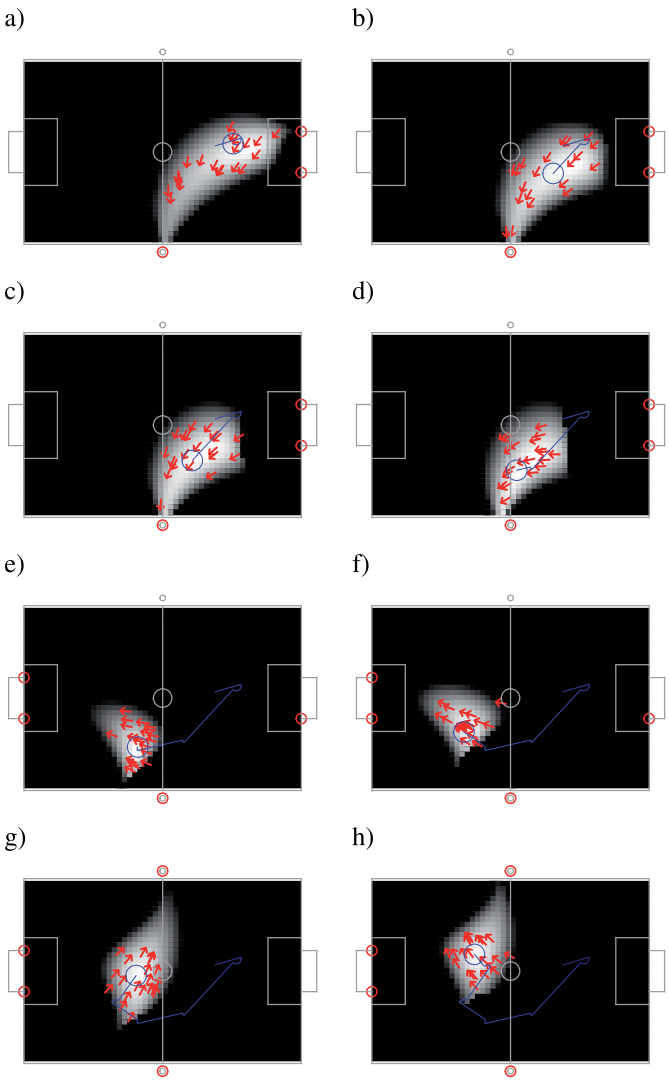

Fig. 6. The Function $F(x, y)$: white - high likelihood, black - low likelihood, Arrows: position templates that can be used for sensor resetting in Monte-Carlo localization - note that usually only a small number of these templates will be used. Small circles: the landmarks that were used for position calculation. Large circle: the robot pose (known from the simulation). Path: the way the robot walked.

and create a template pose at each position (x, y) with $random() < F'(x, y)^n$. Where n is a parameter to adjust how much the sample poses can deviate from the maximum. Figure 6 shows templates obtained from function F.

III. Experimental Results

In this section we describe the setup and the results of the experiments. We used an Aibo ERS-7 robot built by Sony for our experiments. This robot has a pan-tilt camera with a resolution of 208x160 pixels. All tests were done on a RoboCup soccer field (size: 6m x 4m) where the goal posts of the two goals and two beacons at the half way line can be used as landmarks. The horizontal bearings to the landmarks needed for our location approach were extracted from images and joint sensor data by the method given in [4]. One experiment shows

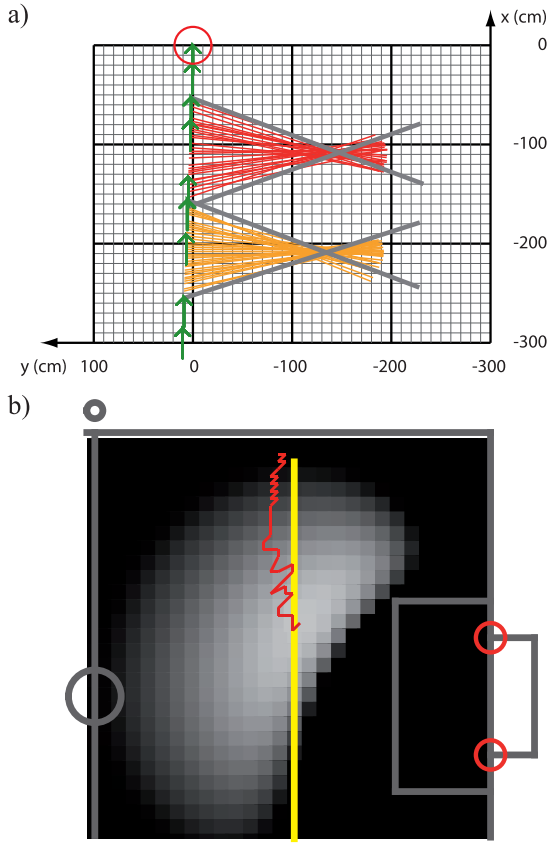

Fig. 7. a) Circle: current relative robot position. Arrows: history of odometry. Thin lines: bearings to right and left goal post. Bold lines: bearings used for localization. b) Background: function $F(x, y)$ when the left goal post was seen. Bold line: the path the robot walked. Thin line: the result of the localization (since possible). Circles: the goal posts used for localization.

that the method equips the robot with something like stereo vision. In another experiment we analyzed the performance of our system.

A. Localizing using only two landmarks

In this experiment the robot uses bearings to only two landmarks, which is not enough information to localize when odometry is not incorporated. We let the robot walk over a RoboCup field from one side line to the other, observing one of the goals. During this walk, it first observes just the right goal post, then both posts, and in the end only the left post. As soon as the left goal post is seen, there is enough information to localize using the first and the last observation of both goal posts. Figure 7 a) shows a visualization of the robot's motion and its observations. Figure 7 b) show a plot of function $F(x, y)$ at the first time the left goal post is seen and the resulting path for the rest of the run. With this experiment we were able to show that incorporating odometry is a benefit for bearing-only localization. Only two beacons are sufficient to determine the position of the robot when the robot changes its observing position.

B. Generating template poses for monte-carlo localization

We developed the bearing-only localization approach as a replacement of the distance-based sample template generation that we use for our Monte-Carlo self-localization [6], [8], [7]. The old method was not usable any longer as with the 2007 rule change in the RoboCup Sony Four Legged league two more beacons were removed and thus there are less beacons and the beacons have a higher average distance to the robots.

Thus we added the method described in section II-B as a sample template generator in a way described in section II-D. The particle filter uses 200 particles.

To measure the quality of our improvements we steered a real robot via remote control over the soccer field in our lab performing an s-like shape on the field. The head of the robot performed the typical Aibo scan motion which looks around searching for the ball and the landmarks. During this process log data was recorded containing camera images, head joint values, odometry data, and ground truth robot positions obtained by a ceiling mounted camera. Such log-files can be played back off-line to feed our algorithms with data. We used the recorded log data to compare different parameterizations of the approach.

We compared three localization methods:

1) The implementation of Monte-Carlo localization we use in RoboCup (not using sample templates)
2) The method described in section II-C using the maximum of $F(x, y)$
3) Monte-Carlo localization using sample templates as described in section II-D.

To compare the ground truth and the determined robot position we calculated the average distance for the whole run in the log-file for each of the methods. The result was that the relative error (compared to field size) of the plain Monte-Carlo method was 9.13%, the error of the maximum-method 5.13%, and the error of the Monte-Carlo method using sample templates 3,18% to 3,74% (depending on the number of samples).

Figure 8 shows a visualization of the path the robot walked and the pathes obtained by our methods. The influence of the number of samples used for reseeding is given in table 9.

The overall result of the experiments is that without template generation there were random jumps and a large deviation from the ground truth robot pose. With sample template generation (using one sample per frame) the resulting trajectories were smoother and closer to the ground truth.

IV. Conclusion

In this paper we presented an approach for bearing-only self-localization incorporating odometry. It does not depend on distance measurements which often are inaccurate when obtained by the size of far-distant landmarks. Our approach works on a buffer of observations and the history of the robot's odometry. We can estimate the position of the robot directly by this data without alternating sensor and motion updates.

However, we showed that our method also provides good positions for sensor resetting in the well known Monte-Carlo

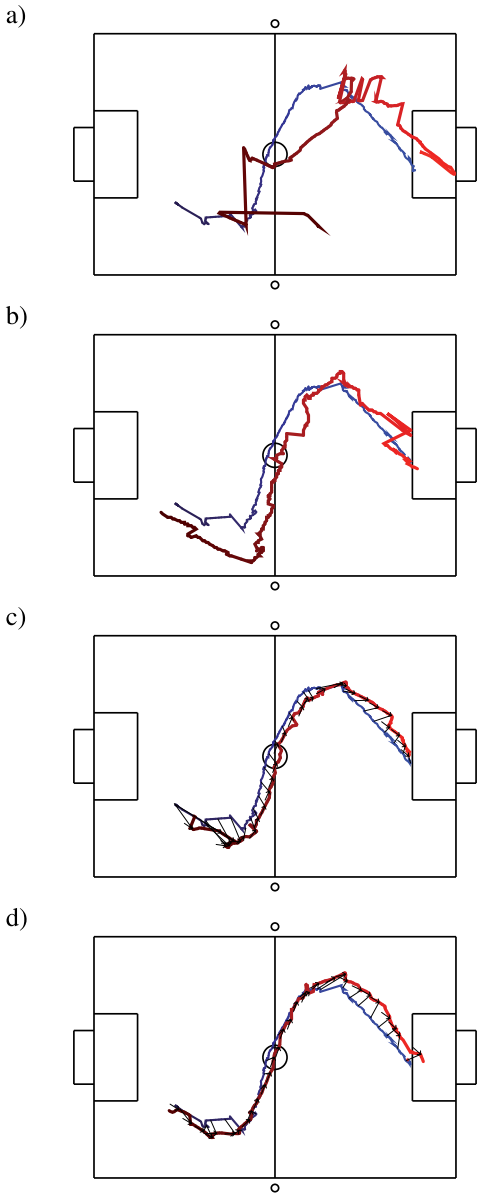

Fig. 8. Comparing ground truth robot position and localization result. Blue line: ground truth robot position. Red line: result of self localization. a) Monte-Carlo localization - no sample templates used. b) Our new approach - maximum of F(x,y). c,d) Monte-Carlo localization using sample templates generated by our approach.

num. of reseeded samples	0	1	2	10	20	plain
deviation in cm	54.8±21.6	21.7±13.1	19.1±13.0	19.5±12.6	22.4±17.0	30,8±20,7
deviation relative to field length	9,13%	3,63%	3,18%	3,25%	3,74%	5,13%

Fig. 9. Results of Localization tests. In our experiment the position obtained by the approach introduced in this paper was compared with the one obtained by the ceiling camera. The table shows the average distance between the two positions for the whole run repeated six times. To show how reseeding influences the localization quality we conducted the experiment with different re-sampling rates (top row). The table shows that even a single reseeded particle in each frame improves self-localization drastically. Adding more samples has almost no effect. The last row gives the results for the plain maximum-method (no MCL).

localization. Tests in simulation and on a real Aibo robot with ground truth by a ceiling camera showed the robustness of our approach. Further experiments have to show whether the localization method can cope with larger errors in odometry caused by strong influence of opponents in RoboCup games.

The method does not need an internal representation of the position estimate which is updated by alternating sensor and motion updates. The observation and motion information out of the robot's sensor history is processed directly. A big advantage is that no wrong model from the past can disturb the current pose estimation. However, false observations lead to a false position. Thus the observations to be used have to be selected very carefully. Especially the perceptions should be designed in a way that false positives are excluded.

REFERENCES

[1] F. Dellaert, W. Burgard, D. Fox, , and S. Thrun. Using the condensation algorithm for robust, vision-based mobile robot localization. In *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 1999.

[2] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte Carlo localization: Efficient position estimation for mobile robots. In *Proc. of the National Conference on Artificial Intelligence*, 1999.

[3] J. Hoffmann, M. Spranger, D. Goehring, and M. Juengel. Making use of what you don't see: Negative information in markov localization. In *Proceedings of the 2005 IEEE/RSJ International Conference of Intelligent Robots and Systems (IROS)*. IEEE, 2006.

[4] M. Juengel. Using layered color precision for a self-calibrating vision system. In D. Nardi, M. Riedmiller, C. Sammut, and J. Santos-Victor, editors, *8th International Workshop on RoboCup 2004 (Robot World Cup Soccer Games and Conferences)*, volume 3276 of *Lecture Notes in Artificial Intelligence*, pages 209–220. Springer, 2005.

[5] S. Lenser and M. M. Veloso. Sensor resetting localization for poorly modelled mobile robots. In *Proceedings of the 2000 IEEE International Conference on Robotics and Automation (ICRA 2000)*, pages 1225–1232. IEEE, 2000.

[6] T. Roefer, R. Brunn, I. Dahm, M. Hebbel, J. Hoffmann, M. Juengel, T. Laue, M. Loetzsch, W. Nistico, and M. Spranger. GermanTeam 2004: The German national RoboCup team. In D. Nardi, M. Riedmiller, C. Sammut, and J. Santos-Victor, editors, *RoboCup 2004: Robot Soccer World Cup VIII*, volume 3276 of *Lecture Notes in Artificial Intelligence*. Springer, 2005. Full team report can be downloaded at http://www.germanteam.org/.

[7] T. Roefer and M. Juengel. Vision-based fast and reactive monte-carlo localization. In D. Polani, A. Bonarini, B. Browning, and K. Yoshida, editors, *Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA)*, pages 856–861. IEEE, 2003.

[8] T. Roefer and M. Juengel. Fast and robust edge-based localization in the sony four-legged robot league. In D. Polani, A. Bonarini, B. Browning, and K. Yoshida, editors, *7th International Workshop on RoboCup 2003 (Robot World Cup Soccer Games and Conferences)*, volume 3020 of *Lecture Notes in Artificial Intelligence*, pages 262–273. Springer, 2004.

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

[10] J. Wolf, W. Burgard, and H. Burkhardt. Robust vision-based localization for mobile robots using an image retrieval system based on invariant features. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2002.