IEEE Transactions on Robotics and Automation,
18(5):670-684, October 2002

Cooperative Probabilistic State Estimation for Vision-based Autonomous Mobile Robots

Thorsten Schmitt, Robert Hanek, Michael Beetz, Sebastian Buck, and Bernd Radig

***Abstract*— With the services that autonomous robots are to provide becoming more demanding, the states that the robots have to estimate become more complex. In this article, we develop and analyze a probabilistic, vision-based state estimation method for individual, autonomous robots. This method enables a team of mobile robots to estimate their joint positions in a known environment and track the positions of autonomously moving objects. The state estimators of different robots cooperate to increase the accuracy and reliability of the estimation process. This cooperation between the robots enables them to track temporarily occluded objects and to faster recover their position after they have lost track of it. The method is empirically validated based on experiments with a team of physical robots.**

***Index Terms*— vision-based localization, multi-robot systems, sensor fusion, cooperative state estimation, multiple hypothesis tracking, robot soccer**

I. Introduction

Autonomous robots must have information about themselves and their environments that is sufficient and accurate enough for the robots to complete their tasks competently. Contrary to these needs, the information that robots receive through their sensors is inherently uncertain: typically the robots' sensors can only access parts of their environments and their sensor measurements are inaccurate and noisy. In addition, control over their actuators is also inaccurate and unreliable. Finally, the dynamics of many environments cannot be accurately modeled and sometimes environments change nondeterministically.

Recent longterm experiments with autonomous robots [1], [2], [3] have shown that an impressively high level of reliability and autonomy can be reached by explicitly representing and maintaining the uncertainty inherent in the available information. One particularly promising method for accomplishing this is *probabilistic state estimation* [4]. Probabilistic state estimation modules maintain the probability densities for the states of objects over time. The probability density of an object's state conditioned on the sensor measurements received so far contains all the information which is available about an object that is available to a robot. Based on these densities, robots are not only able to determine the most likely state of the objects, but can also derive even more meaningful statistics such as the variance of the current estimate.

The authors are with the Department of Computer Science (Fakultät für Informatik) of the Munich University of Technology (Technische Universität München), Boltzmannstrasse 3, 85748 Garching bei München, Federal Republic of Germany, {schmittt,hanek,beetzm,buck,radig}@in.tum.de, Further information about their research and their RoboCup team, *The AGILO RoboCuppers*, can be found at http://www9.in.tum.de/agilo/.

Successful state estimation systems have been implemented for a variety of tasks including the estimation of the robot's position in a known environment [5], the automatic learning of environment maps, the state estimation for objects with dynamic states (such as doors), for the tracking of people locations [6], [7], and gesture recognition [1].

With the services that autonomous robots are to provide becoming more demanding, the states that the robots have to estimate become more complex. Robotic soccer provides a good case in point. In robot soccer (middle size league) two teams of four autonomous robots play soccer against each other. A probabilistic state estimator for competent robotic soccer players should provide the action selection routines with estimates of the positions and may be even the dynamic states of each player and the ball.

This estimation problem confronts probabilistic state estimation methods with a unique combination of difficult challenges. The state is to be estimated by multiple mobile sensors with uncertain positions, the soccer field is only partly accessible for each sensor due to occlusion caused by other robots, the robots change their direction and speed abruptly, and the models of the dynamic states of the robots of the other team are very crude and uncertain.

In this paper, we describe a state estimation module for individual, autonomous robots that enables a team of robots to estimate their joint positions in a known environment and track the positions of autonomously moving objects. The state estimation modules of different robots cooperate to increase the accuracy and reliability of the estimation process. In particular, the cooperation between the robots enables them to track temporarily occluded objects and to faster recover their position after they have lost track of it.

The state estimation module of a single robot is decomposed into subcomponents for self-localization and for tracking different kinds of objects. This decomposition reduces the overall complexity of the state estimation process and enables the robots to exploit the structures and assumptions underlying the different subtasks of the complete estimation task. Accuracy and reliability is further increased through the cooperation of these subcomponents. In this cooperation the estimated state of one subcomponent is used as evidence by the other subcomponents.

The main contributions of this paper are the following ones. First, we show that image-based probabilistic estimation of complex environment states is feasible in real time even in complex and fast changing environments. Second, we show that maintaining trees of possible tracks is particularly useful for estimating a global state based on multiple mobile sensors with

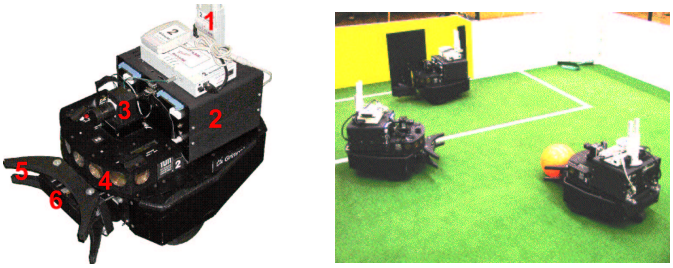
position uncertainty. Third, we show how the state estimation modules of individual robots can cooperate in order to produce more accurate and reliable state estimation.

In the remainder of the paper we proceed as follows. The next section introduces autonomous robot soccer and our research platform. Section III describes the software architecture of the state estimation module and sketches the interactions among its components. Section IV, V and VI provide a detailed description of the individual state estimators. We conclude with our experimental results and a discussion of related work.

II. Autonomous Robot Soccer

Robotic soccer has become a standard "real-world" testbed for autonomous multi robot control [8]. In the middle size league two teams of four autonomous robots — one goal keeper and three field players — play soccer against each other. The soccer field is five times nine meters large and surrounded by walls. The key characteristics of middle size robot soccer is that the robots are completely autonomous. Consequently, all sensing and all action selection is done on board of the individual robots. Skillful play requires our robots to recognize objects, such as other robots, field lines, and goals, and even entire game situations.

The AGILO[1] RoboCup team [9], consists of four Pioneer I robots; one of them is depicted in Figure 1(a). The robot is equipped with a single on board Linux computer (2), a wireless Ethernet (1) for communication, and several sonar sensors (4) for collision avoidance. A color CCD camera with an opening angle of 90^o (3) is mounted fix on the robot. The robot also has a dribbling (5) and a kicking device (6) that enable the robot to dribble and shoot the ball.



(a) (b)

Fig. 1. (a) An AGILO soccer robot and (b) a game situation.

Autonomous robot soccer confronts self-localization and object tracking mechanisms with challenging research problems. The camera system with an opening angle of 90^o and pointed to the front gives an individual robot only a very restricted view of the game situation. Therefore, the robot needs to cooperate to get a more complete picture of the game situation. Vibrations of the camera, spot light effects, and poor lighting conditions cause substantial inaccuracies. Even small vibrations that cause jumps of only two or three pixel lines cause deviations of more than half a meter in the depth estimation, if the objects are several meters away. The opponent robots change their speed and moving directions very quickly and therefore an iteration of the tracking algorithm has to be very fast such that the inaccuracies of the motion model does not have such a huge effect.

[1] The name is an homage to the Agilolfinger, the earliest dynasty ruling Bavaria in the 6th century. The dynasties most famous representatives are Grimoald, Hugibert, Odilo and Tassilo.

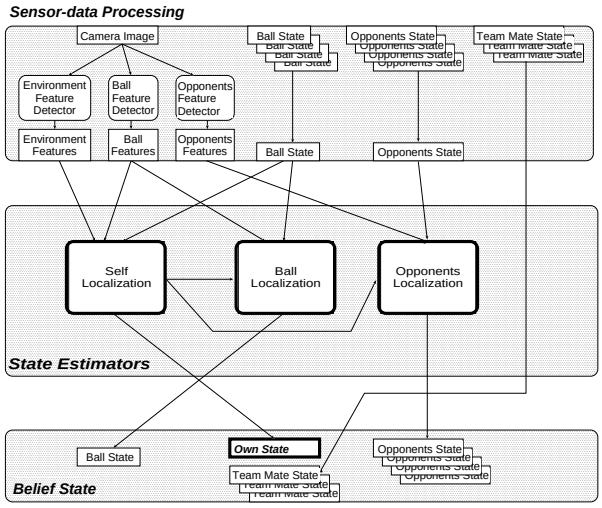


Fig. 2. Architecture of the perception module of the soccer robots.

III. Overview of Sensor Interpretation and State Estimation

The perception module of each soccer robot receives an asynchronous stream of sensor data and maintains a belief state with respect to its own position on the field, the positions of its team mates, the ball, and the opponent players. Figure 2 shows the components of the state estimator and its embedding into the perception module.

This system consists of the sensor-data processing subsystem, the state estimator, and the belief state. The sensor-data processing subsystem itself consists of a camera system with several feature detectors and a communication link that enables the robot to receive information from other robots. The belief state contains a position estimate for each dynamic task-relevant object. In this paper the notion of position refers to the x- and y-coordinates of the objects and includes for the robots of the own team the robot's orientation. The estimated positions are also associated with a measure of accuracy, a covariance matrix.

The sensor-data processing subsystem provides the following kinds of information: (1) partial state estimates broadcasted by other robots, (2) feature maps extracted from captured images, and (3) odometric information. The estimates broadcasted by the robots of the own team comprise the estimate of the ball's location. In addition, each robot of the own team provides an estimate of its own position. Finally, each robot provides an estimate for the position of every visible opponent. From the captured camera images the feature detectors extract problem-specific feature maps that correspond to (1) static objects in the environment including the goal, the borders of the field, and the lines on the field, (2) a color blob corresponding to the ball, and (3) the visual features of the opponents.

The working horse of the sensor-data processing subsystem are a color classification (see section IV-C) and segmentation

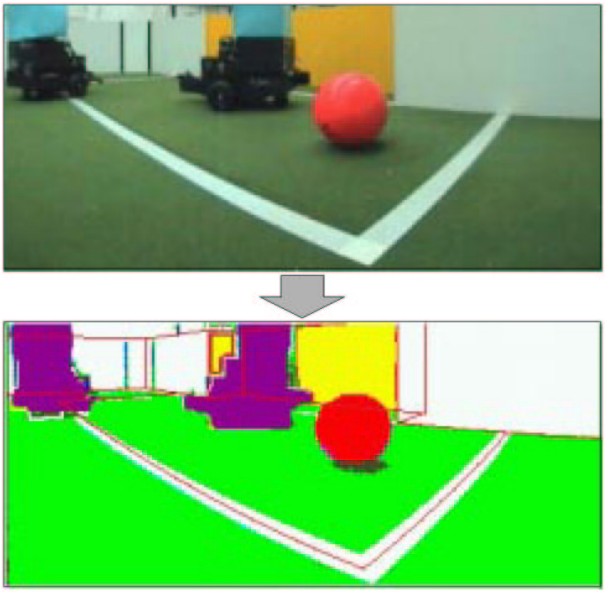


Fig. 3. The figure shows an image captured by the robot and the feature maps that are computed for self, ball, and opponent localization.

algorithm that is used to segment a captured image into colored regions and blobs (see Figure 3). Object detection is performed on the basis of blob analysis. The color segmented image is processed by a feature extraction algorithm (see section VI-C) that estimates the 3D positions with associated uncertainties of the objects of interest. The position of an object is estimated on the basis of a pinhole camera model.

The state estimation subsystem consists of three interacting estimators: the self localization system, the ball estimator, and the opponents estimator. State estimation is an iterative process where each iteration is triggered by the arrival of a new piece of evidence, a captured image or a state estimate broadcasted by another robot. The self localization estimates the probability density of the robot's own position based on extracted environment features, the estimated ball position, and the predicted position. The ball localizer estimates the probability density for the ball position given the robot's own estimated position and its perception of the ball, the predicted ball position, and the ball estimations broadcasted by the other robots. Finally, the positions of the opponents are estimated based on the estimated position of the observing robot, the robots' appearances in the captured images, and their positions as estimated by the team mates.

Every robot maintains a global belief state, which is constructed as follows. The own position, the position of the ball, and the positions of the opponent players are updated by local state estimation processes. The estimated positions of its team mates are the broadcasted results of the self localization processes of the respective team mates. This is done because the accuracy of self localization is much higher than the accuracy of the position estimation for moving objects.

IV. Self-Localization

The self-localization module iteratively estimates, based on a model of the environment, the probability density over the possible robot positions, given the observations taken by the robot. In the remainder of this section we will first describe the environment model used for localization. Then we detail the representation of the robot's position estimate. Finally, we will describe the image pre-processing and the individual steps of the pose estimation process.

A. The Environment Model

The robot's model of the static part of its environment is composed of landmarks together with their positions and orientations. The landmarks themselves are described by 3D curve functions that specify edge curves. Edge curves represent color transitions, that is they separate image regions that differ in the distribution of color vectors. Thus, a 3D curve feature is represented by a pair consisting of a curve function and a color transition.

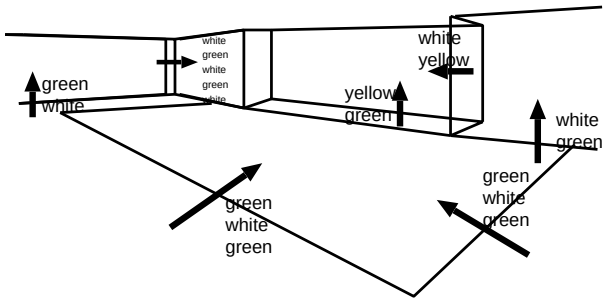


Fig. 4. Model of the neighborhood of a goal. The model contains the edges of the objects and the color transition they are the borderline of.

Figure 4 depicts an excerpt of the environment model representing the neighborhood around a goal, which is used for self localization in the robot soccer domain. The goal is modeled as a set of 3D lines where each line is associated with a color transition.

We use curve functions $C_i : D_i \rightarrow \mathcal{R}^3$ in order to describe the individual features. A curve function C_i defines the set $\mathcal{G}(C_i)$ of curve points in 3D world coordinates by $\mathcal{G}(C_i) = \{C_i(s)|s \in D_i\}$, where $D_i = [s_{i,min}, .., s_{i,max}]$ is the domain for s. All common curves such as circles or B-splines, can be specified or at least approximated by curve functions. A straight line segment is simply given by

$$C_i(s) \quad = \quad s \cdot B_{i1} + (1 - s) \cdot B_{i2} \tag{1}$$

where the points B_{i1} and B_{i2} are the endpoints of the line segment and D_i is equivalent to $[0, .., 1]$. A horizontal circle is defined by

$$C_i(s) \quad = \quad M_i + r \cdot (\sin(s), \cos(s), 0)^T \tag{2}$$

where M_i is the center point, r is the radius, and $D_i = [0, .., 2\pi]$. Line features with multiple color transitions, see Figure 4, are defined by multiple curve functions. Using the world model and a position estimate, the robot can predict where in a captured image lines should be visible and which color transition they represent.

B. The Representation of the Position Estimate

The second data structure used in the self-localization process is the representation of the robot's position. Because the robot is uncertain about its position in the environment, we represent its position estimate using a probability density function that maps the possible positions of the robot into the probability density that the respective position has generated the observations of the robot. We make the assumption that the probability density can be approximated by a multi-variate Gaussian density and model the density using a mean vector and a covariance matrix of the multi-variate Gaussian density.

C. Pre-processing of the Image Data

In mid-size robot soccer all object categories have different colors: the ball is red, field lines are white, and the goals are yellow and blue. These rules allow the application of a simple method for "object recognition": segment the image into colored blobs and identify the blobs based on their color. The color classifier is learned in a training session before tournaments in order to adapt the vision system to the specific light conditions.

The classification of the RGB-16 color vectors is done using a look-up table which combines two advantages: The classification is very fast and no assumptions regarding the shape of class boundaries are required. The look-up table is constructed in a two step process: First, an initial look-up table is obtained from classified training images. This look-up table is sparse since usually most of the 2^{16} RGB vectors are not observed in the classified images. Second, the unclassified RGB vectors are classified using a nearest-neighbor classifier. The Euclidean distances are computed in a 4D space defined by the transformation:

$$\begin{aligned} T(R,G,B) &= \left(\frac{R}{I}, \frac{G}{I}, \frac{B}{I}, \frac{R+G+B}{3C_2}\right)^T \qquad (3)\\ \text{where} \quad I &= (R+G+B)/3 + C_1. \end{aligned}$$

The constant $C_1 > 0$ is chosen to be quite small such that the first three components are roughly independent of the intensity. Only for RGB vector close to black the constant C_1 is important by ensuring a smooth transformation. Other common color spaces such as HSI cause unwanted discontinuities for unsaturated color values. The fourth component of T describes the intensity which is also valuable in order to separate the color classes given in RoboCup. However, within one class usually the intensities vary more than the normalized values, first three elements. Therefore, the intensity is normalized by C_2, e.g. $C_2 = 10$. Figure 3 depicts a captured RGB image (top) and the resulting color labeled image (bottom).

D. The Self-localization Algorithm

The self-localization algorithm (see Algorithm I) running on each robot integrates three different types of data: (1) image data, (2) odometric data, (3) observations made by other teammates. These data are integrated over time to a maximum a posteriori (MAP) estimate of the robot's pose. First, we describe the parts of the self-localization algorithm which are identical

```
algorithm SELFLOCALIZATION()
 1  let (Φ̃, C̃_Φ)    % predicted state mean and covar.
 2     (Φ̂, Ĉ_Φ)    % estimated state mean and covar.
 3     (Φ^l, C^l_Φ)  % last state mean and covar.
 4     projections   % set of projected points.
 5     corr   % correspondences (projections, found points, uncertainty)
 6
 7  do for all data
 8     do (Φ̃, C̃_Φ) ← APPLYMOTIONMODEL(Φ^l, C^l_Φ);
 9        (Φ̂, Ĉ_Φ) ← (Φ̃, C̃_Φ);
10        loop
11           switch (data)
12              case (data is an image) :
13                    projections ← PROJECTCURVEPOINTS(Φ̂);
14                    corr ← SEARCHCORRESP(projections, data);
15                    χ²_I ← QUALITYOFFITIMAGE(corr, Φ);
16                    χ² ← χ̃² + χ²_I;
17              case (data is odometric data) :
18                    χ²_O ← QUALITYOFFITODOM(data, Φ);
19                    χ² ← χ̃² + χ²_O;
20              case (data is ball observation from team-mate) :
21                    χ²_T ← QUALITYOFFITTEAM(data, Φ);
22                    χ² ← χ̃² + χ²_T;
23           Φ̂ ← arg min_Φ χ²;   % one Newton step.
24        until (change of Φ̂ is small or data is not an image)
25        Ĉ_Φ = 2/ Hessian of χ²;
26        (Φ^l, C^l_Φ) ← (Φ̂, Ĉ_Φ);
```

Algorithm I. Self-localization integrating image data, odometric data, and observations of team-mates over time to a MAP estimation.

for all three data types. The three cases corresponding to the three data types (line 12 to 22 in Algorithm I) are explained later. The MAP estimate $\widehat{\mathbf{\Phi}}$ is given by

$$\widehat{\mathbf{\Phi}} = \arg\max_{\mathbf{\Phi}} p(\mathbf{\Phi}) \cdot p(data|\mathbf{\Phi}) \qquad (4)$$

where $p(\mathbf{\Phi})$ is the prior of the pose $\mathbf{\Phi}$ summarizing all evidence gathered in the past. The prior at the current time step is obtained by predicting the pose distribution estimated for the previous time step. Our motion model (line 8) assumes that the robot continues in average with constant rotational and translational velocity. The associated covariance is propagated using a linear approximation of the motion model.

The second term $p(data|\mathbf{\Phi})$ in equation (4) is the likelihood that the robot received $data$ given its current pose $\mathbf{\Phi}$. The computation of the likelihoods corresponding to the three data types (I = image data, O = odometric data, T = data from teammates) is explained below.

The optimization of the product in equation (4) can be transformed into a numerically more favorable optimization of a sum by taking the negative logarithm:

$$\widehat{\mathbf{\Phi}} = \arg\min_{\mathbf{\Phi}} \chi^2 \quad \text{with} \qquad (5)$$

$$\chi^2 = \widetilde{\chi}^2 + \chi^2_{data} \qquad (6)$$

$$\widetilde{\chi}^2 = -2\ln\left(p(\mathbf{\Phi})\right) \qquad (7)$$

$$\chi^2_{data} = -2\ln\left(p(data|\mathbf{\Phi})\right). \qquad (8)$$

The function $\widetilde{\chi}^2$ evaluating the fit between the pose $\mathbf{\Phi}$ and the

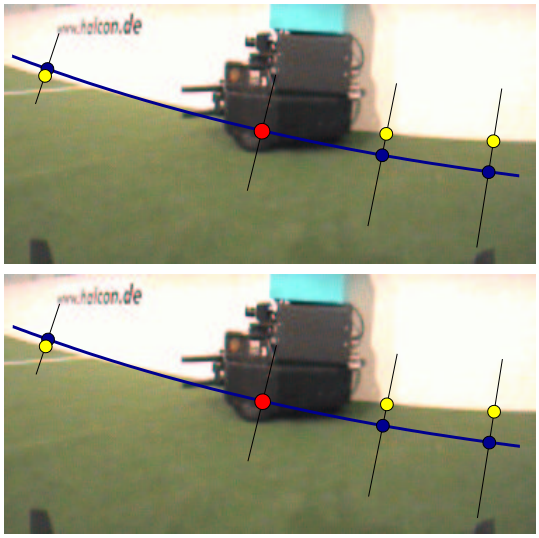


Fig. 5. Search for correspondences: (a) Due to the occlusion caused by the robot, for one projected point no corresponding image point is found with the required color transition. (b) By comparing the line width in the image with the line width of the projection, wrong correspondences (crosses) can be detected.

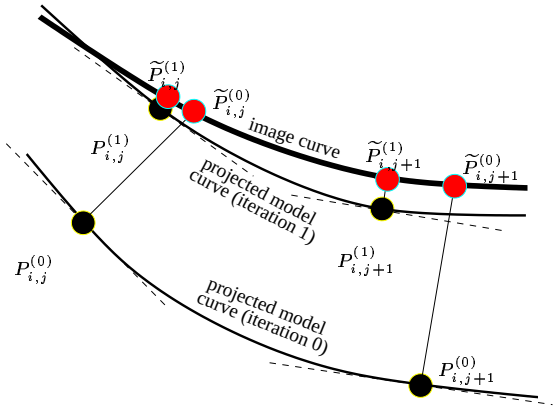


Fig. 6. One iteration step: the new projected model curve is closer to the image curve.

prior is given by

$$\widetilde{\chi}^2 \quad = \quad (\widetilde{\mathbf{\Phi}} - \mathbf{\Phi})^T \widetilde{\mathbf{C}}_{\mathbf{\Phi}}^{-1} (\widetilde{\mathbf{\Phi}} - \mathbf{\Phi}) + \widetilde{C} \qquad (9)$$

where $\widetilde{\mathbf{\Phi}}$ is the mean vector and $\widetilde{\mathbf{C}}_{\mathbf{\Phi}}$ is the covariance matrix of the prior. The variable $\widetilde{C} = \ln \left| 2\pi \ \widetilde{\mathbf{C}}_{\mathbf{\Phi}} \right|$ does not depend on the pose $\mathbf{\Phi}$. Hence, it can be omitted when optimizing χ^2 for $\mathbf{\Phi}$ using Newton iteration [10]. After optimizing χ^2 the covariance matrix $\widehat{\mathbf{C}}_{\mathbf{\Phi}}$ characterizing the uncertainty of the robot's pose can be estimated by

$$\widehat{\mathbf{C}}_{\mathbf{\Phi}} \quad = \quad 2 \, \mathbf{H}^{-1} \qquad (10)$$

where $\mathbf{H}$ is the Hessian of the objective function χ^2 in the estimate $\widehat{\mathbf{\Phi}}$ [10].

In all three cases in Algorithm I a distinct function evaluates the fit between the pose and the latest data. For the case where the data are image data the corresponding function χ_I^2 is obtained by the following steps. In line (13) the 3D curve features that are predicted to be visible are projected into the image plane. In line (14) a local search is performed to establish correspondences between the predicted and detected 3D curve features. In line (15) the function χ_I^2 is obtained by evaluating the distances between the projections and the found image points. Finally, in line (23) the resulting maximum a posteriori criteria is optimized for the robot pose.

1) Projecting 3D Curve Points into the Image: In line (13) points on the 3D curve features that are predicted to be visible are projected into the image plane. The observation of a 3D curve $\mathcal{G}(C_i)$ in the image is a 2D image curve $\mathcal{G}(c_i) = \{c_i(s, \mathbf{\Phi}) | s \in D_i\}$, where $\mathbf{\Phi}$ is the robot pose and c_i is the image curve function given by

$$c_i(s, \mathbf{\Phi}) = \mathit{proj}(C_i(s), \mathbf{\Phi}). \qquad (11)$$

The function *proj* projects a 3D point, given in world coordinates, into the image and returns the pixel coordinates of the resulting 2D point. First, the function *proj* converts the 3D world coordinates into 3D robot coordinates. This conversion depends on the pose $\mathbf{\Phi}$ of the robot. The first two elements of the robot pose $\mathbf{\Phi}$ are the x- and the y-coordinate of the robot position and the third element specifies the robot's orientation. Since we use a calibrated camera mounted on the robot, the 3D robot coordinates can be transformed into 3D camera coordinates and finally, the pixel coordinates of the 2D projection can be computed. The resulting image curve function c_i describes the relation between the robot pose $\mathbf{\Phi}$ and the position of the model curve points in the image.

2) Searching for Correspondences: For each model curve, a small set of projected curve points that lie within the image area is determined. For each of these projected curve points $P_{i,j}$ the algorithm searches in line (14) for color transitions consistent with the transitions specified by the model. To find the respective transition points $\widetilde{P}_{i,j}$ in the image the algorithm locally searches along the perpendiculars of the model curve starting at the projected curve point $P_{i,j}$, see Figure 5a. This search is performed within a fixed search area. Image points outside the search area are regarded as outliers and are omitted. While this is appropriate for the RoboCup environment, search areas adapted to the uncertainty of the curve as in [11] could be beneficial in more complex environments.

For each accepted point $\widetilde{P}_{i,j}$ found in the image, a standard deviation $\sigma_{i,j}$ is used which describes the expected precision of the observation along the perpendicular. In the RoboCup scenario we use for each point $\widetilde{P}_{i,j}$ the same value $\sigma_{i,j}$ which is obtained from experiments. However, for less restricted scenes it might be necessary to use different $\sigma_{i,j}$. For example, individual $\sigma_{i,j}$ could take the image gradient into account.

The lines on the soccer field are treated as double color transitions. The two curve functions defining a line curve are used to predict the line width in the image. By comparing the predicted width with the observed width wrong correspondences can be rejected, see Figure 5b.

3) Evaluating the Fit between an Image and the Robot's Pose: To evaluate the quality of a fit between an image and the robot pose, we define a function χ_I^2. The evaluation given by χ_I^2 is based on the distances between points $\widetilde{P}_{i,j}$ found in the image and the corresponding model curves c_i. The accurate

distance between a point and a curve is defined by a non-linear function and requires usually a non-linear optimization. For efficiency reasons the curve is locally approximated by its tangent. The displacement (signed distance) $d_{i,j}(\mathbf{\Phi})$ between an observed image point $\widetilde{P}_{i,j}$ and the corresponding tangent of the projected model curve c_i is

$$d_{i,j}(\mathbf{\Phi}) = (n_{i,j}(\mathbf{\Phi}))^T \cdot \left[c_i(s_{i,j}, \mathbf{\Phi}) - \widetilde{P}_{i,j}\right] \tag{12}$$

where $n_{i,j}(\mathbf{\Phi})$ is the normal vector of the curve c_i in the point $c_i(s_{i,j}, \mathbf{\Phi})$. The observations $\widetilde{P}_{i,j}$ are noisy. The displacements between the observations $\widetilde{P}_{i,j}$ and the real image curve are assumed to be statistically independent and Gaussian distributed with a mean value of 0 and covariances $\sigma^2_{i,j}$. Under these assumptions, the likelihood for the robot's pose $\mathbf{\Phi}$ is given by

$$p\left(\widetilde{P}|\mathbf{\Phi}\right) = \prod_{(i,j)} \frac{1}{\sqrt{2\pi}\sigma_{i,j}} \exp\left(\frac{-d^2_{i,j}(\mathbf{\Phi})}{2\sigma^2_{i,j}}\right) \tag{13}$$

where $\widetilde{P}$ denotes the set of all curve points $\widetilde{P}_{i,j}$ found in the image. The function χ^2_I evaluating the fit between an image and the robot pose is defined as

$$\chi^2_I = -2 \ln p\left(\widetilde{P}|\mathbf{\Phi}\right). \tag{14}$$

4) Optimizing the Robot's Pose: In line (23) of the self-localization algorithm, a single Newton iteration step [10] minimizing the objective function χ^2 is performed. The Newton step is initialized by the current pose estimate $\widehat{\mathbf{\Phi}}$ and yields a new estimate. Figure 6 illustrates the result of a single iteration step. After one iteration step, the new projected model curves are closer to the observed image points. However, the projected model points $P_{i,j}$ are shifted along the model curves. Hence, these new projected points do not correspond to the initial observations $\widetilde{P}_{i,j}$. Therefore, after each iteration new color transitions are sought along the new perpendiculars defined by the new projections $P_{i,j}$. Since the deviation between image curves and projected model curves is already reduced, the new search can be performed at clearly lower computational cost. The process is iterated until the change of the estimate $\widehat{\mathbf{\Phi}}$ is smaller than a predefined value.

This iterative optimization enables the self-localization process to take non-linearities into account. The non-linearities of the model curve functions are caused by different reasons, such as dependence on the robot's orientation and projection into the image plane. This way the self-localization algorithm can avoid inaccuracies typically yielded by other state estimation algorithms such as the extended Kalman filter, e.g. [12].

For the proposed method usually only few iterations (about two or three) are necessary. Since the computational cost for the self-localization is much smaller than the cost for the blob analysis, see section VI-C, the overall cost increases only slightly by using multiple iterations. In general the higher computational cost is well compensated by a higher accuracy especially in dynamic scenes where predictions are quite uncertain.

5) Incorporating Odometric Data: Similarly to the image data, in line (18) a function χ^2_O is used for evaluating the fit between consecutive odometric sensor readings and consecutive robot poses. From the last two sensor readings the relative motion $\Delta_O = (\Delta_s, \Delta_\phi)^T$ is computed, consisting of the distance Δ_s between the last two positions and the difference Δ_ϕ between the last two orientations. We assume that the errors of Δ_O are zero-mean Gaussian distributed, mutually statistically independent, and statistically independent of the noise corrupting the image measurements. The resulting function χ^2_O is quadratic in the robot's pose. Hence, for the optimization of the objective function, see line (23) in Algorithm I, only one iteration step is needed. The incorporation of ball observations made by other team-mates is explained in the next section.

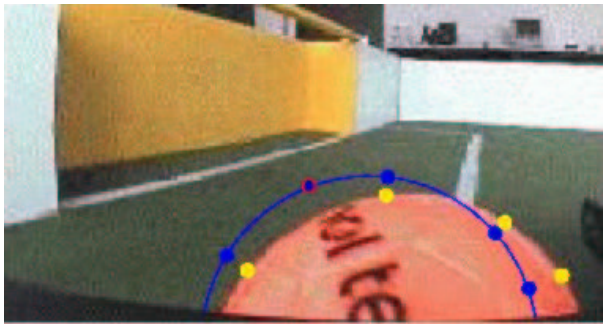


Fig. 7. Search for image points along the perpendiculars. The ball's contour is projected into the image and the algorithm establishes correspondences for several selected points (big filled circles). For one image point no correspondence is found, due to the inscription on the ball.

V. Ball-Localization

A competent soccer player must always know where the ball is. Therefore ball localization is the second state estimation to be performed by the soccer robots. Because the ball is often in the robot's field of view, the ball can also be used for self-localization.

On one hand, the estimated world coordinates of the ball depend on the world coordinates of the observing robot. On the other hand, knowledge of the ball-position can be used for the self-localization of the observing robot. In order to exploit these interdependencies, the self-localization and the localization of the ball are performed simultaneously in a single optimization.

A similar approach is used as for the self-localization. The silhouette of the ball is described by a curve function $c(\mathbf{\Phi})$ (see Figure 7). Here the vector $\mathbf{\Phi}$ to be estimated contains the pose of the observing robot and the position of the ball. The optimization is done (as for the self-localization) simultaneously over all curves, no matter whether a curve feature belongs to the static world model or to the ball. The initialization of the ball position is obtained from an analysis of red blobs. Several consistency checks are applied testing the shape and the relation between the size and the ball distance. The biggest accepted blob is back-projected yielding the initial ball position.

After a robot has updated its estimate $\widehat{\mathbf{\Phi}}$ it broadcasts a second order Taylor approximation χ^2_T of the part of the optimized MAP objective function which depends on the latest observations. A receiving robot R updates its world model by including the approximation χ^2_T into its own estimation. If the same robot

algorithm INTERPRETSENSORDATA($\widehat{\mathbf{\Phi}}, \widehat{\mathbf{C}}_\phi$)

```
let Φ̂   % robot pose
    I   % image data
    R = {r_1, ..., r_{n_k}}   % set of regions
    ω   % augmented mean
    C_ω   % augmented covariance
    Z(k) = {z_1(k), ..., z_{n_k}(k)} % observed feat.

do I ← GETSENSORDATA();
    R ← EXTRACTBLACKREGIONS(I);
    R ← CHECKCONSTRAINTS(R);
    R ← EXTRACTCASCADEDROBOTS(R);
    for i ← 1 to |R|
    do (row, col, width) ← EXTRACTFEATURES(r_i);
        ω ← [Φ̂, row, col, width]^T ;
        C_ω ← ⎛ Ĉ_φ   0       0       0       ⎞
              ⎜ 0     σ_row   0       0       ⎟ ;
              ⎜ 0     0       σ_col   0       ⎟
              ⎝ 0     0       0       σ_width ⎠
        z_i(k) ← UNSCENTEDTRANSFORM(ω, C_ω, opp);
return (Z(k));
```

Algorithm II. The Algorithm used for feature extraction and uncertainty estimation.

algorithm UNSCENTEDTRANSFORMATION($\omega, \mathbf{C}_\omega, f$)

```
let n   % dimension of ω
    ω   % augmented mean
    C_ω   % augmented covariance
    f()   % propagation function
    X = {x_1, ..., x_{2n}}   % set of points
    Y = {y_1, ..., y_{2n}}   % set of prop. points
    ψ̂   % propagated mean
    Ĉ_ψ   % propagated covariance

do % Compute the set X of 2n points
    for i ← 1 to n
    do x_i ← +√(nC_ω);
        x_{n+i} ← −√(nC_ω);
    % Transform each x_i ∈ X to the set Y
    for i ← 1 to 2n
    do y_i ← f(x_i + ω);
    % Compute prop. mean and covariance
    ψ̂ ← (1/2n) Σ_{i=1}^{2n} y_i;
    Ĉ_ψ ← (1/(2n−1)) Σ_{i=1}^{2n} (y_i − ψ̂)^2;
return (ψ̂, Ĉ_ψ);
```

Algorithm III. The unscented transformation.

R afterwards observes the ball, the knowledge of the vague ball position is used in order to refine the own pose. Robot R uses the ball as a dynamic landmark. For the ball a linear motion model is used. However, since a ball prediction in a RoboCup game is very vague the impact on the robot's pose estimate is almost zero unless the robot has no other evidence of its pose.

VI. LOCALIZATION OF OPPONENTS

Localization of opponents considers the following state estimation problem. The world is populated with a set of stationary and moving objects. The number of objects may vary and they might be occluded and out of sensor range. Robots are equipped with sensing routines that are capable of detecting objects within sensor range, of estimating the positions of the detected objects, and of assessing the accuracy of their estimate.

In this section we will describe the opponents tracking algorithm by first detailing the underlying opponents model, then explaining the representation of tracked opponents position estimates, and finally presenting the computational steps of the algorithm: (1) detecting feature blobs in the captured image that correspond to an opponent, (2) estimating the position and uncertainties of the opponent in world coordinates, and (3) associating them with the correct object hypothesis.

A. The Opponents Model

The basic data structure used by the opponents tracking algorithm is the object hypothesis h_i^k. The indices i and k represent the hypothesis number and creation time respectively. An object hypothesis consists of an estimated 2D position, a 2D velocity vector, a diameter, a measure of uncertainty associated with the estimation, and a second measure that represents the degree of belief that this hypothesis accurately reflects an existing object. Because the number of objects might vary new hypotheses might have to be added and old ones might have to be deleted. The set of object hypotheses, $H^k = \{h_1^k, \ldots, h_m^k\}$, represents all valid hypotheses at time k.

B. The Representation of Opponent Tracks

When tracking the positions of a set of opponent robots there are two kinds of uncertainties that the state estimator has to deal with. The first one is the inaccuracy of the robot's sensors. We represent this kind of uncertainty using a Gaussian probability density. The second kind of uncertainty is introduced by the data association problem, i.e. assigning feature blobs to object hypotheses. This uncertainty is represented by a hypotheses tree where nodes represent the association of a feature blob with an object hypothesis. A node h_j^k is a son of the node h_i^{k-1} if h_j^k results from the assignment of an observed feature blob with a predicted state $\widetilde{h}_i^{k-1}$ of the hypothesis h_i^{k-1}. In order to constrain the growth of the hypotheses tree, it is pruned to eliminate improbable branches with every iteration of the opponent tracking algorithm.

C. Feature Extraction and Uncertainty Estimation

This section outlines the feature extraction process (see Algorithm II) which is performed in order to estimate the positions and the covariances of the opponent team's robots. Each opponent robot is modeled in world coordinates by a bi-variate Gaussian density with mean and covariance matrix.

We assume that the opponent robots are colored black and have approximately circular shape. Friend foe discrimination is enabled through predefined color markers (cyan and magenta, see Figure 3) on the robots. Each marker color may be assigned to any of the two competing teams. Consequently it is important that the following algorithms can be parameterized accordingly. Furthermore, we assume that the tracked object almost touches the ground (see Figure 8).

1) Extraction of Blobs Containing Opponent Robots: The following procedure extracts a set of regions from a captured image, where each region corresponds to a currently visible robot. After capturing an image (see Algorithm II line 8) the black color-regions are extracted in line (9) using color classification and morphological operators. In order to be recognized as an opponent robot a black blob has to satisfy several constraints (line 10), e.g. a minimum/maximum size and a red or green color-region adjacent to the bottom region row. These constraints enable the routine to distinguish robots from black logos and adverts affixed on the wall surrounding the field. Furthermore blobs that contain or have a color-region of the own team color in the immediate neighborhood can be discarded.

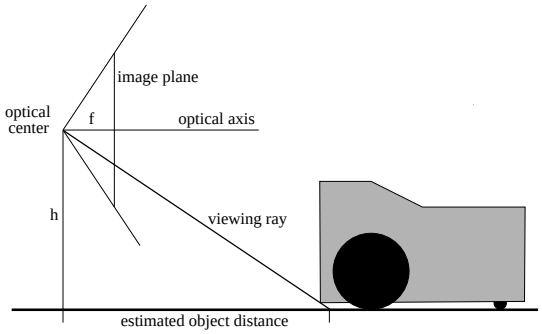


Fig. 8. An estimate of the robot's distance is given through the intersection of the viewing ray with the ground plane of the field.

In the next step the physical size of the object corresponding to a blob is examined. For every extracted region the object's physical diameter is estimated. If it exceeds an upper threshold, it is assumed that two robots are directly next to each other. In this case the blob is divided into two.

In order to detect cascaded robots (line 11), i.e. opponent robots that are partially occluded by other robots, our algorithm uses discontinuities in row width. As soon as the length of a row differs significantly from the length of its lower predecessor and the respective world coordinates is more than 10 cm above the ground it is assumed that a partly occluded object is detected. However before the region can safely be split into two, the resulting subregions have to obey several further constraints, e.g. minimum size.

Finally, for every extracted region three features are computed (line 13): The bottom most pixel row, the column col representing the center of gravity, and a mean blob $width$. For the latter two features only the three bottom most rows which exceed a certain length are used. In order to determine these rows, we allow also for occlusion through the ball.

Empirically we found that this feature extraction procedure is sufficient to determine accurate positions of opponent robots. Mistakenly extracted objects are generally resolved in a fast manner by the opponent tracking algorithm.

2) Estimation of Opponent Position and Uncertainty: In this section we discuss how the position and the respective covariance of an observed robot is estimated. This step takes the estimated pose and the covariance of the observing robots as well as position of the detected feature blob in the image and the associated measurement uncertainties into account.

We define a function *opp* that determines the world coordinates of an opponent robot based on the pose estimate $\widehat{\mathbf{\Phi}}$ of the observing robot, the pixel coordinates row, col of the center of gravity and the $width$ of the opponent robot's blob. Due to rotations and radial distortions of the lenses *opp* is non-linear. First the function *opp* converts the blob's pixel coordinates to relative polar coordinates. On this basis and the width of the observed blob the radius of the observed robot is estimated. Since the polar coordinates only describe the distance to the opponent robot but not the distance to its center, the radius is added to the distance. Finally the polar coordinates are transformed into world coordinates taking the observing robot's pose estimate $\widehat{\mathbf{\Phi}}$ into account.

In order to estimate the position $\widehat{\psi}$ and the covariance $\widehat{\mathbf{C}}_{\psi}$ of an opponent robot, we will use a technique called the unscented transformation [13]. This transformation allows the efficient propagation of uncertainties without creating the necessity to derive the partial derivatives of the propagation functions. Julier and Uhlmann also proved that the unscented transformation provides more realistic uncertainty estimations than a standard Taylor approximation.

In order to apply the unscented transformation, an augmented mean ω (line 14) and covariance $\mathbf{C}_{\omega}$ (line 15) describing jointly the observing robot's pose estimate and the observed robot features is set up. $\widehat{\mathbf{\Phi}}$, row, col and $width$ are assumed to be uncorrelated with variances σ_{row}, σ_{col} and σ_{width}. These sigmas are dependent on the image processing hardware and can be determined from a series of experiments.

The unscented transformation (line 16) is then applied to the augmented mean and covariance using the non-linear mapping *opp*. This yields the opponent robot's position $\widehat{\psi}$ and uncertainty $\widehat{\mathbf{C}}_{\psi}$, which are stored in the element $z_i(k)$ of the feature vector $Z(K)$.

In Figure 9 the uncertainties of objects depending on the uncertainty of the observing robot and their relative distances are displayed using σ-contours. For illustrative purposes the uncertainty ellipses are scaled by an order of five. Each robot observes two obstacles in 3.5 and 7 meters distance. Robot Odilo is very certain about its pose estimate and thus the covariance of the observed robot depends mainly on its distance. Robot Grimoald has a high uncertainty in its orientation ($\approx$ 7 degrees). Consequently the position estimate of the observed obstacle is less precise and highly influenced by the orientation uncertainty.

Finally the extracted features $z_i(k)$ are associated with the predicted hypotheses $\widetilde{h}_j^k$ by the multiple hypothesis tracking algorithm presented in the subsequent section.

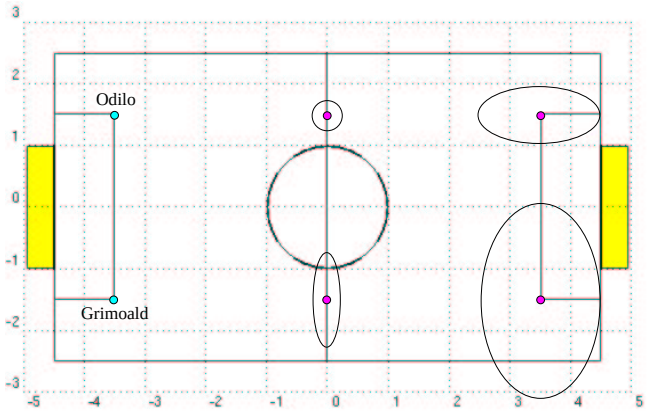


Fig. 9. Propagation of uncertainties. For illustrative purposes the uncertainty ellipses are scaled by an order of five.

D. Multiple Hypothesis Tracking

For associating new observations with existing object hypotheses we use a variant of Reid's Multiple Hypothesis Tracking (MHT) algorithm [14]. The objective of the MHT algorithm is to keep a set of object hypotheses, each describing a unique real object and its position, to maintain the set of hypotheses over time, and to estimate the likelihood of the individual hypotheses. We demonstrate how the MHT framework can be applied to model dynamic environments in multi-robot systems and equip it with mechanisms to handle multiple mobile sensors with uncertain positions.

Before we dive into the details of the MHT algorithm let us first get an intuition of how it works. The MHT algorithm maintains a forest of object hypotheses, that is a set of trees. The nodes in the forest are object hypotheses and represent the association of an observed object with an existing object hypothesis. Each hypothesis has an association probability, which indicates the likelihood that observed object and object hypothesis refer to the same object. In order to determine this probability the motion model is applied to the object hypothesis of the previous iteration, in order to predict the object's position. Then the association probability is computed by weighing the distance between the predicted and the observed object position. Thus in every iteration of the algorithm each observation is associated with each existing object hypothesis.

The computational structure of the algorithm is shown in Algorithm IV. An iteration begins with the set of hypotheses of object states $H^k = \{h_1^k, \ldots, h_m^k\}$ from the previous iteration k. Each h_i^k is a random variable ranging over the state space of a single object and represents a different assignment of measurements to objects, which was performed in the past. The algorithm maintains a Kalman filter for each hypothesis.

With the arrival of new sensor data (7), $Z(k+1) = \{z_1(k+1), \ldots, z_{n_{k+1}}(k+1)\}$, the motion model (8) is applied to each hypothesis and intermediate hypotheses $\widetilde{h}_i^{k+1}$ are predicted. For this step usually a constant velocity model is used. Assignments of measurements to objects (11) are accomplished on the basis of a statistical distance measurement, such as the Mahalanobis distance. Each subsequent child hypothesis represents one possible interpretation of the set of observed objects and, together with its parent hypothesis, represents one possible interpretation of all past observations. With every iteration of the MHT, probabilities (12) describing the validity of hypotheses are calculated. Furthermore for every observed object an new hypothesis with associated probability is created (14).

In order to constrain the growth of the hypothesis trees the algorithm prunes improbable branches (15). Pruning is based on a combination of ratio pruning, i.e. a simple lower limit on the ratio of the probabilities of the current and best hypotheses, and the N-scan-back algorithm [14]. This algorithm assumes that any ambiguity at time k is resolved by time $k+N$. Consequently if at time k hypothesis h_i^{k-1} has m children, the sum of the probabilities of the leaf notes of each branch is calculated. The branch with the greatest probability is retained and the others are discarded. After pruning the world state of X^{k-N} can be extracted (16). Please note that this world state is always N steps delayed behind the latest observations. However, in section VI-F we will demonstrate that this delay can be overcome by N observers performing observations in parallel.

algorithm MULTIPLEHYPOTHESISTRACKING()

```
1  let H̃^k = {h̃_1^k, ..., h̃_{m_k}^k}   % predicted hyps.
2     Z(k) = {z_1(k), ..., z_{n_k}(k)} % observed feat.
3     H^k = {h_1^k, ..., h_{o_k}^k}   % new hyps.
4     X^{k-N}  % world state at time step k-N.
5
6  do for k ← 1 to ∞
7     do Z(k) ← INTERPRETSENSORDATA();
8        H̃^k ← APPLYMOTIONMODEL(H^{k-1}, M);
9        for i ← 1 to n_k
10       do for j ← 1 to m_k
11          do h_ij^k ← ASSOCIATE(h̃_j^k, z_i(k));
12             COMPUTE(P(h_ij^k | Z(k)));
13       for j ← 1 to n_k
14       do H^k ← H^k ∪ {GENERATENEWHYP(z_j(k))};
15       PRUNEHYPOTHESES(H^{k-N});
16       X^{k-N} ← {h_1^{k-N}, ..., h_{o_{k-N}}^{k-N}};
```

Algorithm IV. The Multiple Hypothesis Tracking Algorithm.

E. Computing the Likelihood of Hypotheses

Obviously, the heart of the MHT algorithm is the computation of the likelihood of the different hypothesis-observation associations, $P(h_{ij}^{k+1}|Z(k))$, in line 12 of the algorithm in Algorithm IV. In this section we derive the formula that is used in order to compute this probability. The derivation of this formula is critical because it tells us which probabilities must be specified by programmers in order to apply the algorithm to specific applications.

Let Z^k be the sequence of all measurements up to time k. A new hypothesis of an object at time k is made up of the current set of assignments (also called an event), $\theta(k)$, and a previous state of this hypothesis, h_j^{k-1}, based on observed features up to time step $k-1$ inclusively.

We can transform the probability of an object's hypothesis $P(h_i^t|Z^k)$ using Bayes' rule and the Markov assumption in order to obtain an easier expression:

$$\begin{aligned} P(h_i^k|Z^k) &= P(\theta(k), h_i^{k-1}|Z(k), Z^{k-1}) && (15)\\ &= P(\theta(k), h_i^{k-1}|Z(k), H^k) && (16)\\ &= \alpha * p(Z(k)|\theta(k), h_j^{k-1}, Z^{k-1}) && (17)\\ & \quad P(\theta(k)|h_j^{k-1}, Z^{k-1}) P(h_j^{k-1}|Z^{k-1}). \end{aligned}$$

Here α is a normalization factor ensuring that $P(h_i^k|Z^k)$ sums up to one over all h_i^k. The last term of this equation is probability of the parent global hypothesis that has been computed in the previous iteration. The second factor can be evaluated as follows [15]:

$$P(\theta(k)|h_j^{k-1}, Z^{k-1}) = \frac{\phi!\nu!}{m_k!}\mu_F(\phi)\mu_N(v) \qquad (18)$$
$$\prod_t (P_D^t)^{\delta_t}(1-P_D^t)^{1-\delta_t}(P_T^t)^{\tau_t}(1-P_T^t)^{1-\tau_t}$$

where $\mu_F(\phi)$ and $\mu_N(\nu)$ are prior probability mass functions of the number of spurious measurements, ϕ, and new geometric features, ν. In generally $\mu_F(\phi)$ and $\mu_N(\nu)$ are assumed to be Poisson distributed with mean λ_F and λ_N. P_D^t and P_T^t are the probabilities of detection and termination of track t (originating from hypothesis h_t^{k-1}) and δ_t and τ_t are indicator variables. δ_t (τ_t) is 1, if track t is detected (deleted) at time k and 0 otherwise.

The indicator variable δ_t depends on the observing robots camera orientation. It is 1, if the track t is within the sensor's field of perception and track t is not occluded by another team mate. $P_T^{\tau_t}$ is used to model the declination of an unobserved hypothesis probability over time. It is defined as $P_T^{\tau_t} = 1 - e^{-\frac{\Delta k}{\lambda_T}}$. Δk is the number of consecutive time steps a hypothesis was not observed. λ_T determines the speed of the declination process. Larger λ_T result in a slower declination of the hypothesis probability.

The first term on the right hand side of equation 17 denotes the association probability of a measurement and a hypothesis. In order to determine this term it is assumed that a measurement z_i^k has a Gaussian probability density function if it is associated with object t_i.

$$N_{t_i}(z_i(k)) = N_{t_i}(z_i(k), \widetilde{h}_i(k), S_{t_i}(k)) = \qquad (19)$$
$$((2\pi)^2|S_{t_i}(k)|)^{-\frac{1}{2}} e^{-\frac{1}{2}\{(z(k)-\widetilde{h}(k))^T \{S_{t_i}(k)\}^{-1}(z(k)-\widetilde{h}(k))\}}.$$

Here $\widetilde{h}_i(k)$ denotes the predicted measurement for hypothesis t_i and $S_{t_i}(k)$ is the associated innovation covariance. The probabilities of a new object and a spurious measurement are taken to be uniformly distributed over the observation volume V. In our implementation the observation volume V is the intersection of the field of view (neglecting occlusions) and the soccer field. Thus V is a function of the robot's pose estimate and the camera's field of view.

$$p(Z(k)|\theta(k), h_j^{k-1}, Z^{k-1}) = \prod_i^{m_k} [N_{t_i}(z_i(k))]^{\kappa_i} V^{-(1-\kappa_i)} \qquad (20)$$

The quantity κ_i is another indicator variable which is 1, if $z_i(k)$ came from a known track and 0 otherwise.

F. Applying MHT to Autonomous Robot Soccer

In the following we will discuss the remaining parameters of the MHT algorithm. The probability of detection P_D represents the probability that a track is detected by an observing robot, if it is within it's field of view. During the experiments in section VII we have set P_D to 0.9.

By default we set the termination likelihood λ_T to 40. This allows an unconfirmed hypothesis to survive for approx. 2 seconds. Remembering objects that are currently not visible is important to avoid collisions and thereby reduce the risks of being charged for fouls. The mean rates of false observations λ_F and new tracks λ_N are 0.0002 and 0.04 respectively.

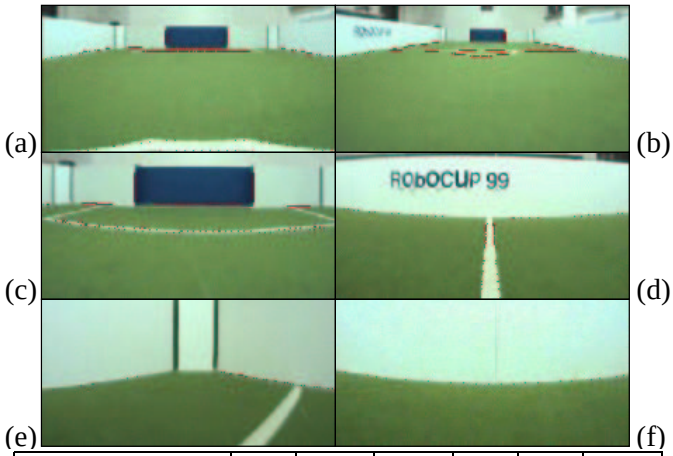
The depth N of the tracking tree is set to four. This is the minimal depth that allows each team mate to contribute an observation to the hypothesis fusion process. The empirical investigations will show that the update-times of the MHT within our application domain are fast enough to handle the observations of all four robots. Thus the first set of global hypotheses, after the initialization of the MHT, is already created after every robot has performed only one local opponent observation. Since these observations are performed in parallel and integrated through the MHT in real time (before the next opponent observations are performed), a depth of four contributes sustainable to the quality of the global opponent hypotheses. The maximum number of hypothesis was limited to 50 and the value for ratio pruning was set to 0.001.

VII. Empirical Investigation

The presented algorithms are applied in our middle-size RoboCup team, the AGILO RoboCuppers. The cooperative localization approach was successfully applied since 1999 during several RoboCup World Championships and several local German contests. The multiple object tracking algorithm described in this paper has been employed by our robot soccer team for the first time in the fifth robot soccer world championship in Seattle (2001). In 2001 the team has played six games for a total of about 120 minutes. The team advanced to the quarter finals.

Unfortunately, in the middle size robot soccer league there is no external sensing device which records a global view of the game and can be used as the ground truth for experiments. Thus for the experimental results in this section we can only use the subjective information of our robots and argue for the plausibility of their behavior and belief states. To do so, we have written log files and recorded the games using video cameras in order to evaluate our algorithm.

Every robot of the soccer team is equipped with its own computer (Pentium III 500 MHz). On this computer several processes are running in order to perform image processing, path planning, action selection and object tracking. Currently every robot processes approx. 15 frames per second. From every frame a pose estimate, a ball estimate and a set of opponent observations is extracted and sent to all other robots of the team via wireless Ethernet. Every robot iterates the opponent tracking algorithm once for every set of opponent observations (own and team mates) and computes a new estimate of the world state. This estimated state serves as input for action selection



Mean pose errors	(a)	(b)	(c)	(d)	(e)	(f)
$\mathbf{\Phi}_x$ (cm)	0.4	26.0	1.9	0.3	4.3	>99
$\mathbf{\Phi}_y$ (cm)	1.0	20.3	15.5	8.0	3.2	2.3
$\mathbf{\Phi}_\varphi$ (degree)	1.1	4.1	8.1	2.4	7.5	2.8

Fig. 10. The robot's views from 6 different poses on the pitch. Mean error of the self-localization for the $x-/y$-coordinates, and the rotation angle φ.

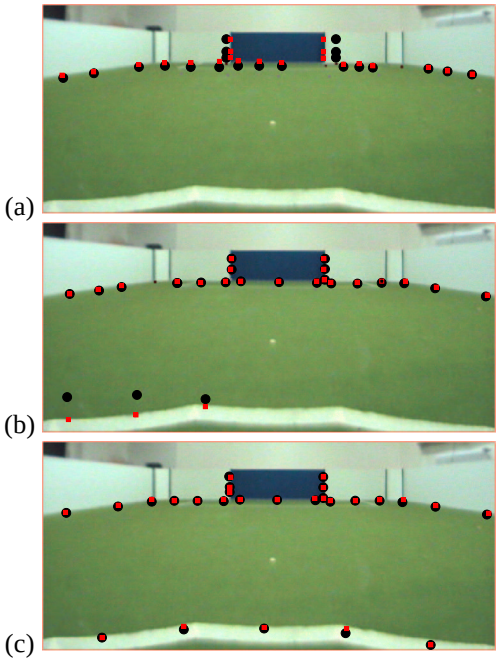


Fig. 11. Illustration of a sequence of iteration steps. Dots indicate projected model points, squares depict corresponding image points. (a) Initialization. (b) The result of one iteration step is equivalent to the estimate of an extended Kalman filter. (c) Further iterations can yield significant improvements.

and path planning. When the robots planning and action selection algorithms' are turned off the vision system and opponent tracking algorithm are easily able to cope with the maximum frame rate of our camera (25 fps).

A. *Cooperative Self-Localization*

In general it was found that the localization algorithm runs with a mean processing time of 18 msec for a 16-Bit RGB ($384*288$) image. Only for 4% of the images the processing time exceeds 25 msec [16].

In the first experiment the accuracy of the vision-based self-localization is investigated. Odometric data and data from team-mates are not used and the state noise is set to very high values. This mainly eliminates the correlation between consecutive pose estimates. A robot is set up at six different positions (see Figure 10) and the self-localization algorithm is run for about 30 seconds. From about 750 pose estimates the mean error of the x-/y-coordinates and the rotation angle are computed and displayed in Figure 10. The accuracy of the estimated pose depends on the features visible and their distance to the camera. The poses (a), (d) and (e) allow a quite accurate pose estimation. In pose (b) the visible features are far away and therefore, the resulting estimate is less accurate. The problem of pose (c) is that an error in the y-coordinate can be compensated by the rotation angle, and vice versa. From pose (f) only one edge is visible which is not sufficient in order to determine all three pose parameters. In this case data-fusion with e.g. odometric data is needed to overcome this problem. In areas where precise robot movements are required, e.g. near the goals, enough features for robust vision-based pose estimation are present.

An analysis of the errors shows that the standard deviations of the pose estimates is small compared to the actual errors, usually less than 15%. The bias of the pose estimate indicates that the impact of image noise is quite small compared to systematic errors. Such errors may be caused for example by inaccuracies of the camera calibration or the environment model as well as unevenness of the floor.

Figure 11 depicts the results for a sequence of iteration steps. The result after one iteration step (see Figure 11a and b) is identical to the estimate of an extended Kalman filter. This estimate is substantially improved by our non-linear method using multiple iteration steps (see Figure 11b and c).

In the next experiment we examine the capability to fuse visual and odometric data over time. A robot starts at the top left corner of the playing field and moves along an 8-shaped trajectory across the field, see Figure 12a. Starting- and ending-point of the trajectory are the same. The four major turning points are at the corners of the penalty-area ($x = +/-3$ m, $y = +/-1.25$ m). The first trajectory is exclusively derived from the vision based pose estimates. The second trajectory is obtained by dead-reckoning. The third trajectory is the result of the fusion process combining data from both sensors. The weaknesses of the pure vision-based localization and dead-reckoning approach are clearly visible. For the vision-based approach comparatively fast changes of the estimated pose may happen if the observed features change. Generally this happens only when too few or too distant features are visible. This can be observed when the robot crosses the field diagonal and does not see the middle-line and center-circle anymore (see Figure 12a). The dead-reckoning trajectory exhibits the clear error accumulation over time, e.g. starting-and ending-point of the trajectory are about 80 centimeters apart. The approach based on both data types is able to cope with both weaknesses. Vision based pose inaccuracies are compensated by the relative movements derived from the odometric data. The accumulation of the dead-reckoning error is corrected by the vision based pose estimates.

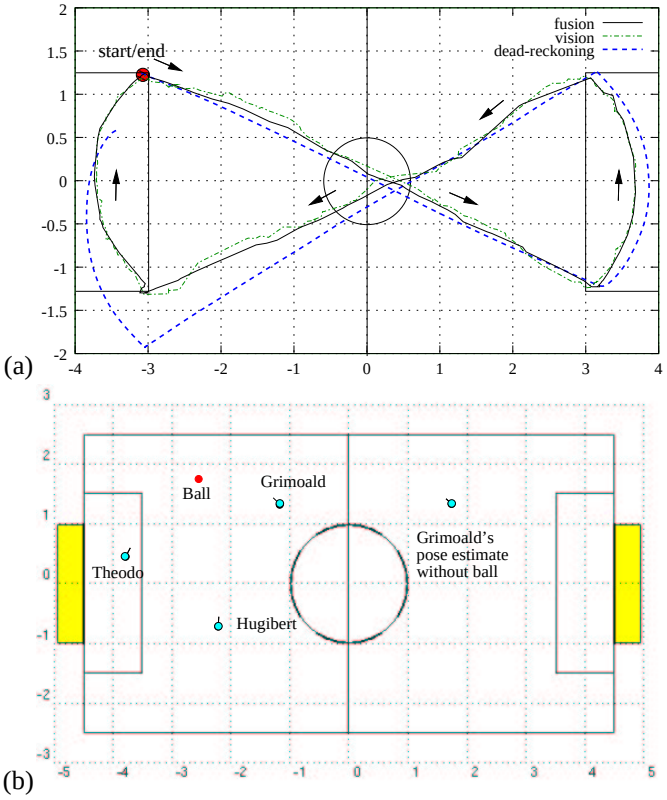


Fig. 12. (a) Data-fusion of vision and dead-reckoning position estimates. (b) Set-up for the cooperative localization experiment.

In the third experiment the robots sensor data fusion is tested. Three robots are put at predefined positions on the field. After the robots have estimated their pose, a red ball is put on the field, such that all robots can see the ball. Figure 12b outlines the experimental set-up. Table I gives a brief summary of the results. For every robot three rows are given. The first row contains the robot's exact pose and the exact ball position. In the second row the robot's pose estimate and the corresponding uncertainties are displayed. The third row shows the pose and position estimates of the ball after it was put on the field. All positions are given in Cartesian world coordinates and measured in meters, the orientations are measured in degrees. Robots Theodo and Hugibert estimate their poses correctly with high accuracies. Grimoald can only detect the edge between the field and the top boundary. Thus, it estimates y and φ correctly but sees itself too far to the right. After the ball was put on the field Grimoald receives the ball's position estimates from the other two robots, and applies them to correct its own pose. This experiment was also performed several times with moving robots and equally good results. This feature of our algorithm will gain even more importance from 2002 on, when the walls surrounding the RoboCup field will be removed.

B. Cooperative Opponent Tracking

The analysis of the log files from RoboCup 2001, revealed that an average iteration of the opponent tracking algorithm for a set of opponent observations takes between 6 to 7 msecs [17], [18]. This allows our implementation to process the observations of all robots (max. frame rate: 25Hz) in real time. The

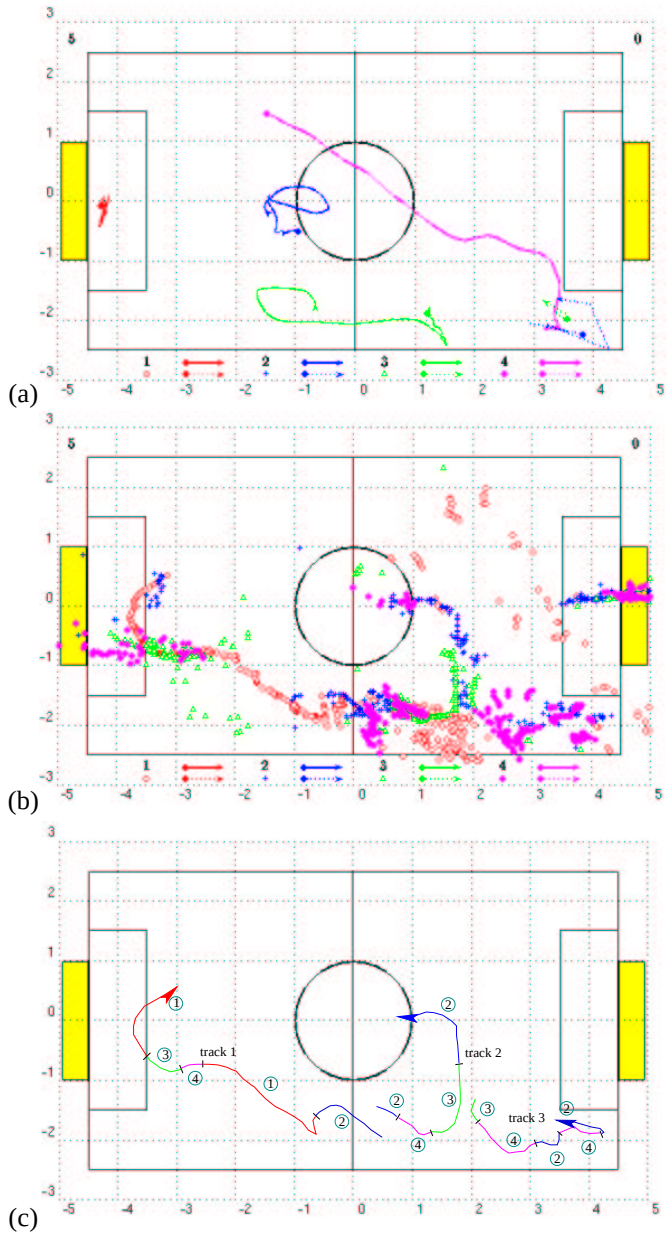


Fig. 13. (a) The trajectories of four AGILO robots, (b) their opponent observations and (c) the resolved tracks.

minimum and maximum iteration times were measured to be 1.1 msecs and 86 msecs respectively. On average 3.2 opponents were tracked. This is a reasonable number since there are maximal 4 opponent players and players can be send off or have to be rebooted off the field. In breaks of the games (when people entered the field) or when there are crowds of robots the opponent tracking algorithm tracked successfully up to 11 objects.

A typical result of the AGILO game state estimator is shown in Figure 13. The upper picture shows the positions of the AGILO players of the own team, computed through vision-based self localization. The middle picture shows the individual observations of the opponent robots. The colors of the observations indicate which AGILO robot made the observation. You have to look at the color figures in order to see the individual

Robot	x	y	φ	ball x	ball y	σ_x	σ_y	σ_φ	σ_{ballx}	σ_{bally}
Theodo	-3.80	0.45	40	-2.30	1.60	—	—	—	—	—
without ball	-3.80	0.43	39	—	—	.0433	.0563	1.439	∞	∞
with ball	-3.77	0.47	40	-2.26	1.63	.0396	.0472	1.106	.0767	.0731
Hugibert	-2.10	-0.80	90	-2.30	1.60	—	—	—	—	—
without ball	-2.13	-0.94	92	—	—	.1130	.0924	8.738	∞	∞
with ball	-2.12	-0.78	90	-2.31	1.63	.0941	.0872	5.072	.1057	.0937
Grimoald	-1.15	1.30	140	-2.30	1.60	—	—	—	—	—
without ball	1.77	1.25	138	—	—	∞	.0149	2.753	∞	∞
with ball	-1.14	1.27	140	-2.31	1.58	.1053	.0143	2.544	.0863	.0146

TABLE I

LOCALIZATION RESULTS WITH AND WITHOUT FEATURES OBSERVED BY SEVERAL ROBOTS.

observations of the different robots and how they are merged into a consistent track. In the lower picture the tracks as they were resolved by the tracking algorithm are displayed. They are divided into subsections. The number of the robot that contributed the most observations to this part of the track is denoted next to the track.

Qualitatively, we can estimate the accuracy of the game state estimation by looking for jumps in the tracked lines. We can see that the tracks of own robots are smooth and can therefore be expected to be accurate. The tracks of the opponent robots look very reasonable too. They are less accurate and sometimes only partial. This is due to the high inaccuracy and incompleteness of the sensory data. However, it is observable that several tracks resulted from merging the observations of different robots. In addition, the merging of the different observations results in fewer hallucinated obstacles and therefore allows for more efficient navigation paths. Several wrong opponent observations made by the goal keeper (1) were correctly omitted by the opponent tracking algorithm and not assigned to a track. We have cross checked the tracks computed by the algorithm using video sequences recorded during the matches. The tracks are qualitatively correct and seem to be accurate. A more thorough evaluation is only possibly based on the ground truth for the situations. We are currently implementing tracking software for a camera mounted above the field that allows us to compute the ground truth for the next RoboCup championship.

C. The Effect of Cooperation

The cooperation of the different robots increases both, the completeness and the accuracy of state estimation. Accuracy can be substantially increased by fusing the observations of different robots because the depth estimate of positions are much more inaccurate than the lateral positions in the image. This can be accomplished through the Kalman filter's property to optimally fuse observations from different robots into global hypotheses with smaller covariances.

The completeness of state estimation can be increased because all the robots can see only parts of the field and can be complemented with observations of the team mates. The other effect we observed was that cooperation allowed to maintain the identity of opponent players over an extended period of time, even though the field of view of the observing robots is limited. This point is well illustrated in Figure 13. The three opponent field players were tracked successfully over a period of 30 seconds.

VIII. RELATED WORK

Related work comprises work done on vision-based localization and object tracking conducted in- and outside the robot soccer domain.

In the robot soccer domain algorithms for probabilistic self-localization have been proposed. Gutmann et al. [19] have proposed a self localization method based on a Kalman filter approach by matching observed laser scan lines into the environment model. We differ from this approach mainly by using vision data instead of laser data. The main challenge for vision algorithms is, that they have to cope with the larger amount of data. Further approaches to vision-based self localization using directional and omnidirectional cameras can be found in [20], [21], [22], [23], [24]. Most of them are data driven, e.g. apply a Hough transformation or other computational expensive feature extraction technique to the complete image, whereas our approach is model driven and requires only the evaluation of a few pixels in the vicinity of a projected model feature point. Blake and Isard [11] also perform model driven localization. However they restrict themselves to weak perspective and linear models. By employing an iterative optimization, our algorithm further extends the extended Kalman filter approach [12] such that it can handle non-linear models more accurately.

Enderle et al. [25], [24] have developed a vision-based self-localization module using a sample-based Markov localization method, that is also known as Monte Carlo localization (MCL). In Fox et al. [26] an extension of this approach to multi-robot systems is proposed. The advantage of MCL is that no assumption about the shape of the probability distribution is made. However, in order to achieve high accuracy, usually a large number of samples is required. The sample set converges to the true posterior as the number of samples n goes to infinity with a convergence speed of $O(\frac{1}{\sqrt{n}})$ [27], [28]. Hence, MCL leads to limited accuracy and/or relatively high computational cost. The self localization method that is proposed here has the advantage that it is fast as it utilizes Newton iteration with quadratic convergence speed. A similar iterative optimization of a MAP criterion is also employed in [29].

Roumeliotis and Bekey [30] present an approach in which sensor data from a heterogeneous collection of robots are combined through a single Kalman filter to estimate the pose of each robot in the team. Further works [31], [32] have described approaches in which robots actively coordinate their movements in order to reduce cumulative odometric errors. While all these methods rely on the capability of robots to detect and identify each other correctly, our approach is able to exploit knowledge about other objects, which do not belong to the team.

To the best of our knowledge no probabilistic state estimation method has been proposed for tracking the opponent robots in robot soccer or similar application domains. Dietl and Gutmann [33], [19] estimate the positions of the opponents and store them in the team world model but they do not probabilistically integrate the different pieces of information. Probabilistic tracking of multiple moving objects has been proposed by Schulz and Burgard [6]. They apply sample-based joint probabilistic data association filter (SJPDAF) estimation to the tracking of moving people with a moving robot using laser range data. In contrast to the SJPDAF the MHT requires less computational power if the pruning parameters are carefully selected. Hue et al. [34] also track multiple objects with particle filters. In their work data association is performed on the basis of the Gibbs sampler. Our approach to multiple hypothesis tracking is most closely related to the one proposed by Cox and Miller [35]. Cox and Leonard [36] use multiple hypothesis tracking to model a static environment consisting of corners and walls. We extend their work on multiple hypothesis tracking in that we apply the method to a more challenging application domain where we have multiple moving observers with uncertain positions. In addition, we perform tracking at an object level rather than at a feature level. Further applications where multiple hypothesis tracking is used include active global localization of a robot within a topological map [37] and navigation of an autonomous system in unknown environments [38].

IX. Conclusions

In this paper, we have developed and analyzed a probabilistic, vision-based state estimation method for individual, autonomous robots. This method enables a team of mobile robots to estimate their joint positions in a known environment and track the positions of autonomously moving objects. Our results suggest that purely image-based probabilistic estimation of complex game states is feasible in real time even in complex and fast changing environments.

The state estimators of different robots cooperate to increase the accuracy and reliability of the estimation process. This cooperation between the robots enables them to track temporarily occluded objects and to faster recover their position after they have lost track of it. This is possible because the estimation method is able to exploit the interdependencies between self-localization and the localization of other observed objects.

The vision-based state estimation method runs within framerate. This high speed is achieved by first, focusing on selected image points in the vicinity of predicted model points and secondly, by efficiently fusing information over time using an extension of the Kalman filter. In contrast to the extended Kalman filter, we explicitly take the nonlinearity of the measurement equations into account. This leads to high accuracies and good predictions.

We have empirically validated the method based on experiments with a team of physical robots. These experiments have shown that cooperative localization leads to a higher localization accuracy. Furthermore, our method allows for solving certain localization problems that are unsolvable for a single robot. For example, one robot can resolve ambiguities in position estimates by taking into account the estimates provided by other robots.

Acknowledgments

We would like to thank Ingemar J. Cox and Matthew Miller of the NEC Research Institute for providing their Multiple Hypothesis Tracking implementation. The research reported in this papers is partly funded by the Deutsche Forschungsgemeinschaft (DFG) under contract Ra-359/71 (”Semi-automatic acquisition of visuomotoric plans”).

References

[1] S. Thrun, M. Beetz, M. Bennewitz, A.B. Cremers, F. Dellaert, D. Fox, D. Hähnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz, “Probabilistic algorithms and the interactive museum tour-guide robot minerva,” *International Journal of Robotics Research*, vol. 19, no. 11, pp. 972–999, 2000.

[2] W. Burgard, A.B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun, “Experiences with an interactive museum tour-guide robot,” *Artificial Intelligence*, vol. 114, no. 1-2, 2000.

[3] R. Simmons, R. Goodwin, K. Haigh, S. Koenig, J. O’Sullivan, and M. Veloso, “Xavier: Experience with a layered robot architecture,” *ACM magazine* Intelligence, 1997.

[4] S. Thrun, “Probabilistic algorithms in robotics,” *AI Magazine*, vol. 21, no. 4, pp. 93–109, 2000.

[5] D. Fox, W. Burgard, and S. Thrun, “Markov localization for mobile robots in dynamic environments,” *Journal of Artificial Intelligence Research*, vol. 11, pp. 391–427, 1999.

[6] D. Schulz, W. Burgard, D. Fox, and A.B. Cremers, “Multiple object tracking with a mobile robot,” in *Computer Vision and Pattern Recognition (CVPR)*, Kauai, Hawaii, 2001, vol. 1, pp. 371–377.

[7] Ernst D. Dickmanns, “Vehicles capable of dynamic vision,” in *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, Nagoya, Japan, 1997, pp. 1577–1592.

[8] P. Stone, M. Asada, T. Balch, R. D’Andrea, M. Fujita, B. Hengst, G. Kraetzschmar, P. Lima, N. Lau, H. Lund, D. Polani, P. Scerri, S. Tadokoro, T. Weigel, and G. Wyeth, “Robocup-2000: The fourth robotic soccer world championships,” *AI Magazine*, pp. 495–508, 2000.

[9] M. Beetz, S. Buck, R. Hanek, T. Schmitt, and B. Radig, “The AGILO autonomous robot soccer team: Computational principles, experiences, and perspectives,” in *First International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS) 2002*, Bologna, Italy, 2001, pp. 805–812.

[10] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C*, Cambridge University Press, Cambridge, 1996.

[11] Andrew Blake and Michael Isard, *Active Contours*, Springer-Verlag, Berlin Heidelberg New York, 1998.

[12] O.D. Faugeras, “Three-dimensional computer vision: A geometric viewpoint,” *MIT Press*, p. 302, 1993.

[13] S. Julier and J. Uhlmann, “A new extension of the kalman filter to nonlinear systems,” The 11th Int. Symp. on Aerospace/Defence Sensing, Simulation and Controls., 1997.

[14] D. Reid, “An algorithm for tracking multiple targets,” IEEE Transactions on Automatic Control, 24(6):843–854, 1979.

[15] Y. Bar-Shalom and T. Fortmann, “Tracking and data association,” Academic Press., 1988.

[16] R. Hanek and T. Schmitt, “Vision-based localization and data fusion in a system of cooperating mobile robots,” in *International Conference on Intelligent Robots and Systems (IROS)*, Takamatsu, Japan, 2000, pp. 1199–1204.

[17] T. Schmitt, M. Beetz, R. Hanek, and S. Buck, "Watch their moves: Applying probabilistic multiple object tracking to autonomous robot soccer," in *AAAI National Conference on Artificial Intelligence*, Edmonton, Canada, 2002, pp. 599–604.

[18] T. Schmitt, R. Hanek, S. Buck, and M. Beetz, "Cooperative probabilistic state estimation for vision-based autonomous mobile robots," in *International Conference on Intelligent Robots and Systems (IROS)*, Maui, Hawaii, 2001, pp. 1630–1638.

[19] J.-S. Gutmann, W. Hatzack, I. Herrmann, B. Nebel, F. Rittinger, A. Topor, T. Weigel, and B. Welsch, "The CS Freiburg Robotic Soccer Team: Reliable Self-Localization, Multirobot Sensor Integration, and Basic Soccer Skills," in *Second International Workshop on RoboCup (Robot World Cup Soccer Games and Conferences)*. 1999, Lecture Notes in Computer Science, Springer-Verlag.

[20] R. Talluri and J.K. Aggarwal, "Mobile robot self-location using model-image feature correspondence," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 1, pp. 63–77, Feb. 1996.

[21] L. Iocchi and D. Nardi, "Self-Localization in the RoboCup Environment," in *Third International Workshop on RoboCup (Robot World Cup Soccer Games and Conferences)*. 1999, Lecture Notes in Computer Science, Springer-Verlag.

[22] P. Jonker, J. Caarls, and W. Bokhove, "Fast and Accurate Robot Vision for Vision based Motion," in *4th International Workshop on RoboCup (Robot World Cup Soccer Games and Conferences)*, P. Stone, T. Balch, and G. Kraetzschmar, Eds. 2000, Lecture Notes in Computer Science, pp. 72–82, Springer-Verlag.

[23] C. Marques and P. Lima, "Vision-Based Self-Localization for Soccer Robots," in *International Conference on Intelligent Robots and Systems (IROS)*, Takamatsu, Japan, 2000, pp. 1193–1198.

[24] G. Adorni, S. Cagnoni, S. Enderle, G.K. Kraetzschmar, M. Mordonini, M. Plagge, M. Ritter, S. Sablatnög, and A. Zell, "Vision-based localization for mobile robots," *Robotics and Autonomous Systems*, , no. 36, pp. 103–119, 2001.

[25] S. Enderle, M. Ritter, D. Fox, S. Sablatnög, G. Kraetzschmar, and G. Palm, "Soccer-robot localization using sporadic visual features," in *IAS-6 International Conference on Intelligent Autonomous Systems*, 2000.

[26] D. Fox, W. Burgard, H. Kruppa, and S. Thrun, "A probabilistic approach to collaborative multi-robot localization," *Autonomous Robots*, vol. 8, no. 3, pp. 325–344, 2000.

[27] S. Thrun, D. Fox, and W. Burgard, "Monte carlo localization with mixture proposal distribution," in *AAAI National Conference on Artificial Intelligence*, D. Kortenkamp, R.P. Bonasso, and R. Murphy, Eds., Austin, TX, 2000.

[28] M.A. Tanner, *Tools for Statistical Inference*, Springer, 1993.

[29] R. Hanek, N. Navab, and M. Appel, "Yet another method for pose estimation: A probabilistic approach using points, lines, and cylinders," in *Proc. of IEEE Conf. Computer Vision and Pattern Recognition*, 1999, pp. II:544–550.

[30] S.I. Roumeliotis and G.A Bekey, "Collective localization: A distributed Kalman filter approach to localization of groups of mobile robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2000, pp. 2958–2965.

[31] R. Kurazume and S. Hirose, "Study on cooperative positioning system: optimum moving strategies for CPS-III," in *IEEE International Conference on Robotics and Automation (ICRA)*, 1998, pp. 2896–2903.

[32] I.M. Rekleitis, G. Dudek, and E.E. Milios, "Multi-robot collaboration for robust exploration," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2000, pp. 3164–3169.

[33] M. Dietl, J.-S. Gutmann, and B. Nebel, "Cooperative sensing in dynamic environments," in *International Conference on Intelligent Robots and Systems (IROS)*, Maui, Hawaii, 2001, pp. 1706–1713.

[34] C. Hue, J.-P. Le Cadre, and P. Perez, "Tracking multiple objects with particle filtering," Tech. Rep. 1361, IRISA, 2000.

[35] I.J. Cox and S.L. Hingorani, "An Efficient Implementation of Reid's Multiple Hypothesis Tracking Algorithm and Its Evaluation for the Purpose of Visual Tracking," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 18, no. 2, pp. 138–150, Feb. 1996.

[36] I. Cox and J. Leonard, "Modeling a dynamic environment using a bayesian multiple hypothesis approach," Artificial Intelligence, 66:311–344, 1994.

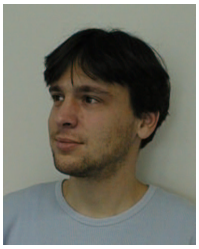
[37] P. Jensfelt and S. Kristensen, "Active global localisation for a mobile robot using multiple hypothesis tracking," in *Workshop on Reasoning with Uncertainty in Robot Navigation (IJCAI'99)*, Stockholm, 1999, pp. 13–22.

[38] D. Maksarov and H. Durrant-Whyte, "Mobile vehicle navigation in unknown environments: a multiple hypothesis approach," *IEE Proceedings: Control Theory & Applications*, vol. 142, no. 4, pp. 385–400, 1995.

Thorsten Schmitt is a Research Assistant in the Image Understanding Group of the Munich University of Technology and the team leader of the middle size RoboCup Team the *AGILO RoboCuppers* since 1998. He received a B.Sc. in Computer Science in 1995 from the University of Newcastle upon Tyne, United Kingdom, and a M.Sc. in Computer Science (Summa Cum Laude) in 1998 from the Munich University of Technology, Germany. Since 2002 he serves as a member of the Technical Committee of the middle size robot league. His research interests include computer vision-based pose estimation, object tracking and cooperation in autonomous robot systems.

Robert Hanek is a Ph.D. candidate at Munich University of Technology, Germany, where he has been a Research Assistant since November 1998. His main scientific interests include statistical model-based methods in robotics and computer vision, such as methods for image segmentation, tracking, pose estimation, and shape refinement. Mr. Hanek received the M.Sc. in Computer Science (Summa Cum Laude) from Erlangen University, Germany, in 1998. He did his Master Thesis on pose estimation at Siemens Corporate Research, Princeton, NJ. Mr. Hanek received the best paper award at the DAGM (German Computer Vision Symposium) in 2001 and together with the members of the *AGILO RoboCuppers* the RoboCup Engineering Challenge Award in 2002.

Michael Beetz is a Lecturer at Munich University of Technology's Department of Computer Science and head of the research group *Intelligent Autonomous Systems*. He is also the principal investigator for several nationally funded research projects, including plan-based control in distributed supply chain management, learning robot action plans, and autonomous robotic soccer. His research interests include AI, plan-based control of autonomous agents, and intelligent autonomous robotics. He received his M.Sc. in Computer Science (Summa Cum Laude) from the University of Kaiserslautern, his Ph.D. in Computer Science from Yale University (1996), and his venia legendi from the University of Bonn in 2001.



Sebastian Buck has been a Research Assistant at the Department of Computer Science of Munich University of Technology since November 1999. His research interests include neural control, coordination, and simulation of automotive systems in dynamic environments. In 1999 he received the M.Sc. in Computer Science from the University of Karlsruhe, Germany, where he has been working on biologically plausible neural models and multi agent learning. He has been a member of the RoboCup teams *Karlsruhe Brainstormers* and *AGILO RoboCuppers*.

Bernd Radig has been a full professor since 1986 at the Department of Computer Science of the Munich University of Technology. Since 1988 he has also been Chief Managing Director of the Bavarian Research Center for Knowledge Based Systems (FORWISS) and head of the FORWISS Image Understanding Group. Since 1993 he has been chairman of *abayfor*, a cooperation of about 20 research networks in Bavaria, conducting high level research in a broad range of fields. His own research interests include Artificial Intelligence, Computer Vision and Image Understanding, Signal Processing and Pattern Recognition. Prof. Radig received a degree in Physics from the University of Bonn in 1973 and a doctoral degree as well as a habilitation degree in Informatics from the University of Hamburg in 1978 and 1982, respectively. Until 1986 he was an Associate Professor at the Department of Computer Science at the University of Hamburg, leading the Research Group Cognitive Systems.