The AGILO Robot Soccer Team — Experience-based Learning and Probabilistic Reasoning in Autonomous Robot Control

Michael Beetz, Thorsten Schmitt, Robert Hanek, Sebastian Buck, Freek Stulp, Derik Schröter and Bernd Radig

***Abstract*— This article describes the computational model underlying the AGILO autonomous robot soccer team, its implementation, and our experiences with it. According to our model the control system of an autonomous soccer robot consists of a probabilistic game state estimator and a situated action selection module. The game state estimator computes the robot's belief state with respect to the current game situation using a simple off-the-shelf camera system. The estimated game state comprises the positions and dynamic states of the robot itself and its teammates as well as the positions of the ball and the opponent players. Employing sophisticated probabilistic reasoning techniques and exploiting the cooperation between team mates, the robot can estimate complex game states reliably and accurately despite incomplete and inaccurate state information. The action selection module selects actions according to specified selection criteria as well as learned experiences. Automatic learning techniques made it possible to develop fast and skillful routines for approaching the ball, assigning roles, and performing coordinated plays. The paper discusses the computational techniques based on experimental data from the 2001 robot soccer world championship.**

***Index Terms*— action selection and planning, coordination of perception, reasoning, and action, integration and coordination of multiple activities.**

I. Introduction

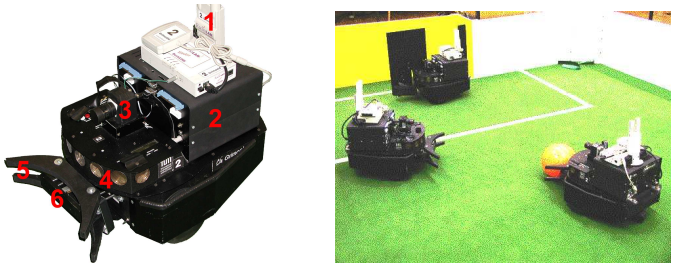
This article describes the computational model underlying the AGILO[1] autonomous robot soccer team [7], its implementation, and our experiences with it. Robotic soccer has become a standard "real-world" test-bed for autonomous multi robot control. In robot soccer (mid-size league) two teams of four autonomous robots – one goal keeper and three field players – play soccer against each other on a four by nine meter large soccer field. The key characteristic of mid-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, goals, and even entire game situations. The robots also need to collaborate by coordinating and synchronizing their actions to achieve their objectives – winning games.

The authors are with the Fakultät für Informatik, Lehrstuhl IX of the Technische Universität München, Boltzmannstrasse 3, D-85748 Garching bei München, Federal Republic of Germany, beetz@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/.

[1] This name is an homage to the Agilolfi nger, the earliest dynasty ruling in Bavaria during the 6th century. The dynasty's most famous representatives are Grimoald, Hugibert, Odilo, Tassilo and Theodo. Our soccer robots bear the same names.

Robot soccer provides a challenging and realistic testbed for cooperative state estimation in complex and dynamically changing environments. These challenges include: (1) a competitive, highly dynamic, and fast changing environment, (2) a changing number of opponent robots with unknown identity, (3) the use of an inaccurate and noisy vision sensors and (4) independently moving sensors with inaccurately known positions.

The AGILO RoboCup team is realized using inexpensive, off-the-shelf, easily extendible hardware components and a standard software environment. The team consists of four Pioneer I robots; one of which 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 fixed color CCD camera with a lens opening angle of 90^o (3) is mounted on the robot. The robot also has a guide rail (5) and a kicking device (6) that enable the robot to dribble and shoot the ball.



(a) (b)

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

Our hardware is somewhat Spartanic. Our off-the-shelf forward facing camera with 90^o field of view has a very restricted view as compared to the omni-directional vision system and laser range finders that most RoboCup mid-size teams use. We also use a differential drive instead of the much more dexterous holonomous drives. Whereas this hardware gives us a substantial disadvantage against teams which use one or more of the above-mentioned components, it also confronts us with challenging research problems.

In this article we investigate the design and realization of an integrated autonomous control system with the following emphases: The first emphasis is on situation assessment and decision making under uncertainty. The second emphasis is on mapping of abstract and possibly conflicting objectives into

appropriate low-level control signals. The third emphasis is on selecting appropriate actions with limited computational resources.

The first and perhaps most important reasoning task to be performed is the assessment of the game situation. Several limitations of the robots as well as environmental conditions make this reasoning task difficult to perform. 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, specularity, and shadows 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.

Thus, estimating the game state both accurately and reliably is very difficult because (1) the sensors have limited accuracy and uncertain positions (2) the soccer field is only partly accessible due to limited camera view and occlusion caused by other robots (3) the robots change their direction and speed very abruptly (4) the models of the dynamic states of the robots of the other team are very crude and uncertain (5) vast amounts of data must be processed in real-time. However, reasoning about uncertainty is not only necessary to infer the most likely game state but also to select the actions that are likely to satisfy the current objectives of the robot and its team. To select the actions properly, the robot must predict the consequences of its possible actions, which have nondeterministic effects, as well as the intentions of the opponents, which are even more difficult to predict.

Another challenge is the selection of the actions that are most appropriate to foster the objectives of the robot. Of course, the highest goal is to score more goals than the opponents. This can be achieved by either playing offensively and scoring many goals or defensively by preventing the other team from scoring. Unfortunately these two objectives are conflicting: to score more goals the team often has to play offensively and risky, and will therefore play with a weaker defense. In the course of the game the robot therefore has to make a trade-off between these conflicting objectives.

The second problem in action selection is the huge gap, in terms of abstraction, between the objectives of the robot, such as defending the goal, and the control signals that it sends, such as setting the voltage of a motor. Clearly, in order to reason about the appropriateness of actions effectively and efficiently, the robot must structure its activity into meaningful tasks, such as a dribbling, a shot, and a fight for ball possession. Many researchers consider the hierarchical organization of tasks as one of the key principles to act and acquire skills effectively.

The complexity and ill-structuredness of many control problems poses another problem. Reaching target positions and dynamic states fast, for example, requires complex and accurate driving maneuvers. As a consequence, simple heuristics for deciding which robot should get to the ball and how do not work well.

The last challenge that we explicitly address in our research is decision making with bounded computational resources. Because soccer games are very dynamic the robot should select its actions with a frequency of at least ten cycles per second. There is often a trade-off between reasoning more thoroughly and reasoning faster and more often. The best of both worlds can often be obtained by compiling acquired knowledge from past experiences into fast inference procedures through experience-based learning.

In this paper, we show how the AGILO autonomous robot soccer team meets these challenges. The AGILO robot controllers employ game state estimation and situated action selection. The game state estimator estimates the complete game situation at frame rate using a cheap off-the-shelf camera system. The game state includes the robot's own position, the position of teammates and opponent robots, as well as the ball. The use of such a vision system yields considerable inaccuracies in the sensor data and highly incomplete information about the game situation. The AGILO game state estimator deals with these problems by employing sophisticated probabilistic reasoning techniques and by exploiting the cooperation between the game state estimators of different robots. The second component of the individual robot controllers selects appropriate actions based on an assessment of the current estimated game state. The situated action selection mechanism is the default mechanism for choosing actions and must therefore propose reasonable actions for all possible situations. Because the robots are difficult to steer, the AGILO soccer robots employ experience-based learning mechanisms to acquire competence in robot motion control.

The remainder of this paper explains how these mechanisms have been implemented and embedded into the AGILO robot control software. We report on our experiences with the software and lay out our plans for the next generation of the system.

II. A Robot Soccer Episode

Figure 2 depicts six snapshots from an episode of a game between *The AGILO RoboCuppers* and *The Ulm Sparrows* at the 2001 RoboCup world championship in Seattle. Note that, although the walls are no longer present in current mid-size league RoboCup, neither our state estimation nor action selection depend on them. The first snapshot, figure 2(a), shows the ball lying at the surrounding wall and two AGILO robots facing the ball. To avoid conflicts, AGILO 1 goes for the ball and AGILO 2 is going into the back field to be the defensive back. This role assignment is favorable because upon reaching the ball AGILO 1 can head directly towards the opponent's goal whereas AGILO 2 would face the wrong direction. Therefore, AGILO 2 takes over the defensive position and moves into the back field. In snapshot 2(b) AGILO 1 is trying to shuffle the ball off the wall, which requires a sophisticated rotational movement. However, by the time the robot has achieved this goal a striker from the Ulm Sparrows blocks its path (figure 2(c)). Due to an obstacle avoiding maneuver AGILO 1 gets stuck at the wall.

Figures 2(d) and 2(e) depict how it is freeing its way. Even though AGILO 2 does not see the ball on its way back into the defensive position, it knows its exact position which is estimated by AGILO 1 and communicated via a wireless connection link to all teammates. Because of Ulm's striker blocking the way to the ball, the AGILO 2 robot estimates that it can now

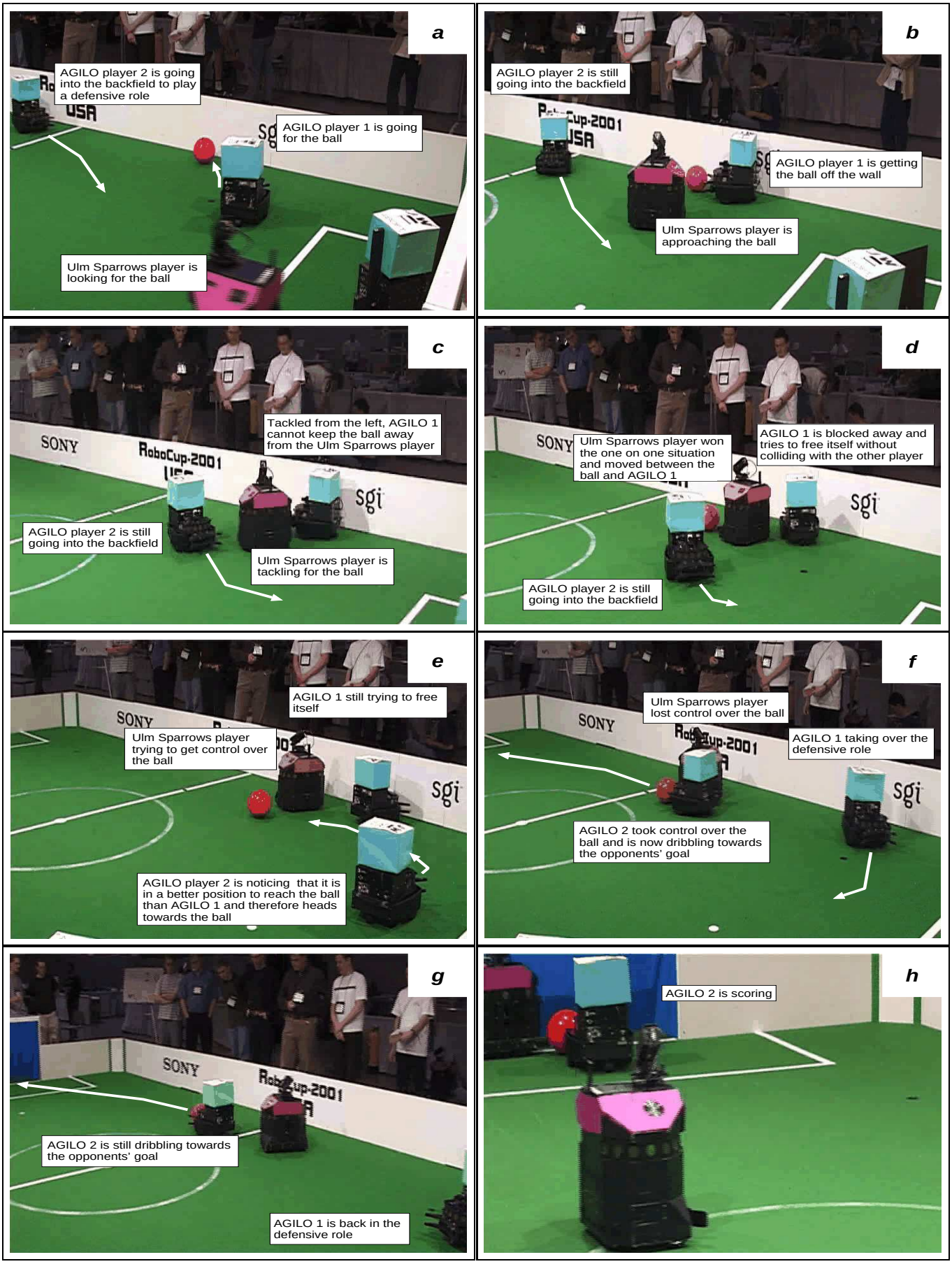


Fig. 2. Episode from an a match between *The AGILO RoboCuppers* and *The Ulm Sparrows*.

reach the ball faster than AGILO 1. Therefore, AGILO 2 turns and goes for the ball (figure 2(e)). As AGILO 1 also estimates that AGILO 2 can reach the ball faster, it takes the defensive position and backs up AGILO 2.

Finally, AGILO 2 dribbles the ball towards the opponent goal and decides whether it should try to score by passing the goal keeper on the left or the right side of the goalkeeper (see Figures 2(f) to 2(h)). In order to maximize the chance of scoring the AGILO 2 is aiming for the right goal post, which allows AGILO 2 to kick the ball into both corners. If the goalkeeper drives towards the right corner a short turn and a kick into the left corner is sufficient to score. On the other hand if the goal keeper decides to stay in the middle of the goal or moves to the left, then a shot into the right corner is adequate. Eventually the goalkeeper stays in the middle of the goal and the striker scores into the right corner. The match ends with a 7:0 win of *The AGILO RoboCuppers* (see also Section VI).

This episode demonstrates several strengths of the AGILO robot control software. The first one is the competent and accurate assessment of game situations. This assessment comprises accurate position estimates for the teammates, the ball, and the opponent robots. It is important to note that because the robots of the AGILO team cooperate in game state estimation and exchange their observations each one of them has a fairly complete picture of the game situation. Consider for example the player AGILO 2 that accurately knows where the ball is even though the ball is behind itself. What is also notable is that the robots plan collision free paths and follow them, which is only possible because the robots keep track of the positions of their opponents. Finally, when dribbling towards the goal the robot simultaneously keeps track of the goal, the goal keeper, and the ball in order to react to the defensive moves of the keeper.

The second class of strengths are concerned with the skills of the robots and the way they adapt and play their roles. The players of the AGILO team reassign roles to themselves based on an assessment of the game situation. In particular, they continually monitor whether they can reach the ball faster than any of their teammates. If so, they immediately, go for the ball. Because of this task assignment strategy AGILO 2 is capable of taking AGILO 1's role as AGILO 1 gets stuck when being blocked by the Ulm Sparrows player. Besides their role assignment, the skills in approaching the ball, dribbling, and moving in dynamic situations are remarkable. We will show in section V how these can be learned.

III. OVERVIEW OF THE SYSTEM MODEL

As our basic conceptualization of the robot control system, the robot it controls, and the environment it acts in, we use the *dynamic system model* [14], [37]. In the dynamic system model, the state of the world evolves through the interaction of two processes: one denoted the controlling process — the robot's control system (or *controller*) — and the other the controlled process, which comprises events in the environment, physical movements of the robot and sensing operations. The purpose of the controlling process is to monitor and steer the evolution of the controlled process so that it satisfies the given constraints and meets the specified objectives.

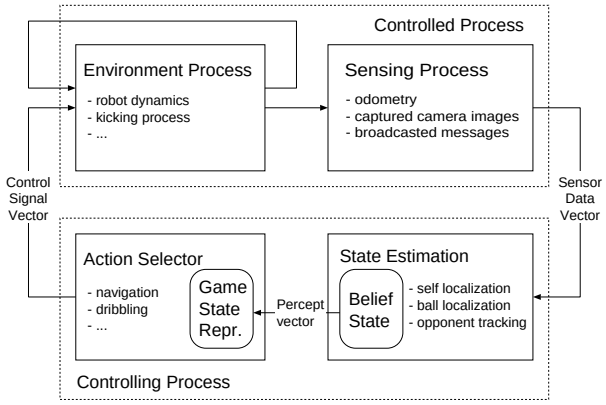


Fig. 3. Block diagram of our dynamic system model for autonomous robot control. Processes are depicted as boxes and interactions as lines with arrows.

Figure 3 shows a block diagram of the dynamic system model that underlies the design of our robot control system. The processes are depicted as boxes and the interactions between them as lines with arrows. There are two interactions between the controlling and the controlled process. The first interaction is initiated by the controlling process, which sends control signals to the controlled process in order to steer its evolution. The second interaction consists of the controlling process observing the controlled process by interpreting the sensor data that are produced by the controlled process.

For the control of soccer robots, it is useful to conceptually decompose the controlled process into an environment and a sensing process [14]. The environment process comprises the robot's physical behavior: different kinds of possibly concurrent movements that cause changes of the robot's state. At the same time exogenous events, such as an opponent robot trying to reach the ball, occur and change the state of the controlled process, too.

The sensing process maps the world state into the sensor data, the input of the controlling process. Sensors can provide continual streams as well as individual bits of sensor data. The sensor data received by the robots contain odometric data, captured camera images, and messages broadcasted by teammates. Odometric readings tell the robot how often the wheels turned since the last reading and therefore provide information about how much the robot moved in the meanwhile. Unfortunately, the information provided by odometric readings might be corrupted, for example due to slippage of the wheels. Broadcasted messages of the teammates contain the teammates' own estimates of their own positions and the positions of the ball and the opponent players.

It is important to note that the robot's sensors can, in general, access only partial information about the state of the environment. The sensor data measured might also be inaccurate and must therefore often be interpreted to be useful for controlling the robot. The inaccuracy, locality of sensor data as well as the data interpretation processes yield data that are incomplete, inaccurate, and ambiguous.

The decomposition of the controlled process suggests an analogous decomposition of the controlling process into a state estimation and an action selection process. The incoming video

streams are processed by the *vision-based cooperative state estimation module*, which computes the *belief state* of the robot with respect to the current game situation. The belief state contains estimates of the positions of the robot itself, its teammates, the ball, and the opponent players. 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 processes compute the robot's beliefs about the state of the controlled system. The results of the state estimation are stored in the robot's belief state, which contains information such as the robot's estimated position, and the accuracy and a measure of the ambiguity of the position estimate.

The action selector continually receives the percept vector generated by the state estimation processes and generates control signals for the controlled process. The action selection module then computes an abstract feature description of the estimated game state that can be used to recognize relevant game situations. Based on the abstract game state description, the situated action selection module selects actions based on a limited horizon utility assessment.

IV. Vision-based, Cooperative Game State Estimation

The game state estimators [45] of the AGILO robots provide their action selection routines with estimates of the positions and the dynamic states of each player and the ball. The AGILO robots employ *probabilistic* state estimation to assess the game state. Probabilistic state estimators maintain probability densities for the states of objects over time conditioned on the sensor measurements received so far. 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. As a result, a probabilistic robot can gracefully recover from errors, handle ambiguities, and integrate sensor data in a consistent way. Moreover, a robot employing probabilistic reasoning techniques knows about its own ignorance, a key prerequisite of truly autonomous robots.

Approached probabilistically, the state estimation problem can be considered as a density estimation problem, where a robot seeks to estimate a posterior distribution over the space of its poses and the poses of other objects conditioned on the available data. Denoting the game state at time t by s_t, and the data leading up to time t by $d_t, \ldots, d_0$, we write the posterior as $p(s_t|d_t, ..., d_0; m)$. Here m is the model of the world (e.g., a map). We will also refer to this posterior as $Bel_t(s_t)$, the robot's belief state at time t [2]. The game state consists of the compound state variables $Robot^1$, ..., $Robot^4$, $Ball$, $Opponent^1$, ..., $Opponent^n$. The number of opponents varies and robots might be taken out of the field and might reenter the filed. The compound state variable $Robot^i = \langle x^i, y^i, \theta^i, \dot{x}^i, \dot{y}^i, \dot{\theta}^i \rangle$ comprises the position (x^i, y^i) and orientation θ^i of robot i and its translational $(\dot{x}^i, \dot{y}^i)$ and rotational velocity $\dot{\theta}^i$. $Robot^i_t$ refers to the value of these variables at time step t. Analogously, $Ball = \langle x^{ball}, y^{ball}, \dot{x}^{ball}, \dot{y}^{ball} \rangle$ denotes the position and velocity of the ball, where the ball velocity is interpolated from the last ball position estimates. Finally, $Opponent^j = \langle x^j, y^j, \dot{x}^j, \dot{y}^j \rangle$ where $(\dot{x}^j, \dot{y}^j)$ is again interpolated from previous estimates. The AGILO game state estimators do not observe the orientations of opponent robots. Determining the pose of components would require sophisticated 3D models of the opponent robots.

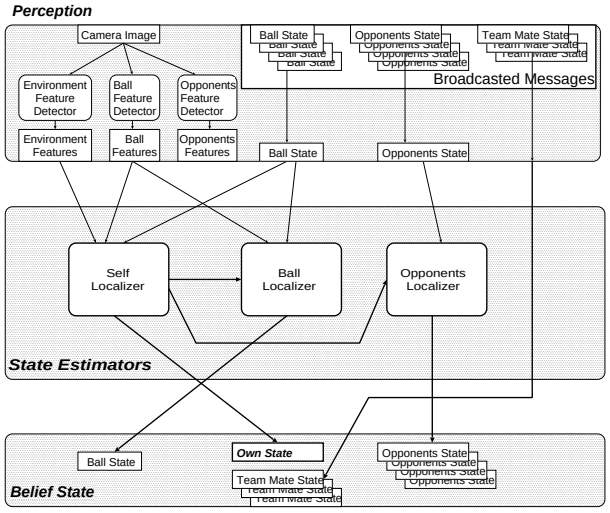


Fig. 4. Software architecture of the state estimator.

A key problem in solving this state estimation problem is the complexity of the joint probability density and the huge amount of data that the probability density is conditioned on. This requires us to factorize, approximate, simplify (by making assumptions), and decompose the probability density [48]. The AGILO game state estimators decompose the estimation problem into subproblems for self-localization and for tracking different kinds of independently moving objects: the ball and the opponent players. 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. We will detail our mechanisms for data compression and the approximations and simplifications that we make in the remainder of this section.

Figure 4 shows the components of the state estimator and its embedding into the control system. The subsystem consists of the perception subsystem, the state estimator itself, and the belief state. The perception 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 perception subsystem provides the following kinds of information: (1) partial state estimates that are 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 all opponents in its field of view. 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.

As stated above, the state estimation subsystem consists of three interacting estimators: (1) the self localization system, (2) the ball estimator, and (3) 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 teammates.

Every robot maintains its own belief state with respect to the game state, which is constructed as follows. The own position, the position of the ball, and the positions of the opponent players are computed by the local state estimation processes. The estimated positions of the teammates are the broadcasted results of the self localization processes of the corresponding teammates because a robot can localize itself with much higher accuracy than other robots.

The state estimation module is realized as a collection of different variants of Bayes filters. These filters are iterative update rules for probability densities given new evidences in the form of sensor data and robot actions. The filter algorithms drastically reduce the computational complexity of the state estimation problem by exploiting the Markov assumption $p(x|z_{t...0}) = p(x|z_t)$ and transforming the density estimation problem of $p(x|z)$ through the application of Bayes rule to a computationally and implementationally more favorable form using the observation model $p(z|x)$. Based on these assumptions, the complexities of the observation model and the motion model are substantially reduced.

Figure 5 lists the generic Bayes filtering algorithm. With every iteration of the algorithm, the belief state $Bel(x_t)$ is updated according to new evidence including a broadcasted message of a team mate, a newly captured image or an odometric reading. Here, x_t refers to the state of one of the compound state variables $Robot^1$, ..., $Robot^4$, $Ball$, $Opponent^1$, ..., $Opponent^n$.

Depending on the type of data (sensor data or control signal), the algorithm can be divided into two different stages: (1) prediction, and (2) update. During the *prediction stage*, a model of the environment process and the control signal u_{t-1} are used to obtain the prior probability density function of the state at time

```
algorithm BAYES FILTER (Bel(x), data)
 1  let
 2    Bel(x)      % previous belief state
 3    Bel'(x)     % updated belief state
 4    data        % data item (action or signal)
 5    ν           % normalising constant
 6
 7  do
 8    ν ← 0;
 9    switch (data)
10
11      case (data is a control signal vector u) :
12          % prediction stage
13          for each x do
14            Bel'(x) ← ∫ p(x|x',u) Bel(x) d x';
15
16      case (data is a sensor data vector z) :
17          % update or measurement stage
18          for each x do
19            Bel'(x) ← p(z|x) Bel(x);
20            ν ← ν + Bel'(x);
21          for each x do
22            Bel'(x) ← ν^-1 Bel'(x);
23
24  return (Bel'(x));
```

Fig. 5. The Bayes Filtering Algorithm.

t, via the Chapman-Kolmogorov equation:

$$Bel(x_t) = \int p(x_t|x_{t-1}, u_{t-1})\ Bel(x_{t-1})\ d\ x_{t-1} \qquad (1)$$

During the *update stage*, an observation z_t becomes available and is used to update the prior density to obtain the required posterior density of the current state:

$$Bel(x_t) = \nu\ p(z_t|x_t)\ Bel(x_{t-1}) \qquad (2)$$

where the likelihood function $p(z_t|x_t)$ is defined by the observation model.

To implement the Bayes filter, we must provide three probability distributions: (1) the initial belief $Bel(x_0)$, (2) the next state probability $p(x_t|x_{t-1}, u_{t-1})$ given the current state and the last action, (3) and the observation likelihood $p(z_t|x_0)$ for a given state.

In the AGILO robot system, these densities are approximated by Gaussian distributions. Gaussians have the advantage that they represent a state and the associated uncertainty in a compact and uniform way with a mean vector and a covariance matrix, respectively. This distribution can efficiently be communicated over channels with narrow bandwidth (such as wireless LAN) and processed by the state estimation algorithms. To account for the lack of reliability of Gaussian Belief filters we run a more robust and computationally more expensive state estimator as a monitor in parallel.

A. Perception

The information needed for game state estimation is provided by the perception system and includes 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 teammates comprise the respective robot's location, the ball's location, and the locations of the opponents. 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.

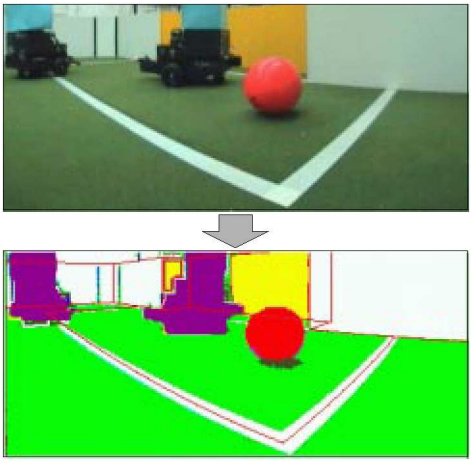


Fig. 6. The figure shows an image captured by the robot and the feature map that is computed for self, ball, and opponent localization.

The workhorse of the perception component is a color classification and segmentation algorithm that is used to segment a captured image into colored regions and blobs (see figure 6). The color classifier is learned in a training session before tournaments in order to adapt the vision system to specific lighting conditions and effects.

B. Self and Ball Localization

The self- and ball-localization module [23] iteratively estimates, based on a model of the environment, the probability density over the possible robot positions, given the observations taken by the robot.

A robot's belief $Bel(x_t)$ about its own position is approximated by a multi-variate Gaussian density and represented by its mean vector $\bar{\mathbf{x}} = (x, y, \phi, x_{ball}, y_{ball})^T$, and 5×5 covariance matrix $\Sigma_{\mathbf{x}}$.

$$\begin{aligned} Bel(x_t) &= N(x_t; \bar{\mathbf{x}}, \Sigma_{\mathbf{x}}) \\ &= \frac{1}{\sqrt{(2\pi)^5 det(\Sigma_{\mathbf{x}})}} e^{-\frac{1}{2}\{(x_t - \bar{\mathbf{x}})^T \Sigma_{\mathbf{x}}^{-1} (x_t - \bar{\mathbf{x}})\}} \end{aligned} \quad (3)$$

The robot's model of the static part of its environment that is used for self localization is composed of landmarks together with their positions and orientations. The landmarks include goals, field lines, and walls surrounding the pitch. Each entity is modeled as a curve feature [10]. Figure 7 depicts an excerpt of the environment model representing the neighborhood around a goal. The goal is modeled as a set of 3D lines where each line is associated with a color transition. Using this environment model and a position estimate, the robot can predict where in a captured image lines should be visible and which color transition they represent.

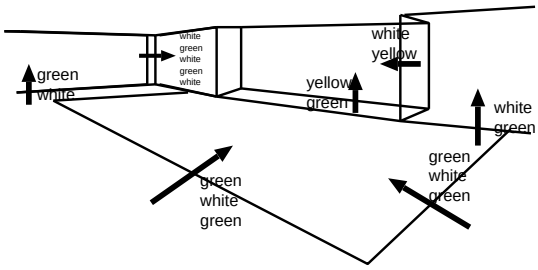


Fig. 7. Model of the neighborhood of a goal. The model contains the edges of the objects and the color transition they are the borderline of.

The self-localization algorithm performs a maximum a posteriori (MAP) estimate of the robot's pose. The MAP estimate $\widehat{x}_t$ is given by $\widehat{x}_t = \text{argmax}_{x_t}\ p(x_t) \cdot p(z|x_t)$, where $p(x_t)$ is the prior of the pose x_t 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. The second term $p(z|x_t)$ is the likelihood that the robot observed z given its current pose x_t.

The likelihood function $p(z|x_t)$ uses correspondences between the predicted and obtained data and determines the respective quality of fit. In order to approximate this function for given image data, the following steps are necessary: First, the 3D curve features that are predicted to be visible are projected into the image plane. Second, a local search is performed to establish correspondences between the model curve and image features. During this search knowledge about the expected color transitions is used. Measurement errors of the resulting observations are assumed to be mutually independent Gaussian random variables with zero mean. Based on these assumptions, we derive a likelihood function that evaluates the deviations between the found image points and the corresponding projections of the 3D curve features.

Finally, the resulting maximum a posteriori criterion is optimized for the robot pose. A maximum a-posteriori (MAP) estimation step computes an estimate $\widehat{x}_t$ of the robot pose which best fits to the position prediction $p(x_t)$ and to the image observations. Since the projection of the 3D features is non-linear, the resulting objective function can not be optimized in a closed-form. Hence, we use an iterative optimization based on Newton's method. The assumption of Gaussian errors together with the iterative optimization method make this method equivalent to an iterated Kalman filter. A nice feature of this algorithm is that a robot can use ball observations performed by teammate robots as dynamic landmarks and thus can solve under determined localization problems when the ball is in its field of view. The details of the algorithm can be found in [23], [45].

The above self-localization algorithm is used to track the pose of an AGILO player and the ball. It is initialized with a pose estimate generated by a global self-localization procedure [36], which is based on a particle filter. The self-localization

algorithm then refines this pose and tracks the robot's pose and the ball until it fails and requires reinitialisation. By providing two self-localization procedures computational resources are saved. The fast and accurate self-localization algorithm runs at frame rate (30 Hz), while the particle filter runs at 10 Hz or less.

C. Ball and Opponent Tracking

The ball and opponent tracker [44] is based on a multiple hypothesis tracking approach [39] and computes the belief state with respect to the existence and position of opponent robots. This part of the belief state is computed by (1) detecting feature blobs in the captured image which might correspond to an opponent, (2) estimating the world coordinates and uncertainties of these blobs, and (3) associating them with the object hypotheses, which correspond to tracked opponents.

The belief $Bel(x_t)$ of a opponent tracker is represented by a set of weighted Hypotheses, $H_t = \bigcup_{i=1}^{n}\{h_t^i, p_t^i\} = \bigcup_{i=1}^{n}\{< \bar{\mathbf{h}}_t^i, \Sigma_{\mathbf{h}_t^i} >, p_t^i\}$. The h_t^i's are Gaussian random variables, $h_t^i \sim N(\bar{\mathbf{h}}_t^i, \Sigma_{\mathbf{h}_t^i})$, representing an object hypothesis (a possible object state) by a mean $\bar{\mathbf{h}}_t^i = (x, \dot{x}, y, \dot{y})^T$ consisting of a 2D position and velocity estimate, an associated 4×4 covariance matrix $\Sigma_{\mathbf{h}_t^i}$, and non-negative numerical factors p_t^i called the *importance factors*. The importance factors determine the weight (=importance) of a hypothesis and can be thought of as the probability of representing a real opponent on the field. This representation is also sometimes called a sum or mixture of Gaussians and can be evaluated as follows:

$$Bel(x_t) = \sum_{i=1}^{n} p_t^i * N(x_t; \bar{\mathbf{h}}_t^i, \Sigma_{\mathbf{h}_t^i}) \tag{4}$$

The opponent tracker maintains a Kalman filter, for every hypothesis. In order to update the belief state five things need to be done for the set of hypotheses: (1) predict the positions according to the system model, (2) associate predicted hypotheses and observations, (3) fuse feasible associations and generate new hypotheses, (4) compute the hypotheses weights, and (5) prune the set of hypotheses.

The implementation of the prediction stage is straightforward. The only difference is that the system model has to be applied to every hypothesis. The update stage has to be extended such that it can handle multiple measurements, multiple hypotheses (add new, update existing and delete unlikely hypotheses), and can perform the probability computations for all hypotheses.

For the following, it is assumed that the measurement vector consists of a number of possible opponent observations, which were determined by the sensor data processing and feature extraction algorithms [45]. The first task performed by the update stage is to copy all predicted hypotheses to the set of new hypotheses. This accounts for the fact that none of the hypotheses might be reconfirmed by a measurement and is also referred to as track splitting.

Then, the update stage assigns the new observations to existing hypothesis. This process is called data association. An association is usually performed on the basis of a validation gate. Typically, for Gaussian probability densities the Mahalanobis distance is used. If an observation falls within the validation gate of an existing hypothesis, then they are assumed to have originated from the same physical object. In this case, the measurement and the hypothesis are fused by the Kalman filter update equations and are used to create a new hypothesis.

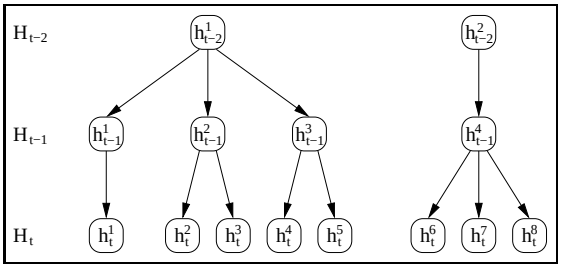


Fig. 8. Hypotheses trees maintained by the opponent tracker. Every branch of a tree represents one possible track of a physical object. New branches are introduced as an attempt to solve data association ambiguities through track splitting.

The ambiguities arising in this step can in the general case not be resolved. For example, it is possible that one opponent observation reconfirms several hypotheses or that an existing hypothesis is reconfirmed by more than one observation. A way to overcome this problem is to avoid unique associations and consider all possible assignments instead. This procedure generates a new hypothesis for every possible combination of a hypothesis from the previous time step and an observation from the current time step and stores them in a tree like data structure (see Figure 8). Every newly created hypothesis is added to the set of current hypotheses and is linked with its predecessor contained in the previous set of hypotheses. Over time this creates a tree like structure, where each branch represents a possible track of a dynamic object. This procedure delays the final unique associations for a predefined number of time steps in the hope that further information acquired in subsequent time steps will automatically resolve all existing ambiguities.

The computation of the importance factor (probability) of an hypothesis requires the evaluation of the expression $P(h_t^i|Z_{t\ldots0})$, where h_t^i is the hypothesis for which the probability is computed, and $Z_{t\ldots0}$ is the set of all current and previous observations. The form of this expression is highly task dependent [3], [13], [29], [45].

Observations that cannot be assigned to an existing hypothesis, are used to initialize new hypotheses. The probabilities of these hypotheses are initialized with a predefined constant probability. Alternative approaches are to derive an initial probability from the measurement's covariance matrix.

Finally, to constrain the growth of the set of hypotheses and the computational demand of the opponent tracker, the set of hypotheses is pruned. Several different and efficient pruning strategies exist and are applied. Similar hypotheses are merged, unlikely ones are discarded, and an upper bound on the number of hypotheses allows to save computational resources. Typically pruning strategies and their parameters exploit application specific information and heuristics. For alternative pruning strategies see, for example [3], [13], [29].

V. Situated Action Selection and Execution

After having described the mechanisms for game state estimation we will now turn to the issue of action selection. Throughout the game the AGILO robots have a fixed set of tasks with different priorities. The tasks are *shoot the ball into the goal, dribble the ball towards the goal, look for the ball, block the way to the goal, get the ball, ...* The situated action selection module enables the robots to select a task and to carry out the task such that, in conjunction with the actions of the teammates, it will advance the team's objectives the most. We consider a task to be the intention of the AGILO robot team to perform certain actions. Action selection and execution is constrained by (1) tasks being achievable only if certain conditions hold (e.g., the robot has the ball) and (2) a robot being able to only execute one action at a time.

We define that a task assignment a_1 is better than a_2 if there exists a task in a_2 that has lower priority than all the ones in a_1 or if they achieve the same tasks but there exists a task t in a_1 such that all tasks with higher priority are performed at least as fast as in a_2 and t is achieved faster by a_1 than by a_2. This performance criterion implies that if an AGILO robot can shoot a goal it always will try because this is the task with the highest priority. Also, if the AGILO team can get to the ball it tries to get there with the robot that can reach the ball the fastest. This strategy might not yield optimal assignments but guarantees that the highest priority tasks are achieved as quickly as possible.

To achieve a high degree of autonomy the AGILO robots perform the task assignment distributedly on the individual robots. This makes the task assignment more robust against problems in inter robot communication. These problems can be caused by robots being sent off the field, computers being crashed after heavy collisions, and communication being corrupted due to interferences with other communication channels.

The most salient features of the situated action selection are the following ones. First, to realize a competent and fast task assignment and execution mechanism the AGILO controllers make ample use of automatic learning mechanisms. Second, the task assignment mechanism works distributedly on the individual robots and are robust against communication corruptions. Finally, the task assignment and execution mechanism always produces purposeful behavior and always aims at the achievement of high priority tasks.

A. AGILO Simulator: a Tool for Learning

An important means for developing competent robot soccer skills is a robot simulator that allows for realistic, controllable, and repeatable experiments. For this reason we have developed a robot simulator that accurately simulates how the dynamic state of the robot changes as the robot's control system issues new driving commands such as setting the target translational and rotational velocities. The AGILO software development environment therefore provides a robot simulator that uses multi layer neural networks to simulate the dynamics of the AGILO soccer robots. We have used the RPROP algorithm [41] for supervised learning in order to teach the neural net mapping dynamic states and control signals into the subsequent states.

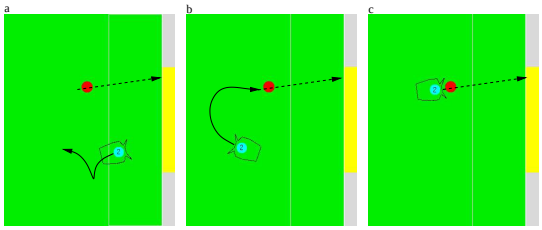


Fig. 9. A training scenario in the multi robot simulation environment: To acquire training patterns for the neural projector a robot is set to a randomly defined initial state x_i (position of the robot in subfigure a) and has to drive to a randomly defined target state x_t indicated by the dashed arrow. The direction and length of this arrow indicate the target state's orientation and velocity. The time the robot needs to reach its target state (subfigure b & c) is taken to complete the training pattern $\langle\langle x_i, x_t\rangle, time\rangle$.

For specializing the simulator to the AGILO robots we have performed a training session in which we have collected a total of more than 10000 training patterns from runs with real AGILO robots for a large variety of navigation tasks. Using a test set of patterns that was not contained in the training patterns we determined that prediction for the patterns for moderately difficult navigation tasks was about 99%. The accuracy decreased to about 92% in situations where both velocities, the translational and rotational one, were changed abruptly at the same time. These inaccuracies are caused by the lack of representative training patterns as well as the high variance in navigation behavior with maximal acceleration.

B. Task Assignment

A very simple algorithm suffices to compute task assignments that satisfy the performance criterion that we have stated before:

algorithm ASSIGN-TASKS(robs,tasks)
  for I ← 1 *to* LENGTH(tasks)
    ACTION($argmin_{r \in AGILO} cost(r, tasks[I]) \leftarrow tasks[I]$)

The algorithm works as follows. In the beginning of each iteration the action of each AGILO robot is reset to *idle*. Then the algorithm iterates over all tasks in the order of their priority. It then assigns the task to the idle AGILO robot that can achieve the task the fastest. The task assignment algorithm does two things: first, it computes the task that should be achieved by the robot itself and second, it computes which higher priority tasks will probably be achieved by which other robot.

The algorithm assumes knowledge of the cost of task achievement. For robot soccer we define the cost $cost(r_i, a_j)$ of a robot r_i performing an action a_j as the time needed to complete an action a_j, that is, the time to reach a given target state. To make accurate predictions a robot has to take its dynamic behavior, the intentions of its teammates, and possible opponent movements into account.

The AGILO task cost estimator performs three steps. First, the selection of the multi robot navigation method that matches the game state best. By taking the estimated game state into account the cost estimator can take even expectations about the movements of the opponents into account. Second, computing

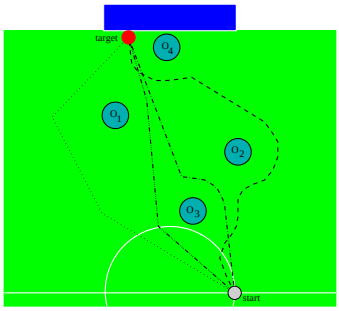


Fig. 10. Navigation plans for a given single robot navigation task as proposed by different navigation planning methods.

a path in the context of the navigation paths of the teammates. This is done by computing navigation paths that avoid negative interferences with the paths computed for the higher priority tasks (see section V-C). Third, the proposed path is then decomposed into a sequence of simpler navigation tasks for which the time cost can be accurately predicted using a neural network. The mapping from navigation tasks, given by start points and destinations, into the time cost needed for the task completion is realized through a multi layer artificial neural network and learned through the backpropagation derivative RPROP [41]. We have trained the network using about 300.000 training patterns generated from accomplishing random navigation tasks in the learned simulator (see figure 9).

C. Multi Robot Navigation Planning

Each robot employs a multi robot navigation planner in order to plan its own path in the context of the intentions of the teammates. The planner is given a joint navigation task that specifies a target state (position, orientation and velocity) for each robot of the team. The objective of the navigation system is to achieve a state where each robot is at its target state as fast as possible.

But which navigation planning method should a robot apply to achieve its objectives? Contemplate figure 10. The figure depicts a single robot navigation task in a typical game situation and the navigation plans proposed by different navigation planning algorithms with different parameterizations. The figure illustrates that the paths computed by the different methods are qualitatively very different. While one path is longer and keeps larger distances to the closest obstacles another one is shorter but requires more abrupt directional changes. The performance that the paths accomplish depends on many factors that the planning algorithms have not taken into account. These factors include whether the robot is holonomic or not, the dynamic properties of the robot, the characteristics of change in the environment, and so on. As a consequence, it seems impossible to analytically predict which navigation algorithm and parameterization works best for our application.

Rather than designing yet another multi robot navigation algorithm we have decided to equip the AGILO robots with a hybrid robot navigation planning system. The system [11] employs different single robot navigation and plan merging mechanisms and selects the appropriate methods based on an assessment of the given navigation task and game situation. This way the system can exploit the different properties of the individual methods by learning for which navigation tasks the methods are best suited.

The planning methods employed by the AGILO navigation system include the *Potential Field Method*, the *Shortest Path Method*, *Circumnavigating Obstacles*, and *Maximizing the Clearance*, all described in [32]. Plan merging and repair methods include methods for merging plans that add waiting steps in order to avoid interferences. Path replanning methods revise the individual plans such that no negative interferences will occur including the *Definition of Temporary Targets*, the *Hallucination of Obstacles at Critical Sections*, and the *Insertion of New Obstacles*. The first one modifies the path by introducing additional intermediate target points. The second one hallucinates additional obstacles at the positions where collisions might occur. The third one simply considers the other robot at its respective position as a static obstacle.

The predictive model of the expected performance of different navigation planning methods is specified by rules such as the following one:

if there is one intersection of the navigation problems
$\wedge$ the navigation problems cover a small area ($\leq 10.7m^2$)
$\wedge$ the target points are close to each others ($\leq 1.1m$)
$\wedge$ the starting/target point distances are small ($\leq 5m$)
then fastest-method($\langle$*potential field,temp. targets* $\rangle$)

This rule essentially says that the potential field method is appropriate if there is only one intersection and the joint navigation problem covers at most one fourth of the field, and the target points are close to each others. This is because the potential field algorithm tends to generate smooth paths even for cluttered neighborhoods.

We have learned a set of 10 rules including the one above using the C4.5 decision tree learning algorithm [38] with standard parameterization and subsequent rule extraction. To do so we have collected a training set of 1000 data records, where each data record contained a description of a randomly generated navigation task and the time resources required to complete the task for each possible combination of navigation planning and plan repair method.

The language for characterizing navigation tasks uses 7 features (see figure 11): (1) the number of intersections between the line segments that represent the navigation tasks, (2) the size of the bounding box of the navigation tasks, (3) the minimal linear distance between different starting positions, (4) the minimal linear distance between different target positions, (5) the minimal distance between the line segments that represent the navigation tasks, (6) the maximum length of the linear distances of the individual navigation tasks, and (7) the number of obstacles in the bounding box of the joint navigation task.

To sum up, the AGILO multi robot navigation algorithm works as follows. First, the appropriate planning mechanism is selected based on the assessment of the given navigation task and the situation in which it is to be executed. In the second step, the joint navigation task is decomposed into single robot navigation problems. The individual problems are then solved using the selected planning methods. Then, the individual plans are repaired in order to avoid negative interferences with the higher priority plans. Finally, the algorithm extracts sequences

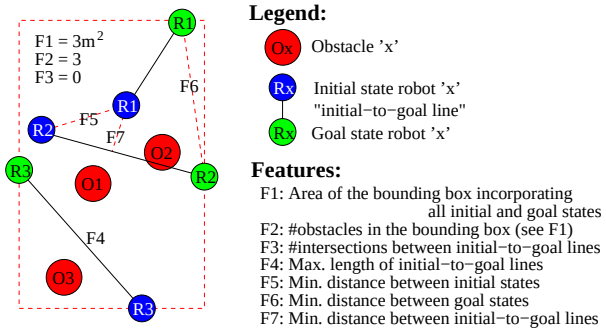


Fig. 11. Visualization of navigation task features that are used for classifying navigation tasks.

of target states from the robot's own navigation plan and sends those sequences to the robot's neural network controller, which is further described in the next section.

D. Execution of Navigation Plans

The *robot motion controllers* are used for achieving given dynamic states as fast as possible. The motion controller receives the target state (for example, the next state on a planned path) of a robot and returns low level commands that transform the current state into the target state as fast as possible. To arrive at the target state different trajectories are possible. But how to set them to quickly reach the target state? The AGILO robot controllers learn a direct mapping from the robot's current state and the robot's target state to the next command to be executed using multi layer artificial neural networks and the RPROP [41] algorithm: *Net*.

VI. Experiments and Discussion

The AGILO RoboCuppers, described in this paper, have participated in the fifth robot soccer world championship in Seattle (2001). The team has played six games for a total of about 120 minutes. The team advanced to the quarter finals playing games against Sharif CE (7:0), SPQR (7:0), Eigen (0:4), Ulm Sparrows (7:0), GMD Robots (1:1), and COPS Stuttgart (0:1). In the tournament none of the AGILO players was sent off the field because of causing collisions with opponent players or not leaving the penalty area in time. Most of the occasions in which the AGILO players had to be taken off to be restarted seemed to be caused by hardware problems. More results related to RoboCup 2001 can be found in [45].

For this paper we have developed a ceiling camera system, which provides ground truth information for any given match situation. For the following experiments our robots have played three matches under tournament conditions together with *The Ulm Sparrows* with a net playing time of about 2 hours. During these experiments belief states of the AGILO robots were recorded in a log file and compared after the match to the data provided by the ground truth camera system.

A. Game State Estimation

A typical result of the AGILO game state estimator is shown in Figure 12. Subfigure 12(a) shows the trajectories of the AGILO players, computed through vision-based self localization [23]. Subfigures 12(b) and (c) display the ball and opponent observations performed by the AGILO players, respectively. The tokens indicate which AGILO robot made the observations. Figure 12(d) visualizes how the individual observations of the ball and the opponent robots are merged into consistent tracks.

Qualitatively, we can estimate the accuracy of the game state estimation by looking for the jumps in the tracked lines. We can see that the own tracks are smooth and can therefore be expected to be accurate. Only defender Grimoald looses track of its pose and is reinitialized by its particle filter. The tracks of the ball and the opponents look very reasonable. They are less accurate and sometimes incomplete. We can also see that several tracks resulted from cooperative perception, i.e. from merging the observations of different robots. In addition, the exchange of observations results in fewer hallucinated obstacles and therefore allows for more efficient navigation paths. Several incorrect opponent observations made by the goal keeper (Theodo) and ball observations made by defender (Grimoald) were correctly omitted by the ball and opponent tracker and not assigned to a track.

Quantitative data for the experiments can be found in Tables I and II. Table I summarizes the localization accuracies for the AGILO robots and the observation accuracies for ball and opponent observations for the three friendly matches. The localization worked very well for the goalkeeper (#1) and the striker Odilo (#4). Their mean localization accuracies are estimated to be 12 and 19 cm, respectively. This is not amazing for the goalkeeper, since it is quite stationary and can observe the penalty area lines most of the time very well and use them for precise localization. The accuracy achieved by Odilo is quite remarkable since it traveled long distances across the field and scored several goals. The inferior accuracies of Grimoald (#2) and Hugibert (#3) lead to further investigations and it was found, that both robots were using suboptimal camera parameterizations. Furthermore, Grimoald (#2) was also using a suboptimal color lookup table, and as such failed to produce good classification results for a wide range of images. As a consequence, the localization algorithm failed more often and the achieved accuracy was less good than for the other two robots. However, the achieved accuracies are still quite good and prove that the localization algorithm is robust up to a certain degree of noise and the use of suboptimal camera parameters. The addition of a suboptimal color classifier causes the localization algorithm to be unstable and fail more often by two orders of magnitude.

Table I summarizes also the input data used to test the opponent tracking algorithm. The left, center and right column display the accuracies (RMSE) and the standard deviation of the self-localization, of the ball observations and of the opponent observations of the individual robots. Self-localization errors and inaccuracies often cause errors in ball and opponent observations. As a rule of thumb, the errors for opponent observations are usually greater than the errors for ball observations. This is due to the unique circular shape of a ball. Arbitrary robot shapes hamper the opponent detection routines and as such add an indirect level of noise. Unfortunately the influence of the wrong intrinsic camera parameters of Grimoald and Hugibert on the observations is clearly visible.

The results of the opponent tracking algorithm for all three

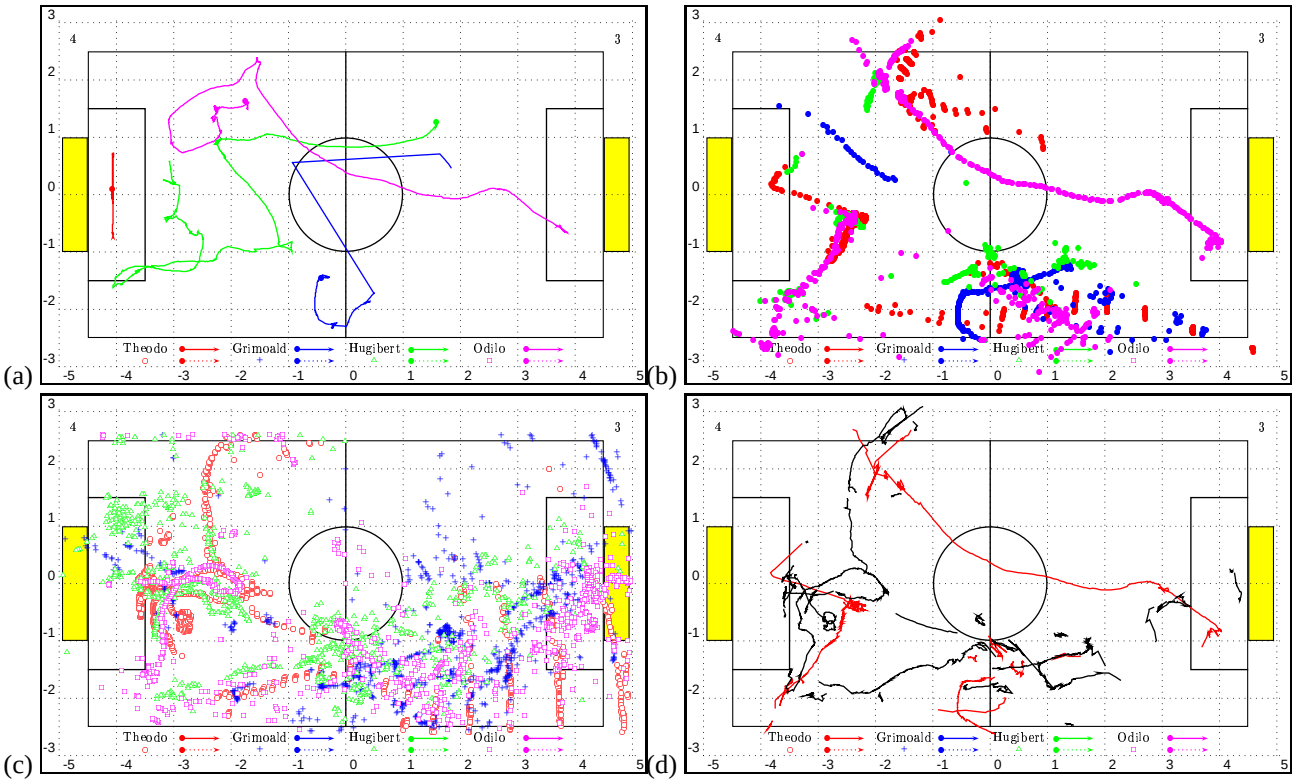


Fig. 12. (a) Trajectories of *The AGILO RoboCuppers*, (b) ball observations of *The AGILO RoboCuppers*, (c) opponent observations of *The AGILO RoboCuppers*, (d) cooperatively estimated trajectories of the ball and of *The Ulm Sparrows*.

matches are displayed in Table II. The column ball and opponent tracks reveal the achieved statistics for the respective object class. Every column displays the percentage of correct tracks, the percentage of incorrect tracks, the track accuracies, and the standard deviation. For every match, the statistics for individual as well as for cooperative perceptions are given. In order to generate the statistics for the individual observations the opponent tracker was run four times using only the observations performed by one robot.

Cooperative perception increases both the percentage of the correctly determined tracks and the accuracy of these tracks. During the match, between 46 to 57 % of the ball's trajectory was detected with an accuracy of 0.19 to 0.23 m. This is a good result, since the ball is often occluded by robots, lifted up by the referee and moved to a new location or shot off the pitch by one of the robots. Opponent tracking worked equally well. On average, 51 % of the opponent tracks were determined correctly by the opponent tracking algorithm. The number of false tracks was reduced to an average of 9 % and the mean track accuracy was 0.35 m. This is also a good result, since broken robots are regularly moved off the field and repaired outside. Furthermore, the opponent goal keeper is usually only observed during an attack.

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 teammates. 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 Fig. 12(d). The three opponent field players were tracked successfully over a period of 120 seconds.

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. We have also seen that maintaining trees of possible tracks is particularly useful for estimating a global state based on multiple mobile sensors with position uncertainty. Finally, we have seen how the state estimation modules of individual robots can cooperate in order to produce more accurate and reliable state estimates.

B. *Action Selection*

The action selection is even more difficult to evaluate. A weak indication of the coherence of coordination is the number of robots performing *go2ball* at the same time. Ideally there should always be exactly one robot going for the ball if the team knows where the ball is. The statistics extracted from the log files of the Seattle tell us that 98.64% of the cycles exactly one robot was going to the ball, in 0.34% no robot, and in 1.02% of the cycles more than one. The average duration that a robot performs *go2ball* or handles the ball without being interrupted

	Self-Localization		Ball Observations		Opponent Observations	
Robot	RMSE (m)	std.dev. (m)	RMSE (m)	std.dev. (m)	RMSE (m)	std.dev. (m)
1. Match						
#1	0.12	0.07	0.31	0.23	0.38	0.26
#2	0.33	0.15	0.38	0.25	0.50	0.26
#3	0.24	0.11	0.24	0.22	0.46	0.26
#4	0.19	0.09	0.25	0.23	0.37	0.25
2. Match						
#1	0.11	0.06	0.33	0.22	0.42	0.26
#2	0.33	0.19	0.35	0.25	0.48	0.27
#3	0.21	0.10	0.28	0.24	0.40	0.24
#4	0.20	0.10	0.25	0.26	0.35	0.24
3. Match						
#1	0.12	0.09	0.27	0.22	0.40	0.26
#2	0.37	0.21	0.34	0.26	0.51	0.26
#3	0.23	0.11	0.26	0.22	0.44	0.25
#4	0.19	0.10	0.18	0.20	0.38	0.25
Mean of all matches						
#1	0.12	0.08	0.29	0.22	0.40	0.26
#2	0.34	0.18	0.36	0.25	0.50	0.26
#3	0.23	0.11	0.26	0.23	0.44	0.25
#4	0.19	0.10	0.21	0.22	0.37	0.25

TABLE I

ACCURACIES ACHIEVED FOR SELF-LOCALIZATION, BALL, AND OPPONENT OBSERVATIONS.

by a decision of a fellow robot is 3.35 seconds. In only 0.25% of the time a robot that is stuck is determined to go for the ball by the other robots. These results suggest that the task assignment algorithm together with our task cost estimation mechanism works well. Buck et al. [11] present more conclusive results obtained in the RoboCup simulation league that show that the team performance using the task assignment algorithm degrades gracefully as the corruption level for communication is increased. They also show how the task assignment algorithm achieves more complex patterns of cooperation such as double passes.

Figure 13 shows a kind of situation that has occurred several times during the RoboCup and is replayed in the AGILO simulator. Robot number 2 is supposed to be the fastest to get the ball and therefore approaches the ball (fig. 13a). Near the ball robot 2 collides with an opponent robot. Robot 2 is in a deadlock situation and cannot move forward anymore. The only action feasible to execute remains *get_unstuck*. Thus robot 3 approaches the ball now (fig. 13b) Having reached the ball robot 3 dribbles towards the opponent goal while robot 2 is moving backwards (fig. 13c): Afterwards robot 2 is not stuck anymore and robot 3 is still dribbling.

We have also evaluated our learned hybrid multi robot navigation system in the AGILO robot simulator. To do so, we have compared its performance with the performance obtained by the individual navigation methods. We have performed a bootstrapping t-test based on 1000 different joint navigation tasks to empirically validate that the hybrid navigation planner performs better than the individual planning methods. Based on these experiments we obtained a 99.9% confidence in the test set (99.9% in the training set) that the hybrid method outperforms the potential field method (with its respective parameterization). The respective probabilities for the shortest path method are 99.9% (99.9%), for the maximum clearance method 99.84%

	Ball track				Opponent tracks			
	Corr.	Incorr.	RMSE	std. dev.	Corr.	Incorr.	RMSE	std. dev.
Robot	(%)	(%)	(m)	(m)	(%)	(%)	(m)	(m)
1. Match								
#1	21	8	0.32	0.23	25	13	0.43	0.26
#2	17	11	0.37	0.26	21	20	0.50	0.24
#3	18	14	0.23	0.23	27	19	0.45	0.25
#4	31	9	0.27	0.24	34	13	0.38	0.24
Coop.	46	10	0.23	0.21	57	10	0.35	0.19
2. Match								
#1	12	7	0.33	0.21	20	8	0.46	0.25
#2	16	15	0.31	0.25	19	12	0.46	0.27
#3	20	8	0.32	0.26	28	16	0.40	0.24
#4	28	13	0.29	0.27	31	11	0.36	0.23
Coop.	49	13	0.23	0.22	50	9	0.34	0.21
3. Match								
#1	24	9	0.28	0.22	20	11	0.46	0.25
#2	10	9	0.37	0.26	16	11	0.50	0.25
#3	18	12	0.27	0.22	21	13	0.44	0.25
#4	34	17	0.22	0.22	24	9	0.39	0.25
Coop.	57	16	0.19	0.18	46	8	0.38	0.21
Mean of all matches								
Coop.	51	13	0.21	0.20	51	9	0.35	0.20

TABLE II

ACCURACIES AND COVERAGE ACHIEVED BY THE OPPONENT TRACKER FOR BALL AND OPPONENT TRACKING.

(99.71%), and for the viapoint method 96.25% (94.62%). This validates our hypothesis that the hybrid planner dominates the other planning methods with statistical significance ($\geq 95\%$).

Besides the performance the frequency with which the AGILO system selects actions is also impressive. There are on average 10 action selection cycles per second despite the sophistication of task cost prediction and multi robot navigation. This speed can be reached because the results of complex computations are estimated through neural networks and decision trees that have been trained using experience-based learning mechanisms.

So far our plan-based control mechanisms have only been applied in the AGILO RoboCup simulator and not yet in extensive experiments. Our next steps in the advancement of our action selection mechanisms are the autonomous learning of more complex tasks, such as dribbling towards the opponent goal and shooting in the right moment in the right corner. Another important issue is action selection under uncertainty. The skills that we have learned so far were all acquired with the simulator using a perfect world model. We believe that we can learn much better skills if our simulator can also learn probabilistic models of the AGILO state estimation processes. To learn such a probabilistic perception model we need however a ceiling camera that records the ground truth that the estimated states can be

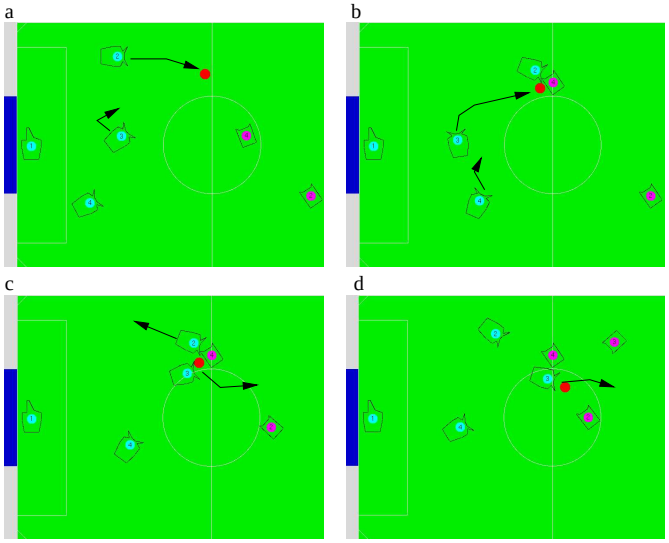


Fig. 13. An example for intelligent cooperation in a real robot soccer environment. Robot 2 approaches the ball (subfi gure a) and thereby collides with a robot of the opponent team (b). As the opponent robot constantly pushes robot 2 is stuck and temporary not regarded by the other robots. Thus robot 3 moves towards the ball while robot 2 tries to get unstuck (c). Finally robot 3 dribbles towards the opponent goal while robot 2 is staying back in its own half (d).

Algorithm	Mean time (1000 problems)	
	μ/sec	signifi cance $P(\mu_{tree} < \mu)$
Simple Potential Field	15.92	99.99 %
Shortest Path	13.14	99.99 %
Maximum Clearance	12.31	99.84 %
Viapoint	11.95	96.25 %
Decision Tree	11.44	

TABLE III

RESULTS OF FOUR EVALUATED ALGORITHMS AND THE TRAINED DECISION TREE. THE SIGNIFICANCE LEVEL IS BASED ON A T-TEST.

compared to. Finally, we believe that in order to acquire more skills more autonomously it is crucial to better integrate learning mechanisms into robot control languages. To this end we extend our plan-based control language RPL such that it is capable of declaratively specifying learning problems within the control routines.

VII. RELATED WORK

The research described in this paper can be discussed with respect to several dimensions. Within the robot soccer application we will compare it with the control techniques employed by other mid-size teams and those employed by teams playing in the simulator and the small size league. In addition, we will compare the system with other autonomous control systems that share the control principles that they apply.

In the mid-size league most competing teams apply behavior-based control techniques and avoid the problem of estimating the complete game state with the CS Freiburg team being a notable exception [35]. However, because the Freiburg team is using Laser range finders as their primary sensors, which are very accurate in depth estimation, they can get away with a simpler state estimation mechanism in which can assume almost perfect sensors with known positions. Most other mid-size teams coordinate the play of their teammates by negotiating or assigning roles to the different players [27]. In contrast, in the AGILO team the coordination is implicit and based on a sophisticated cost estimate for task assignment. The AGILO team is also distinguished in the mid-size league with respect to its extensive use of learning and plan-based control mechanisms. Technologically, the AGILO software shares control mechanisms with teams in the simulator league. In particular, it applies similar learning techniques as the Karlsruhe Brainstormers [42]. The use of such techniques in autonomous robot soccer is much more difficult due to the difficulties in obtaining sufficient training data, high variances in physical effects, extremely noisy sensors, and the incompleteness of available information.

With respect to the software architecture and employed software techniques the AGILO control software shares commonalities with autonomous robotic agents such as the extended RHINO control system [6]. The RHINO system, too, makes extensive use of probabilistic state estimation, has a default mechanism for action selection, and a plan-based control mechanism. The AGILO software extends this work in that it applies these techniques to a multi robot control problem in a very dynamic and adversary environment.

In the research area of autonomous robot soccer several algorithms for probabilistic self-localization have been proposed. Gutmann et al. [21] 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 advantage of using vision sensors is that it is a more general sensor which can be applied to a broader spectrum of applications and environments. A key challenge for vision algorithms is the amount of data they have to process and interpret. Other approaches to vision-based self localization using conventional and omnidirectional camera systems can be found in [47], [27], [30], [34], [1]. 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 [10] also perform model driven localization. However they restrict themselves to perspective and linear models that have limited modeling capabilities. Through the addition of an iterative optimization step our algorithm extends the standard Kalman filter approach to self localization [18] to handle arbitrary non-linear models.

Enderle et al. [17], [1] have developed a vision-based self-localization module for the RoboCup scenario using a sample-based Markov localization method, that is also known as Monte Carlo localization (MCL) [15]. In Fox et al. [19] 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. Our self localization method has the advantage that it is faster as it utilizes Newton iteration with quadratic convergence speed.

Roumeliotis and Bekey [43] 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], [40] have described approaches in which robots actively coordinate their movements in order to reduce cumulative odometric errors. While all theses methods rely on the capability of a team's robots to detect and identify each other correctly, our approach is able to exploit geometric 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 [16], [21] 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 [46]. 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 multiple hypothesis tracking approach requires less computational power if the pruning parameters are carefully selected. Hue et al. [26] are 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 Hingorani [12]. Cox and Leonard [13] 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 much more challenging application domain where we have multiple moving observers with uncertain positions. In addition, we perform object tracking at an object rather than on a feature level. Further applications where multiple hypothesis tracking is used include active global localization of a robot within a topological map [28] and navigation of an autonomous system in unknown environments [33].

VIII. Conclusion

This article has described and discussed the control software of the AGILO autonomous robot soccer team. Similar to advanced autonomous robotic agents acting in human working environments the AGILO team employs sophisticated state estimation and control techniques, including experience-based learning and plan-based control mechanisms.

We have shown that the application of probabilistic state estimation techniques together with information exchange between the robots results in game state estimators that are capable of estimating complete states including robots with surprising accuracy and robustness even with restrictive and noisy camera systems. We have also seen that the ample use of experience-based learning has resulted in powerful control mechanisms, including competent coordination, with little runtime computational cost. The results of the 2001 robot soccer world championship have shown that these techniques allow for competitive soccer play despite an inferior hardware equipment.

There are several research directions that we are currently pursuing and that extend the research reported in this article. First, we have extended an existing control language with constructs for specifying control tasks, process models, learning problems, exploration strategies, etc. Using these constructs, the learning problems can be represented explicitly and transparently and become executable. With the extended language we have started to rationally reconstruct parts of the action selection mechanism that we have described in this article [9].

Another branch of our active research is to add plan-based control mechanisms for improved performance [4], [6], [5]. The key idea of plan-based control is the explicit and transparent representation of control routines such that the action selection mechanism cannot only execute the routines but also reason about and manipulate them. We plan to use these mechanisms to learn and execute more sophisticated plays [8].

In the area of probabilistic game state estimation we have developed a state estimation system using a camera mounted at the ceiling that provides us with ground truth data about the game evolution. We currently, use the data from the ceiling camera to evaluate and analyze our state estimation routines. In particular the data will enable us to learn more informative probabilistic models that will hopefully improve the reliability and accuracy of state estimation.

Finally, we are investigating more general and robust mechanisms for image segmentation and object tracking [22]. These mechanisms should enable the robots to perform the state estimation tasks without making the restrictive assumptions about distinct colors of the objects of interest and replace the current segmentation algorithms based on color labeling [24], [25].

Acknowledgements.: The research reported in this paper is partly funded by the Deutsche Forschungsgemeinschaft in the context of the national research initiative "Cooperative teams of mobile robots in dynamic environments" (contract number: RA359/7-1).

References

[1] 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*, 0(36):103–119, 2001.

[2] K. J. Aström. Optimal control of markov decision processes with incomplete state estimation. *Journal of Mathematical Analysis and Applications*, 10:174–205, February 1965.

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

[4] M. Beetz. Structured Reactive Controllers. *Journal of Autonomous Agents and Multi-Agent Systems*, 4:25–55, March/June 2001.

[5] M. Beetz. Plan representation for robotic agents. AAAI Press, 2002.

[6] M. Beetz, T. Arbuckle, M. Bennewitz, W. Burgard, A. Cremers, D. Fox, H. Grosskreutz, D. Hähnel, and D. Schulz. Integrated plan-based control of autonomous service robots in human environments. *IEEE Intelligent Systems*, 16(5):56–65, 2001.

[7] M. Beetz, S. Buck, R. Hanek, T. Schmitt, and B. Radig. The AGILO autonomous robot soccer team: Computational principles, experiences, and perspectives. In *Procs. of the First International Conference on Autonomous Agents and Multi-agent Systems*, pages 805–812, Bologna, Italy, 2002.

[8] M. Beetz and A. Hofhauser. Plan-based control for autonomous robot soccer. In *Advances in Plan-based Control of Autonomous Robots. Selected Contributions of the Dagstuhl Seminar Plan-based Control of Robotic Agents, Lecture Notes in Artificial Intelligence (LNAI)*. Springer-Verlag, 2002.

[9] M. Beetz, M. Stulp, A. Kirsch, A. Müller, and S. Buck. Autonomous robot controllers capable of acquiring repertoires of complex skills. In *RoboCup International Symposium*, Padova, jul 2003.

[10] A. Blake and M. Isard. *Active Contours*. Springer-Verlag, Berlin Heidelberg New York, 1998.

[11] S. Buck, U. Weber, M. Beetz, and T. Schmitt. Multi robot path planning for dynamic evironments: A case study. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2001.

[12] I.J. Cox and S.L. Hingorani. An Effi cient 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*, 18(2):138–150, February 1996.

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

[14] T. Dean and M. Wellmann. *Planning and Control*. Morgan Kaufmann Publishers, San Mateo, CA, 1991.

[15] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1322–1328, Detroit, Michigan, USA, 1999.

[16] M. Dietl, J.-S. Gutmann, and B. Nebel. Cooperative sensing in dynamic environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1706–1713, Maui, Hawaii, 2001.

[17] 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.

[18] O.D. Faugeras. Three-dimensional computer vision: A geometric viewpoint. *MIT Press*, page 302, 1993.

[19] D. Fox, W. Burgard, H. Kruppa, and S. Thrun. A probabilistic approach to collaborative multi-robot localization. *Autonomous Robots*, 8(3):325–344, 2000.

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

[21] J.-S. Gutmann, T. Weigel, and B. Nebel. A fast, accurate, and robust method for self-localization in polygonal environments using laser-range-finders. *Advanced Robotics Journal*, 14(8):651–668, April 2001.

[22] R. Hanek and M. Beetz. The contraction curve density algorithm: Fitting parametric curve models to images using local self-adapting separation criteria. *International Journal of Computer Vision*, 2004. (to be published).

[23] R. Hanek and T. Schmitt. Vision-based localization and data fusion in a system of cooperating mobile robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1199–1204, Takamatsu, Japan, 2000.

[24] R. Hanek, T. Schmitt, S. Buck, and M. Beetz. Fast image-based object localization in natural scenes. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Lausanne, Switzerland, 2002.

[25] R. Hanek, T. Schmitt, S. Buck, and M. Beetz. Towards robocup without color labeling. In *6th International RoboCup Symposium (Robot World Cup Soccer Games and Conferences)*, volume 6 of *Lecture Notes in Computer Science*. Springer-Verlag, 2002.

[26] C. Hue, J.-P. Le Cadre, and P. Perez. Tracking multiple objects with particle fi ltering. Technical Report 1361, IRISA, 2000.

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

[28] 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)*, pages 13–22, Stockholm, 1999.

[29] Patric Jensfelt and Steen Kristensen. Active global localisation for a mobile robot using multiple hypothesis tracking. *IEEE Transactions on Robotics and Automation*, 17(5):748–760, October 2001.

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

[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)*, pages 2896–2903, 1998.

[32] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA, 1991.

[33] D. Maksarov and H. Durrant-Whyte. Mobile vehicle navigation in unknown environments: a multiple hypothesis approach. *IEE Proceedings: Control Theory & Applications*, 142(4):385–400, 1995.

[34] C. Marques and P. Lima. Vision-Based Self-Localization for Soccer Robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1193–1198, Takamatsu, Japan, 2000.

[35] B. Nebel and T. Weigel. The CS Freiburg 2000 team. In *4th International RoboCup Symposium (Robot World Cup Soccer Games and Conferences)*, Lecture Notes in Computer Science. Springer-Verlag, 2000.

[36] D. Neumann. Bayes pose estimation by modeling colour distributions. Masterthesis, Munich University of Technology, Department of Computer Science, November 2003.

[37] K. Passino and P. Antsaklis. A system and control-theoretic perspective on artifi cial intelligence planning systems. *Applied Artificial Intelligence*, 3:1–32, 1989.

[38] R. Quinlan. Induction of decision trees. In *Machine Learning 1 (1)*, 1986.

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

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

[41] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learining: The rprop algorithm. In *ICCN*, San Francisco, 1993.

[42] M. Riedmiller, A. Merke, D. Meier, A. Hoffmann, A. Sinner, O. Thate, Ch. Kill, and R.Ehrmann. Karlsruhe Brainstormers 2000 - A Reinforcement Learning approach to robotic soccer. In *4th International RoboCup Symposium (Robot World Cup Soccer Games and Conferences)*, Lecture Notes in Computer Science. Springer-Verlag, 2000.

[43] S.I. Roumeliotis and G.A Bekey. Collective localization: A distributed Kalman fi lter approach to localization of groups of mobile robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2958–2965, 2000.

[44] 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*, pages 599–604, Edmonton, Canada, 2002.

[45] T. Schmitt, R. Hanek, M. Beetz, S. Buck, and B. Radig. Cooperative probabilistic state estimation for vision-based autonomous mobile robots. *IEEE Trans. on Robotics and Automation*, 18(5):670–684, October 2002.

[46] D. Schulz, W. Burgard, D. Fox, and A.B. Cremers. Multiple object tracking with a mobile robot. In *Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 371–377, Kauai, Hawaii, 2001.

[47] R. Talluri and J.K. Aggarwal. Mobile robot self-location using model-image feature correspondence. *IEEE Transactions on Robotics and Automation*, 12(1):63–77, February 1996.

[48] S. Thrun. Probabilistic algorithms in robotics. *AI Magazine*, 21(4):93–109, 2000.

Michael Beetz is an Substitute Professor in 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 Ph.D. in computer science from Yale University (1996) and his venia legendi from the University of Bonn in 2001.

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 is serving 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 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 is currently working towards his Ph.D. degree at the 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. He 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.

Sebastian Buck received the M.Sc. in Computer Science in 1999 from the University of Karlsruhe, Germany. Since 1999 he is a Research Assistant at Munich University of Technology, Germany where he is working to complete his Ph.D., and is teaching courses in machine learning. His research interests include control, coordination, and simulation of automotive systems in dynamic environments. Mr. Buck has been involved in the RoboCup teams *Karlsruhe Brainstormers* and *AGILO RoboCuppers* as well as in the development of biologically plausible neural models and the multi robot simulation environment M-ROSE.

Freek Stulp received the M.Sc. in Artificial Intelligence (Cum Laude) from the Rijksuniversiteit Groningen, the Netherlands, in 2001. His Master's Thesis on 3D reconstruction of occluded surfaces was done whilst working as a Research Assistant at the University of Edinburgh, Scotland, and as a Visiting Researcher at the Instituto Superior Tecnico in Lisbon, Portugal. Mr. Stulp has been a Research Assistant at the Munich University of Technology, Germany, since May 2002, where he is working towards his Ph.D. in the field of reinforcement learning, planning, and action modelling, applied to autonomous robot systems.

Derik Schröter received the M.Sc. in Electrical Engineering from the Technical University Hamburg-Harburg, Germany, in 2000. He did his Master Thesis on object tracking at the Munich University of Technology, Germany. There he is working since 2001 as a Research Assistant in the research group *Intelligent Autonomous Systems*. The focus of his work is map building in the context of mobile autonomous robotics. His research interest include 2D mapping, vision based 3D reconstruction, localization, autonomous exploration and navigation.

Bernd 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. Since 1986 he is a full professor at the Department of Computer Science of the Munich University of Technology. Since 1988 he is Chief Managing Director of the Bavarian Research Center for Knowledge Based Systems (FORWISS) and head of the Image Understanding Group, there. Since 1993 he is chairman of *abayfor*, a cooperation of about 20 research networks in Bavaria, doing 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.