# Self-Localization Using Odometry and Horizontal Bearings to Landmarks

Matthias Jüngel and Max Risler

[1] Humboldt-Universität zu Berlin, Künstliche Intelligenz
Unter den Linden 6, 10099 Berlin, Germany
matthias@matthias-juengel.de
[2] Technische Universität Darmstadt, Simulation and Systems Optimization Group
Hochschulstr. 10, 64289 Darmstadt, Germany
risler@sim.informatik.tu-darmstadt.de

**Abstract.** On the way to the big goal - the game against the human world champion on a real soccer field - the configuration of the soccer fields in RoboCup has changed during the last years. There are two main modification trends: The fields get larger and the number of artificial landmarks around the fields decreases. The result is that a lot of the methods for self-localization developed during the last years do not work in the new scenarios without modifications. This holds especially for robots with a limited range of view as the probability for a robot to detect a landmark inside its viewing angle is significantly lower than on the old fields. On the other hand the robots have more space to play and do not collide as often as on the small fields. Thus the robots have a better idea of the courses they cover (odometry has higher reliability). This paper shows a method for self-localization that is based on bearings to horizontal landmarks and the knowledge about the robots movement between the observation of the features.

**Key words:** Self-Localization, Constraints, Aibo, Bearing-Only

## 1 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 robot 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 [1] and there are improvements that build short-time history of observations to create more accurate position templates [2]. 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.

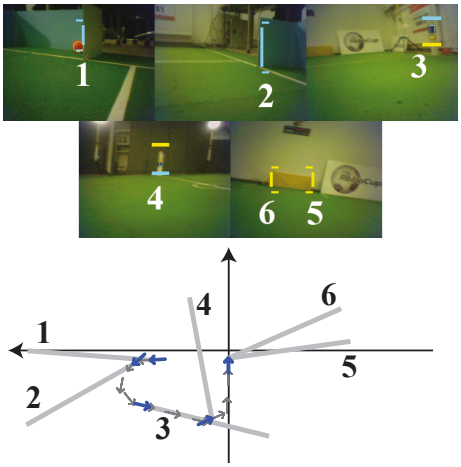


**Fig. 1.** Odometry and horizontal bearings. Top: Five images with six horizontal bearings (1: right goal post, 2: left goal post, 3 and 4: center landmarks, 5 and 6 goal posts) Bottom: Gray arrows show the robots odometry at different times, bold arrows show the odometry associated with the horizontal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

This work was motivated by the experiences we collected with the localization method we use in RoboCup for our Aibo robots. We use a standard Monte-Carlo localization as described in [4–6]. As with the latest rule changes in the RoboCup Sony Four-Legged League besides the goals there are only two artificial landmarks on the field, the distance-based sensor resetting method does not give the desired results any longer. Size-based distance measurements have a to large error when the objects are too far away.

In this paper we provide a bearing-only method for localization that incorporates odometry and can be used as a template generator for MonteCarlo-Localization. This paper shows an approach to bearing-only self-localization that incorporates odometry in a new way. Section 2 describes the method in detail. Section 3 describes the experiments we performed with our Aibo robots.

## 2 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 $\boldsymbol{\alpha} = (\alpha_{l_1}, \alpha_{l_2}, ..., \alpha_{l_n})$ contains the measured bearings to the landmarks $l_1, l_2, ..., l_n$. These angles were measures at different times $t_1, t_2, ..., t_n$. The second input is the knowledge about the motion of the robot. The vector $\boldsymbol{u} = (u_1, u_2, ..., u_n)$ contains the robot's odometry at times $t_1, t_2, ..., t_n$.

A robot can obtain these vectors $\boldsymbol{\alpha}$ and $\boldsymbol{u}$ 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 $F(x, y, \boldsymbol{\alpha}, \boldsymbol{u})$ 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.

### 2.1 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.

**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.

**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 $\vartheta_l$ 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 $\alpha_l$ is the relative angle to the landmark. When two bearings to two landmarks are given, the function

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

describes the likelihood for being at position $(x, y)$. 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 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 $\vartheta_{l_1}$ and $\vartheta_{l_2}$ 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

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

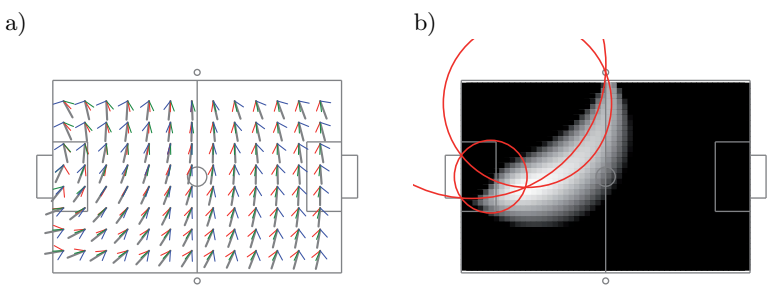


**Fig. 2.** Similarity of angles. a) The thin lines show the angles $\vartheta_l(x, y)$ for three different observations. The bold line shows the average angle. The robot's position is constrained to the positions where the angles are similar. b) Function $G(x, y)$ displayed as height map. White: small difference between the angles, black: large difference between the angles. The red circles are obtained from the method using simple geometry described above.

Figure 2 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

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

which describes how similar the angles $\vartheta_l$ 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 2 b) shows this function for three observations.

### 2.2 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 $\alpha_l$ and the robot moved $\Delta_{odometry}(\Delta_x, \Delta_y, \Delta_\phi)$ since the observation. Figure 3a) illustrates these parameters and the resulting angle $\upsilon_l$. To determine $\upsilon_l$ we define a triangle with its corners at the position $(x_l, y_l)$ of the landmark $l$ (angle $\beta$), at the position $(x, y)$ (angle $\gamma$) and at the position $(x_0, y_0)$ where the observation was taken (angle $\delta$). Figure 3b) 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 $\omega$ from $(x, y)$ to $(x_l, y_l)$ and the distance $\Delta_d$ the robot walked:

$$x_0 = x + \cos(\omega + \gamma) \cdot \Delta_d; y_0 = y + \sin(\omega + \gamma) \cdot \Delta_d$$

where $\gamma$ follows using sine rule:

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

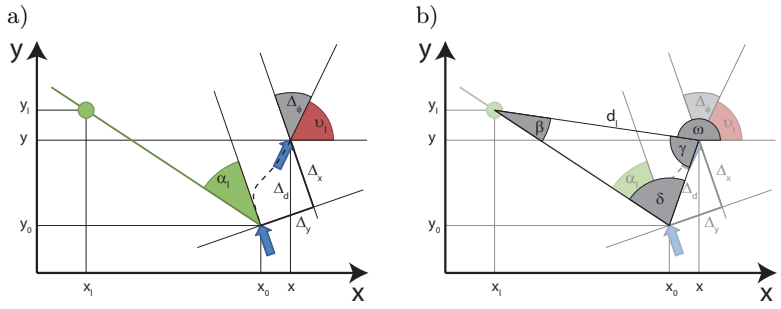


**Fig. 3.** Bearing + odometry define the robots angle for a given position $(x_0, y_0)$

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 $\alpha_l$ some time ago, and has moved by $\Delta_{odmetry_l}$ since that observation, the function $\upsilon_l$ gives the angle the robot must have. Similar to the function $G$ from section 2.1 we define a function

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

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 $\boldsymbol{\alpha}$ and the according motion information $\boldsymbol{u}$ are processed at once.

### 2.3 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 $\upsilon_{l_0}$ 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.

### 2.4 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

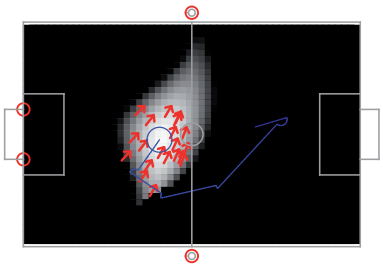

**Fig. 4.** The Function $F(x, y)$: white - high likelihood, black - low likelihood, Arrows: position templates that can be used for sensor resetting in Monte-Carlo localization - note that usually only a small number of these templates will be used. Small circles: the landmarks that were used for position calculation. Large circle: the robot pose (known from the simulation). Path: the way the robot walked.

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 2.2 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 using function $F' := 1/(1 + F^2)$ 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 4 shows templates obtained from function $F$.

## 3 Experimental Results

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 [7–9]. The old method was not usable any longer as with the 2007 rule change in the 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 2.2 as a sample template generator in a way described in section 2.4. 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 logfiles can be played back off-line to feed our algorithms with data. The angles to the landmarks needed for our location approach were extracted from images and

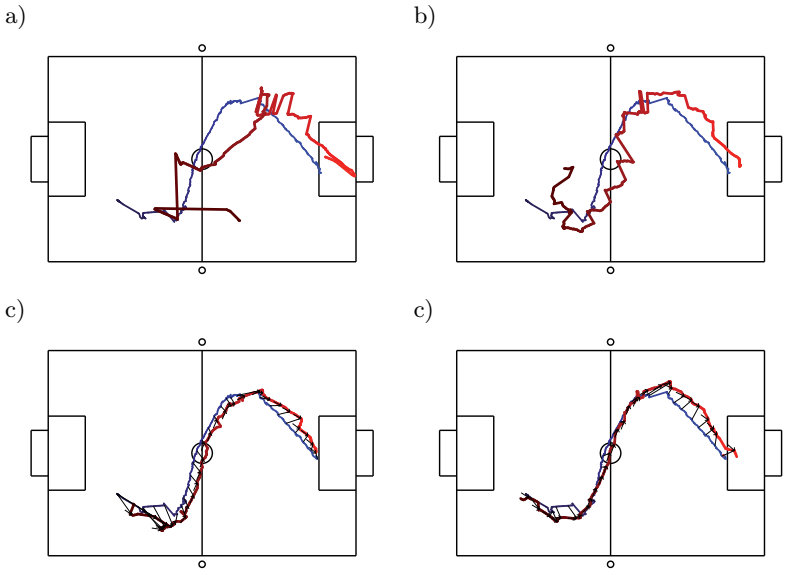


**Fig. 5.** blue line: ground truth robot position, red line: result of self localization a,b) no sample templates used. c,d) sample templates used.

joint sensor data. We used the recorded log data to compare different parameterizations of the approach.

Figure 5 shows a visualization of the path the robot walked and the pathes obtained by our method. We also tested the influence the number of samples used for reseeding has. Table 1 gives the results.

The 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.

| **num. of reseeded samples** | **0** | **1** | **2** | **5** | **10** |
|---|---|---|---|---|---|
| position error in cm | 54.8±21.6 | 21.7±13.1 | 19.1±13.0 | 19.5±12.6 | 22.4±17.0 |
| position error percentage | 9,13% | 3,63% | 3,18% | 3,25% | 3,74% |

**Table 1.** 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.

## 4 Conclusion

In this paper we presented an approach for bearing-only self-localization incorporating odometry. The method does not need an internal representation of the position estimate which is updated by alternating sensor and motion updates. The history of observation and motion information (stored in a small buffer) is processed directly. A big advantage is that no wrong model from the past can disturb the current pose estimation.

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

## References

1. Lenser, S., Veloso, M.M.: Sensor resetting localization for poorly modelled mobile robots. In: Proceedings of the 2000 IEEE International Conference on Robotics and Automation (ICRA 2000), IEEE (2000) 1225–1232
2. Sridharan, M., Kuhlmann, G., Stone, P.: Practical vision-based monte carlo localization on a legged robot. In: IEEE International Conference on Robotics and Automation. (April 2005)
3. Hoffmann, J., Spranger, M., Goehring, D., Juengel, M.: 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. Fox, D., Burgard, W., Dellaert, F., Thrun, S.: Monte Carlo localization: Efficient position estimation for mobile robots. In: Proc. of the National Conference on Artificial Intelligence. (1999)
5. Dellaert, F., Burgard, W., Fox, D., Thrun, S.: 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)
6. Wolf, J., Burgard, W., Burkhardt, H.: 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)
7. Roefer, T., Brunn, R., Dahm, I., Hebbel, M., Hoffmann, J., Juengel, M., Laue, T., Loetzsch, M., Nistico, W., Spranger, M.: GermanTeam 2004: The German national RoboCup team. In: RoboCup 2004: Robot Soccer World Cup VIII. (2005)
8. Roefer, T., Juengel, M.: Fast and robust edge-based localization in the sony four-legged robot league. In Polani, D., Bonarini, A., Browning, B., Yoshida, K., eds.: 7th International Workshop on RoboCup 2003 (Robot World Cup Soccer Games and Conferences). Volume 3020 of Lecture Notes in Artificial Intelligence., Springer (2004) 262–273
9. Roefer, T., Juengel, M.: Vision-based fast and reactive monte-carlo localization. In Polani, D., Bonarini, A., Browning, B., Yoshida, K., eds.: Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA), IEEE (2003) 856–861