Robot Recognition and Modeling in the RoboCup Standard Platform League

Alexander Fabisch [#1], Tim Laue [*2], Thomas Röfer [*3]

[#] *Department for Mathematics and Informatics, Universität Bremen*
Postfach 330440, 28334 Bremen, Germany
[1] afabisch@informatik.uni-bremen.de

[*] *Safe and Secure Cognitive Systems, German Research Center for Artificial Intelligence (DFKI)*
Enrique-Schmidt-Str. 5, 28359 Bremen, Germany
[2] Tim.Laue@dfki.de [3] Thomas.Roefer@dfki.de

***Abstract*— To achieve a reasonable level of play in the RoboCup Standard Platform League, a number of basic abilities such as obstacle avoidance and passing are necessary. Most of these abilities have one thing in common: they rely on information about other robots. In this paper, we present a vision-based approach for robot recognition in the RoboCup Standard Platform League as well as an algorithm to track the recognized robots. Both approaches were developed considering the limited computing resources of the Nao to allow an application in actual games.**

I. Introduction

Over the years, the level of play in the RoboCup Standard Platform League (SPL) has increased significantly. Teams are able to perform robust obstacle avoidance and to target shots at uncovered parts of the goals. Higher level skills such as passing can be expected to become common within the next few years. All these abilities have to take the positions of teammates and opponents into account. Given the low computational resources of the Nao robot, their perception is a rather difficult task, making the application of state-of-the-art computer vision approaches such as the *Histogram of Oriented Gradients (HOG)* [1] impossible. In addition, the robot's sensorial limitations – the robot has only one active camera with a quite limited field of view – require an additional sophisticated modeling of the other robots.

A workaround is to use the robot's built-in ultrasonic sensors instead. This has been described by the B-Human team [2] who use a simple occupancy grid [3] for increasing robustness. However, these sensors are quite imprecise due to their very low resolution and, among other things, are not able to detect robots that lie on the ground, making the avoidance of such robots impossible.

Vision-based solutions that do not require the explicit detection of robots were presented by [4] and [5]. Their approaches perceive the floor in the robot's close environment and consider significant gaps as obstacles.

For detecting robots in the RoboCup Four-legged League, a solution based on decision tree learning has been presented [6]. For the same domain, a tracking approach that requires only basic and unprecise perceptions of robots has been shown by [7]. A quite different approach that is based on neural networks has been presented by [8] for detecting solid black robots in the RoboCup Middle Size League.

This paper presents solutions to reliably detect Nao robots via vision and to track these detected robots. The tracking is realized by a Kalman filter-based approach. A preliminary version has already been presented in [2]. This paper describes a revised robot detection that is more independent of proper color segmentation by means of growing the regions provided by the existing vision system. In addition, more constraints on the team markers are verified to eliminate false robots early. The robot tracking algorithm is described more detailed and has been evaluated in different experiments. This work has been realized in the context of the B-Human SPL team and thus uses its software framework and infrastructure as described in [2].

This paper is organized as follows: in Sect. II, we present our approach for the visual robot recognition. In the following Sect. III, we describe the algorithm that is used to track the detected robots. Finally, we present the results of experiments to evaluate the quality of both algorithms in Sect. IV.

II. Detecting Other Robots

Since 2010, the robots of the RoboCup Standard Platform League are mostly white with a few gray spots and wear

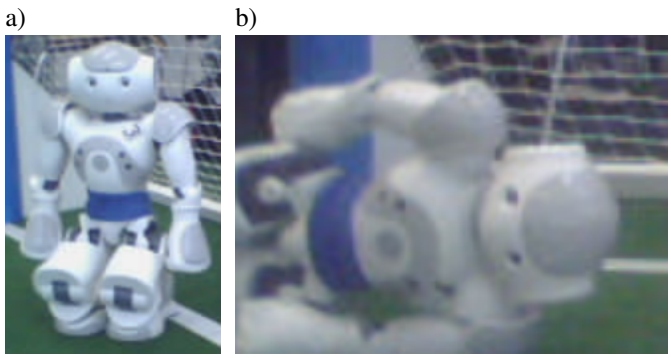

Fig. 1. a) A Nao robot playing in the RoboCup Standard Platform League. b) The color of the blue team marker and the blue goal differ hardly on the images taken by the Nao's camera.

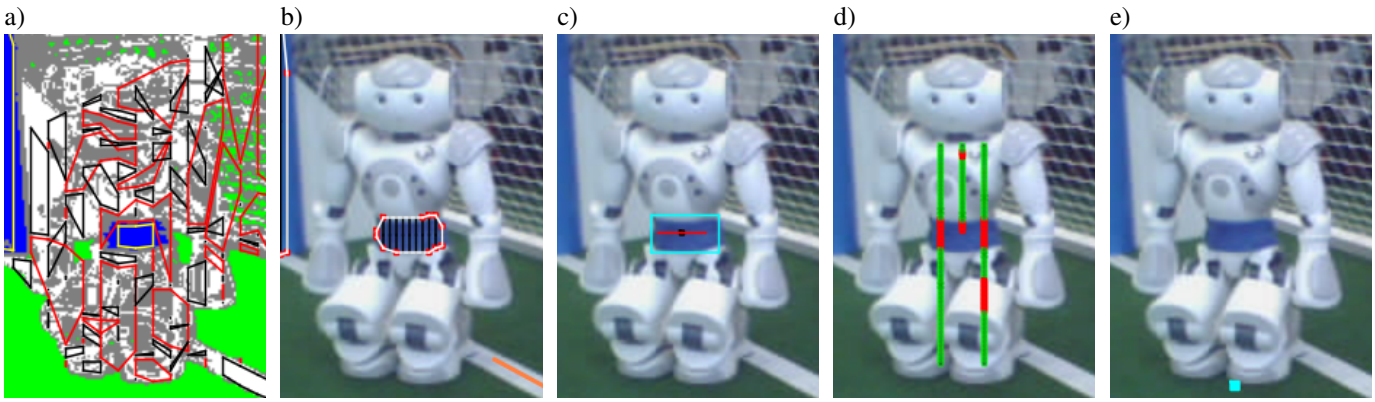

Fig. 2. a) The classified image of the robot. The team marker region – marked by a yellow border – is not completely classified as blue. b) More vertical scanlines (depicted in black) are added to determine the real border of the team marker. Afterwards, the team marker is described as a polygon. The red dots mark vertices of the polygon and the white lines display the edges. Collinear vertices have already been deleted. c) The blue box denotes the bounding box of the team marker, the black cross is the center of mass of the team marker's region and the red line describes the orientation of the team marker as well as the potential minimal width of the robot. d) Five scanlines verify the robot above and under the marker. The green dots show that the difference of the reference pixel and the marked pixel is within the boundaries and the red dots show that the difference of the colors is too high. e) The position of the recognized robot is determined by transforming the pixel marked by the blue dot into the soccer field's coordinate system.

blue or pink team markers around their waists to indicate their team (cf. Fig. 1a). The current vision system of the team B-Human is based on color segmentation and region building as described in detail in [2]. Hence, it is necessary to extract information about possible robot perceptions from the previously generated regions. A particular challenge is the distinction of blue team markers and blue goal elements (cf. Fig. 1b).

A. Detecting Team Markers

The current vision of the RoboCup team B-Human is based on color segmentation [2]. Thus, we already start with blue and pink regions representing possible team markers. These are usually not sufficient to reliably estimate the dimensions of the robot (cf. Fig. 2a). Therefore, we start at the center of mass of the team marker region and scan horizontally to determine a more precise width of the team marker. Based on this scanline, we add vertical scanlines to determine a polygon that describes the team marker region (cf. Fig. 2b). In contrast to the previous region building, we use a higher resolution and another method to determine the end of regions. The scanlines will end, if

$$|cb - cb'| \geq t_1 \text{ or } |cr - cr'| \geq t_2,$$

where the color of the current pixel is given by (y, cb, cr) in the YCbCr color space and is compared to the reference pixel's color (y', cb', cr'). The reference pixel is the center of mass of the initial region. t_1 and t_2 are thresholds that depend on the individual camera settings and are, as well as all the other parameters mentioned in this paper, only manually tuned. We do not calculate the combined error of both components because a slight change of one color component can result in a completely different color. If we would calculate the combined distance of the colors, we had to choose a higher threshold to not discard real team markers but this would allow higher deviations in one component if the other component did not change. The y-component is completely omitted because it is usually prone to variations of the illumination.

The result of these scanlines is a polygon around the team marker. To reduce the complexity of the description, we remove collinear vertices.

B. Evaluating Team Markers

Subsequent to their detection, the team marker regions have to be checked for some constraints.

First of all, the following numerical qualitative features of the polygon will be calculated and checked: curvature, extent and extremum. They allow us to verify the team marker's shape efficiently and are invariant against scale, translation and rotation. See [9] for a detailed description and review. All the features are normalized for polygons, i. e. they are within the range of $[0, 1]$. To evaluate the shape, we simply check whether the features are within a certain interval.

We also check for a minimal and a maximal size of the team marker's area. Our approach is able to detect horizontal robots, i. e. those that are lying on the field, almost as well as standing robots or robots that are standing up. To determine whether a robot is standing or not, we compare the length along the x-axis to the maximal length along the y-axis. Usually, the width of a team marker exceeds its height. To distinguish team markers from nearly quadratic blue regions or longish blue regions such as goal posts, we check if the ratio of the width and height is within a certain interval. This enables us to eliminate false robots early and thus to be more efficient.

For the remaining potential robots' team markers we compute a line that describes the team marker's orientation as well as the robots assumed width (cf. Fig. 2c). The orientation is calculated as the average of the orientations of the vectors from the first vertex of the polygon's upper half to the other vertices of the upper half and the orientation of the first vertex of the lower half of the polygon and the other vertices of the lower half.

C. Detecting Robots

After detecting regions that are potential team markers, we can now scan their environment to determine whether they belong to a robot or not. The region above and below the team marker should be white and almost homogenous. Thus, we can calculate scanlines where the robot's body should be (cf. Fig. 2d) based on the previously calculated line of the team marker by rotating and stretching it.

To check the "whiteness" of the team marker's environment, we have to define a reference pixel. We use the first pixel that is classified as white of every scanline as the reference pixel for that scanline. The following pixels on the scanline belong to the robot, if

$$|cb - cb'| + |cr - cr'| \leq t_3,$$

where t_3 is another threshold. Since we calculate the combined error of both color components here, its value has to be chosen higher than for t_1 and t_2. After each scanline, we check if the ratio of white pixels and scanned pixels is tolerable. If this is not the case, we discard the potential robot.

If a potential robot passed all these tests, we have to find a point that allows the localization of the robot on the field. Since it is inaccurate to determine the robot's position if we just know the position of the team marker, we search for the first green area below the robot, so that we can project a pixel on the field (cf. Fig. 2e).

III. Robot Tracking

To track robots that have been recognized by the vision system, we use a Kalman filter for each detected robot. The pseudo code of our robot tracking algorithm is listed as algorithm 1. Based on the notation of [10], the mean and covariance after the motion update of the tth iteration are denoted as $\overline{\mu}_t$ and $\overline{\Sigma}_t$, respectively the mean and covariance after the sensor update are denoted as μ_t and Σ_t. The notation $tr.\Sigma_t$ refers to the covariance matrix of the Kalman filter for the robot tr. We use several noise parameters σ_{a_b} with $a \in \{m, s, i, n\}$ and $b \in \{x, y\}$, where m stands for motion update, s for sensor update, i for initial and n for no sensor update, so that σ_{s_x} denotes the standard deviation of the distance measurement if the recognized robot is ahead. These can be constants or can be estimated by considering the head motion, distance of the robot or other elements.

Actually this is not a real Kalman filter, since we did not implement a realistic motion model yet. We are currently assuming that all recognized robots do not move. Instead, we add a significant noise in each cycle so that we can react faster to the robot's movements. Thus, motion is considered as noise, which is not very inaccurate since most of the robots of the Standard Platform League usually do not move very fast yet. The main purpose of the motion update is to apply the motion of the observing robot by subtracting the odometry offset.

The robots' environment is partially observable. The challenging part of the tracking module is to match recognized robots with tracked robots. The basic idea is to calculate the Mahalanobis distance of the detected robots and the tracked robots and always match those robots with the lowest distance. This approach would work perfectly for fully observable environments. The Mahalanobis distance, as developed in [11], can be calculated as follows:

$$d(x, y) = \sqrt{(x - y)(\Sigma^{-1}(x - y))}.$$

This requires a good estimation of the measurement noise. We are usually quite sure about the angle of the recognized robot and unsure about the distance. So if the perceived robot is in front of the observing robot, i. e. the bearing is 0, the deviation along the x-axis is high and the deviation along the y-axis is low. If the perceived robot is not in front of the observer, we have to rotate the covariance of the measurement.

Algorithm 1 Robot Tracking Algorithm

Initialization:
$T \leftarrow$ list of previously tracked robots (initially empty)
$P \leftarrow$ list of currently perceived robots
$u_t \leftarrow$ translational odometry offset
$\theta \leftarrow$ rotational odometry offset
Motion Update:
for all $tr \in T$ **do**
  $A_t \leftarrow \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$
  $R_t \leftarrow \begin{pmatrix} \sigma^2_{m_x} & 0 \\ 0 & \sigma^2_{m_y} \end{pmatrix}$
  $tr.\overline{\mu}_t \leftarrow A_t \cdot \mu_{t-1} + u_t$
  $tr.\overline{\Sigma}_t \leftarrow A_t \cdot \Sigma_{t-1} \cdot A_t^T + R_t$
end for
Sensor Update:
for all $pr \in P$ **do**
  $z_t \leftarrow$ relative position of pr on the field
  $\alpha \leftarrow$ angle to the measured robot
  $tr \leftarrow \text{argmin}_{tr \in T}(z_t - tr.\overline{\mu}_t)(tr.\overline{\Sigma}_t^{-1}(z_t - tr.\overline{\mu}_t))$
  if $euclideanDist(tr.\overline{\mu}_t, z_t) \leq$ maxAllowedDist **then**
    if tr is not updated yet **then**
      $Q_t \leftarrow \Sigma(\alpha, \sigma_{s_x}, \sigma_{s_y})$
      $K_t \leftarrow \overline{\Sigma}_t \cdot (\overline{\Sigma}_t + Q_t)^{-1}$
      $tr.\mu_t \leftarrow \overline{\mu}_t + K_t \cdot (z_t - \overline{\mu}_t)$
      $tr.\Sigma_t \leftarrow (I - K_t) \cdot K_t$
    end if
  else
    $pr.\Sigma_t \leftarrow \Sigma(\alpha, \sigma_{i_x}, \sigma_{i_y})$
    $pr.\mu_t \leftarrow z_t$
    Append pr to T
  end if
end for
for all $tr \in T$ **do**
  if $2\pi\sqrt{det(tr.\Sigma_t)} <$ minProbabilityAtMean **then**
    Remove tr from T
  else if tr has not been updated but should be visible **then**
    $\alpha \leftarrow$ angle to the tracked robot
    $tr.\Sigma_t \leftarrow tr.\Sigma_t + \Sigma(\alpha, \sigma_{n_x}, \sigma_{n_y})$
  end if
end for

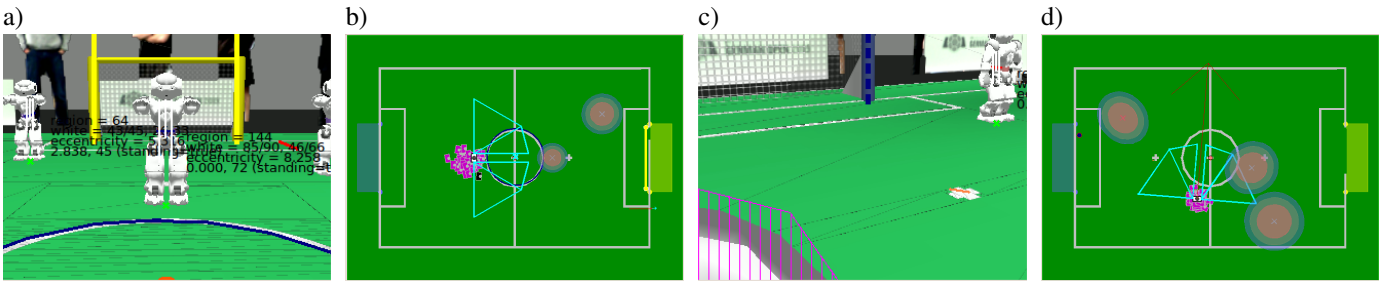

Fig. 3. a) Simulated camera image before kickoff. The white scan lines mark recognized robots. b) The corresponding debug drawings on the field. The crosses mark the assumed position of the robots and the slight transparent circles represent the error ellipse calculated from the covariance matrix. c) A recognized robot on the left side. d) The corresponding error ellipse shows that the uncertainty of the distance is greater than the uncertainty of the angle.

We do this by multiplying the covariance with a rotation matrix from both sides:

$$\Sigma(\alpha, \sigma_x, \sigma_y) = Rot(\alpha) \cdot \begin{pmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{pmatrix} \cdot Rot(\alpha),$$

where α is the angle of the measurement vector in the robot coordinate system and

$$Rot(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}.$$

σ_x and σ_y are parameters that have to be defined. The result of the rotation is displayed in Figure 3d. See [12] for a detailed proof.

In fact, it is unlikely that all robots are tracked at the same time. So it is difficult to find out whether a detected robot correlates with a robot in the model or is a new robot. To solve this problem, we introduced the Euclidean distance as a criterion to determine whether we update an existing robot or add a new robot to our model. If the Euclidean distance of the recognized robot exceeds a defined threshold, it will not update the previously tracked robot.

There are several reasons why a tracked robot will not be recognized when it should be visible:

- A robot could be hidden by other robots or a referee.
- It could have moved out of the visible area.
- The robot is in the image but it was not recognized.
- The tracked robot was a false positive. This would be a problem that significantly affects the usability of the model.

As we lose track of robots if we did not get any sensor updates for a long time, robots will not be tracked any longer if the value of the probability density function at the mean falls below a threshold. So we can solve this case by increasing the covariance of tracked robots that should be visible but have not been detected on the camera image, i. e. we do a special sensor update for these robots. Thus, false positives will be removed from the model very fast.

IV. Evaluation

To evaluate the robot detection and modeling, we conducted four experiments, including a standing as well as a moving observer. During each experiment, there were three sitting robots with a pink team marker on the positions (180 cm, -137 cm), (265 cm, 0 cm) and (120 cm, 70 cm) in the global coordinate system as defined in [2]. The robot always starts at the center of the field with the coordinates (0 cm, 0 cm). The robot cannot see all of the other robots at once, hence it has to look around and the image will be distorted and slightly blurred. We used the SSL-Vision application [13] to track the robot's pose to eliminate any self localization errors during the experiment in which the robot was walking. To reduce rotational errors of the localization due to asynchronism and delays, we used a special construct to attach the SSL patterns to the robot that is independent of the head motions (cf. Fig. 4). The results of the experiments are shown in Fig. 6.

In the first experiment we just let the robot stand on the center and look around. The robot's model of the world is pictured in Fig. 5. As you can see in Fig. 6a, the average localization error of all three robots is around 20 cm and about half of the time we have one false positive. The false positive is caused by cardboard boxes (cf. Fig. 4) that are partially classified as pink and partially classified as white. As the position is correctly estimated outside the field, such false positives would not affect any of the robot's decisions. During the whole experiment, all three real robots have been recognized. In the following experiments, almost no false positives occured.

For the second experiment, the robot moves its head left

Fig. 4. The general setup of the experiments. Three robots are on fixed positions on the field. The observing robot is provided with ground truth pose information from the SSL-Vision system. For this purpose, it has a pattern attached to its head that is tracked by a camera.

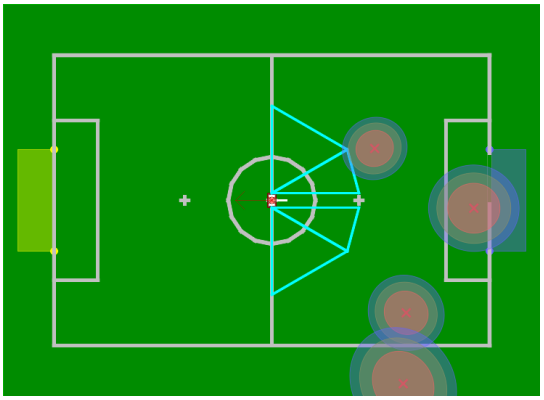

Fig. 5. The robot's model of the world during the first experiment.

and right and walks on the center of the field without any translation or rotation. The result is shown in figure 6b. In this experiment, the goalie has not been perceived continuously. The localization error for the goalie is the largest, since the distance to the goalie is the greatest and there is a white line in front of the robot, hence sometimes the first green area below the team marker could be in front of this white line. This results in an estimated position that is displaced by 30 cm. Thus, the average localization error decreases when the goalie is not recognized.

During the last experiments, we controlled the robot via joystick. The robot's paths are displayed in Fig. 6e and f. Especially in the third experiment, the localization error oscillated significantly. The path of the robot is very curvaceous. So imprecise rotational odometry data have a greater impact on the localization error. Since we measured the localization error relative to the distance between the observing and the observed robot, the percentual localization error is huge, when the observed robot is near the observed robots. This the reason for the peak at the end of the experiment. For comparison, we conducted an experiment with a walking robot, that did almost not rotate. The result is displayed in Fig. 6d. The error is smaller and less oscillating.

V. Conclusions

The experiments show that robustly recognizing robots that are more than 2.6 m away is hardly possible, but robots that are 2.2 m away can be reliably detected. The tracking algorithm depends on accurate information about the observing robot's rotation and needs frequent updates about the observed robot's position. So we have to optimize the head motions to follow the ball, lines and goals and, additionally, other robots. The average localization error was always less than 25 cm, if the observing robot did not walk. Further tests have to be done with moving opponents and other more realistic situations such as real game situations.

The future applications for a reliable model of the other robots will be numerous. Passing and aiming at the largest free part of the opponent goal are only the simplest improvements. But those are the situations we consider as good test cases to improve our approach. Our goal is to implement both reliably until RoboCup 2011.

Acknowledgments

The authors would like to thank all B-Human team members for providing the software base for this work and in particular Katharina Gillmann for her assistance in conducting the experiments and Alexander Härtl for constructing the frame for the SSL pattern and his assistance in analyzing the evaluation data.

References

[1] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *International Conference on Computer Vision & Pattern Recognition*, C. Schmid, S. Soatto, and C. Tomasi, Eds., vol. 2, June 2005, pp. 886–893.

[2] T. Röfer, T. Laue, J. Müller, A. Burchardt, E. Damrose, A. Fabisch, F. Feldpausch, K. Gillmann, C. Graf, T. J. de Haas, A. Härtl, D. Honsel, P. Kastner, T. Kastner, B. Markowsky, M. Mester, J. Peter, O. J. L. Riemann, M. Ring, W. Sauerland, A. Schreck, I. Sieverdingbeck, F. Wenk, and J.-H. Worch, "B-Human Team Report and Code Release 2010," 2010, only available online: http://www.b-human.de/file_download/33/bhuman10_coderelease.pdf.

[3] H. P. Moravec and A. Elfes, "High resolution from wide angle sonar," in *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, vol. 2, St. Louis, MO, USA, 1985, pp. 116–121.

[4] S. Lenser and M. Veloso, "Visual sonar: Fast obstacle avoidance using monocular vision," in *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, Las Vegas, NV, USA, 2003.

[5] J. Hoffmann, M. Jüngel, and M. Lötzsch, "A vision based system for goal-directed obstacle avoidance," in *RoboCup 2004: Robot Soccer World Cup VIII*, ser. Lecture Notes in Artificial Intelligence, D. Nardi, M. Riedmiller, C. Sammut, and J. Santos-Victor, Eds., vol. 3276. Springer, 2005, pp. 418–425.

[6] D. Wilking and T. Röfer, "Real-time object recognition using decision tree learning," in *RoboCup 2004: Robot Soccer World Cup VIII*, ser. Lecture Notes in Artificial Intelligence, D. Nardi, M. Riedmiller, C. Sammut, and J. Santos-Victor, Eds., vol. 3276. Springer, 2005, pp. 556–563.

[7] T. Laue and T. Röfer, "Integrating simple unreliable perceptions for accurate robot modeling in the four-legged league," in *RoboCup 2006: Robot Soccer World Cup X*, ser. Lecture Notes in Artificial Intelligence, G. Lakemeyer, E. Sklar, D. G. Sorrenti, and T. Takahashi, Eds., vol. 4434. Springer, 2007, pp. 474–482.

[8] U. Kaufmann, G. Mayer, G. Kraetzschmar, and G. Palm, "Visual robot detection in robocup using neural networks," in *RoboCup 2004: Robot Soccer World Cup VIII*, ser. Lecture Notes in Artificial Intelligence, D. Nardi, M. Riedmiller, C. Sammut, and J. Santos-Victor, Eds., vol. 3276. Springer, 2005, pp. 262–273.

[9] B. Gottfried, A. Schuldt, and O. Herzog, "Extent, extremum, and curvature: Qualitative numeric features for efficient shape retrieval," in *KI '07: Proceedings of the 30th annual German conference on Advances in Artificial Intelligence*. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 308–322.

[10] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. The MIT Press, 2005.

[11] P. C. Mahalanobis, "On the Generalised Distance in Statistics," in *Proceedings of the National Institute of Sciences of India*, vol. 2, no. 1, 1936, pp. 49–55.

[12] T. Soler and M. Chin, "On transformation of covariance matrices between local cartesian coordinate systems and commutative diagrams," in *Technical Papers. 45th Annual Meeting ACSM*, Washington, D.C.,USA, March 1985, pp. 393–406.

[13] S. Zickler, T. Laue, O. Birbach, M. Wongphati, and M. Veloso, "SSL-Vision: The Shared Vision System for the RoboCup Small Size League," in *RoboCup 2009: Robot Soccer World Cup XIII*, ser. Lecture Notes in Artificial Intelligence, J. Baltes, M. G. Lagoudakis, T. Naruse, and S. Shiry, Eds., vol. 5949. Springer, 2010, pp. 425–436.

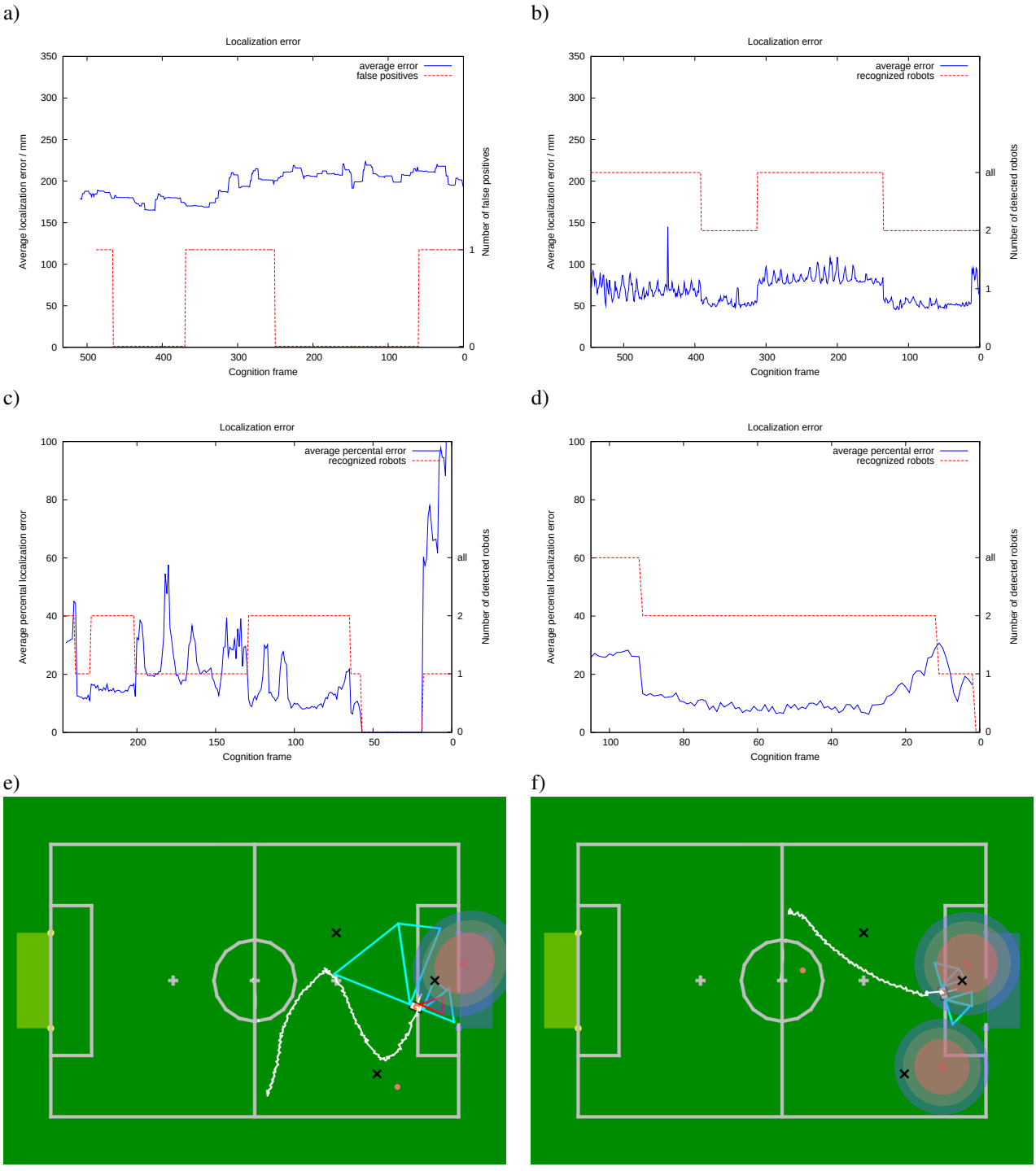

Fig. 6. a) The result of the first experiment. The x-axis displays how many cognition frames have passed since the last measurement, hence the values are decreasing from left to right. The localization error displayed at the y-axis is the average distance of every perceived robot to the real position of that robot. b) The same diagram for the second experiment. Here the error seems to be less. This is the case because the placement of the robots was not exactly the same. The average displacement is about 10 cm. The localization error is decreased by this value because we placed the robots with their tiptoes on the positions instead of with their centers. Another reason for this offset is, that the robot in front of the goal which has the greatest localization error, is not recognized all the time. c) The result of the third experiment. In this case, we displayed the average percentual localization error, which is the ratio of the absolute localization error and the distance between the observing and the observed robot. d) The result of the fourth experiment. e) The robot's path during the third experiment is marked by the white line. The black crosses show the position of the other robots. f) The robot's path during the fourth experiment.