CS Freiburg: Coordinating Robots for Successful Soccer Playing

T. Weigel, J. -S Gutmann, M. Dietl, A. Kleiner and B. Nebel

Post Print

N.B.: When citing this work, cite the original article.

CS Freiburg: Coordinating Robots for Successful Soccer Playing

Thilo Weigel, Jens-Steffen Gutmann, Markus Dietl, Alexander Kleiner, Bernhard Nebel

Abstract—**Robotic soccer is a challenging research domain because many different research areas have to be addressed in order to create a successful team of robot players. This paper presents the *CS Freiburg* team, the winner in the middle size league at RoboCup 1998, 2000 and 2001. The paper focuses on multi-agent coordination for both perception and action. The contributions of this work are new methods for tracking ball and players observed by multiple robots, team coordination methods for strategic team formation and dynamic role assignment, a rich set of basic skills allowing to respond to large range of situations in an appropriate way, an action selection method based on behavior networks as well as a method to learn the skills and their selection. As demonstrated by evaluations of the different methods and by the success of the team, these methods permit the creation of a multi-robot group, which is able to play soccer successfully. In addition, the developed methods promise to advance the state of the art in the multi-robot field.**

I. INTRODUCTION

In 1993, Mackworth proposed robotic soccer as an application for demonstrating integration of methods from AI, vision, and robotics [22]. The *RoboCup* initiative [19] went one step further and proposed to use this domain as a benchmark problem for AI and robotics, and they started to organize international competitions. Nowadays, RoboCup comprises a scientific symposium, robotic demonstrations, and competitions with real and simulated robots in different leagues; and it enjoys high popularity among researchers and in the general public.

The *CS Freiburg*[1] team participates since 1998 in the middle-size real robot league (F2000). In this league a maximum number of 4 autonomous robots per team with a footprint not greater than 2000 mm^2 compete on a field of approximately 9×5 meters fully surrounded by 50 cm high walls[2]. A game lasts 2×10 minutes. The particular challenge in this league is to cover a whole spectrum of research issues ranging from robotic hardware development and low level sensor interpretation up to planning and multi-agent coordination.

In the F2000 league, global sensing systems, e.g., a camera capturing the whole scene from a bird's eye of view, are prohibited and only local perception is allowed. However, wireless communication between players and with processing units outside the soccer field is allowed and many teams make extensive use of it. All players of a team are usually started and stopped by wireless communication. Once a game is started, no further human interaction is allowed and all decisions have to be taken autonomously by the robots.

The *CS Freiburg* team has competed four times at international RoboCup competitions. The team came in third at one tournament [37], [26] and won the World Champion title three times [1], [15], [28], [39], [4], [40].

This paper presents the *CS Freiburg* team as a case study of a successful robotic soccer team with a focus on coordination in both, perception and action. The main contributions we will focus on are:

- new methods for tracking the ball and players observed by multiple robots,
- team coordination methods for strategic team formation and dynamic role assignment,
- a rich set of basic skills allowing to respond to large range of situations in an appropriate way,
- an action selection method based on behavior networks, and
- a learning method to adapt to new hardware and to new environments.

We will not describe our self-localization method based on laser scans [17], [15]. Instead, we will only note that we get almost perfect self-localization in the particular environment our robots are acting in.

The paper is organized as follows. Section II gives an overview of the hardware and the software architecture of the *CS Freiburg* team. The perception technology and cooperative sensor interpretation approach is presented in Section III. Section IV illustrates the team coordination mechanism including dynamic role assignment and team positioning. The player's basic skills and the method of selecting them are described in Section V. Section VI concludes the paper.

II. OVERVIEW

A. Hardware

The basis of the *CS Freiburg* soccer robots are *Pioneer I* robots as manufactured by *ActivMedia Robotics*. However, they have been heavily modified and enhanced to meet the special requirements of soccer playing. Figure 1 shows one of the players.

Fig. 1. A CS Freiburg player.

---

[1] *CS Freiburg* stands for **C**omputer **S**cience Department, University of **Freiburg**. In addition, it is a pun on *SC Freiburg*, a famous local (human) soccer team.

[2] Note, that from 2002 on, all walls around the field are removed.

Equipped with a *Pioneer II* controller board instead of the original one, a robot is able to move considerably faster. A caster roller instead of the original rear caster wheel allows for more precise motion control. Furthermore, nickel-cadmium batteries are utilized because of their light weight and high speed charging capability.

For ball handling and shooting a kicking device with movable ball steering flippers is incorporated. A close-up of the device is shown in Figure 2. The kicking plate is strained by a wind-screen wiper motor and released by a solenoid. The springs pressing on the plate are strong enough to produce a kick that shoots the ball well over the whole field. Two DC-motors allow to turn the flippers to an upright position and back. As the flippers are only needed for controlling the ball, they are turned upwards when the ball is not present in order to decrease the risk of entanglements with other robots [39].

Fig. 2. Close-up of the kicking device.

For self-localization and recognition of other robots on the field a *SICK LMS200* laser range finder is employed. It provides depth information for a 180° field of view with an angular resolution of 0.5° and an accuracy of 1 cm [26].

The ball is perceived by a *Sony DFW-V500* digital camera. Frames are provided in YUV format with a resolution of 320x240 pixels and are processed at a frame rate of 30fps [40].

The "brain" of a robot is a *Sony Vaio PCG-C1VE* notebook. Via the firewire interface it connects directly to the camera. Serial-to-USB converters are necessary to connect to the motor controller board (by RS232) and the laser range finder (by RS422). As no commercially available RS422-to-USB converter is capable of the range finder's 500Mbaud rate, a custom made one has been built by *SICK AG*.

The described hardware setup allows a player to play in a fully autonomous way. However, for coordinated perception and coordinated team play it communicates with its teammates and an off-field (standard) computer using a *WaveLan* 11 Mbit/s (2.4 GHz) PCMCIA-card.

B. Software Architecture

The *CS Freiburg* players are capable of playing in a fully autonomous way based solely on information from their own sensors. Additionally, they are able to exploit the possibility of communicating with their teammates during a game. Unfortunately, wireless communication can be unreliable in many cases

and teams relying on it ran into severe problems in the past. *CS Freiburg's* design is to rely only on information gathered locally on-board of each robot but to benefit from information broadcasted by wireless communication if available.

Communication enables the players to organize themselves as a team and to benefit from the sensory information of their teammates. Figure 3 depicts the team architecture. Via radio Ethernet link the players communicate with an off-field computer. A *global sensor integration* module running on the off-field computer[3] integrates the player's local information about the world into one consistent global world model. In turn the global world model is distributed among the players giving them the possibility to enhance their local models of the world. A *graphical user interface* visualizes the global world model together with various states of the robots and allows to send commands to the players, i.e., for starting and stopping the game.
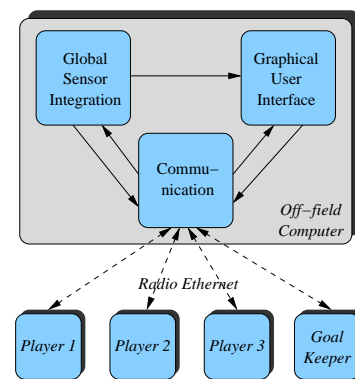

Fig. 3. Team architecture.

Figure 4 shows the architecture of a player. The *perception* module takes information from the sensors and – if available – from the global world model and maintains a local world model. The local world model provides the basis for the *action selection* mechanism that decides which action from a set of *basic skills* should be carried out by the player. The *strategy* component considers both the world model and messages from the teammates, and ensures that an action according to the player's current team role is selected. All modules are activated every 100 msec in order to determine the next action to execute.
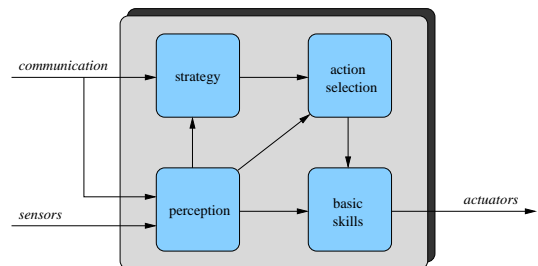

Fig. 4. Player architecture.

---

[3]Note, that the global sensor integration could be performed locally on a player as well.

III. Perception and Cooperative Sensor Interpretation

A. Perception system

The perception system of the *CS Freiburg* robot players is based on laser range finder data and vision information from a fixed-mounted monocular camera. Figure 5 depicts the perception module which is the core of each robot. Briefly, a robot first localizes itself by extracting line segments from a scan taken by the range finder and matching them with a hand-crafted *a priori* line model of the RoboCup environment. Only lines of a certain minimum extent are taken into account to discard other players present in the sensor's field of view. The scan-matched position is fused in a Kalman filter with the estimate from odometry [15], [16]. Experiments show that this localization technique is very accurate (with an error of about 2 cm and 1°) and that it is faster and more robust than competing methods [17].
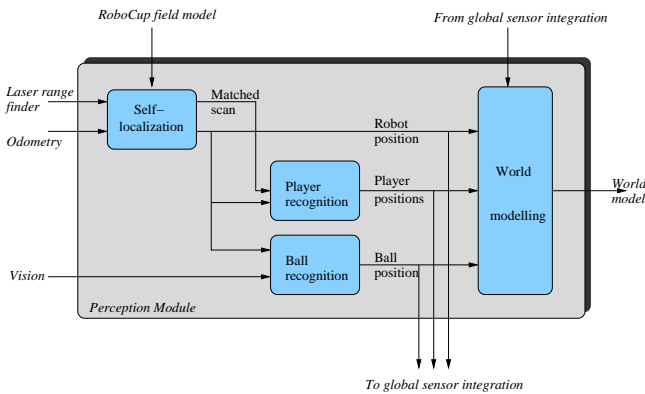

Fig. 5. Perception module running on each robot.

As in 2002 the field walls were replaced by poles surrounding the playing field the described technique was extended to extract line segments by identifying the gaps between neighbouring poles in the scan. For this, a scan is clustered and a "virtual line" is recorded if the distance between two clusters matches the known pole distance. Virtual lines lying on the same indefinite line are then merged to "candidate lines" for being matched with the field outline. However, only those candidate lines are matched which have a certain minimum length and consist of a sufficient number of virtual lines. Additionally, a candidate line is filtered away if it is partially or completely covered by a candidate line closer to the scan's center (see Figure 6).

In anticipation of further rule changes leading to a reduction of the number of field poles the monte-carlo-localization technique [34] was adopted as a localization method which will still be applicable, even if the number of poles will be cut in half [13].

After a robot has localized itself, players are extracted from the scan by removing all points belonging to the field walls and clustering the remaining ones. For each cluster the center of gravity is computed and considered as the center of a player. Inherent to this approach is the systematical error due to the different shapes of the robots [38]. At least for some players this error can be reduced, e.g., by assuming that the opponent goalkeeper is usually oriented parallel to the goal line, adding a
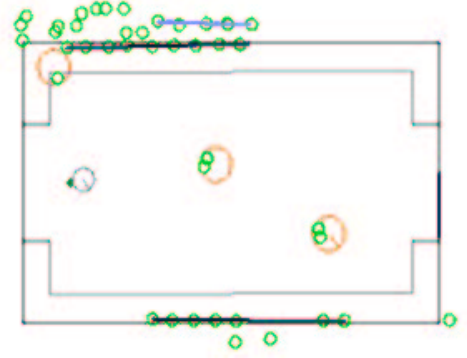

Fig. 6. Screenshot of how the goalkeeper localized itself and recognized three objects on the field. The small circles correspond to clustered scanpoints, the light lines are candidate lines which are filtered away and the dark lines are candidate lines which are matched with the field outline.

constant offset to the center of gravity generally reduces the position error for the opponent goalie[4]. From the extracted players, one cannot decide which is friend or foe. This is the task of the multi-robot sensor integration described in the next section.

Since the laser range finders are mounted at a level that prohibits the detection of the ball, a camera is used for obtaining information about the ball. Algorithms in the CMVision software library [6] are utilized for extracting color-labeled regions (called *blobs*) from YUV images taken by the camera and employing previously learned color tables.

A filter tests the plausibility of blobs and discards the ones whose shape, size or position make it very unlikely to correspond to the ball. From the remaining blobs the one closest to the previously selected blob is chosen and various properties such as center, radius, or size in pixels are determined.

From the computed blob center the global ball position is determined by using an off-line learned mapping table that contains distance and heading pairs for each pixel. This lookup table is autonomously learned by the robot before the game by positioning itself at various position on the field and taking measurements of the ball which stays at a fixed position. Interpolation is then used to compute the distance and heading pairs for pixels where no explicit information is available [32].

Despite of the applied filters, wrong ball measurements still occurred due to reflections on shiny surfaces or poorly trained colors, e.g., the white field markings appeared to have a similar color as shiny reflections on the ball. In order to detect such wrong ball measurements, a global sensor integration module integrates the observations of all players.

B. Multi-Robot Sensor Integration

All players send their own position, the position of observed other players and the position of the ball (if detected) to the global sensor integration module (see Figure 3). For each object on the field (own players, opponent players and the ball) the module maintains a track where observations sent by the robots are associated with and fused to.

For player observations, tracks containing position measure-

---

[4]Of course this offset depends on the shape of the opponent's goalie and has to be adjusted before a game.

ments from an own player are marked as *teammate*, all others as *opponent*.

The ball position is determined by a probabilistic integration of all ball measurements coming from the players. A combination of Kalman filtering and Markov localization as described below is used for achieving maximum accuracy and robustness.

If no measurements can be associated with a track for a certain amount of time (e.g., 5 seconds in the current implementation), the track is deleted.

Because the global model integrates more data than a single player is able to perceive, it should be more accurate and comprehensive than any local world model of a single player. The fused positions of players and ball are sent back to all robots on a regular basis where they are integrated into each robot's world model. For the integration, only objects currently not observed by the robot are considered. Each robot usually knows its own pose with high accuracy, therefore the information about a robot's pose in the global model is not taken into account. Furthermore, since the information in the global model is usually 100 - 200 ms old, information about objects observed directly by the robot's sensors is more recent and more accurate than the one contained in the global model. Therefore, only objects not observed by the robot are integrated from the global model. Additionally, the identification of players (teammate or opponent) is taken from the global model.

Using the sensor information of the whole team enables a player to consider hidden objects when selecting an action. This proved to be especially advantageous for the global ball position, since the ball is seen almost all the time by at least one player. Furthermore, knowing whether an observed object is a teammate or an opponent is, of course, very helpful for an effective cooperative team play.

C. Multi-Player Tracking from Reliable Data

For each object detected by one of the players the on-board perception system computes heading and velocity information based on the last few observations belonging to this object using differentiation. Each observation is then communicated to the multi-sensor integration module. Thus, the observation model is a random variable $\mathbf{x}_s = (x_s, y_s, \theta_s, v_s, \omega_s)^T$ with mean $\hat{\mathbf{x}}_s$ and covariance Σ_s where the state vector contains the object's position, heading and translational and rotational velocities.

As the *CS Freiburg* robots know their own position with high accuracy and the LRF provides accurate data, it is assumed that for player observations Σ_s is a constant diagonal matrix with the diagonals manually determined through experiments.

Whenever a robot sends information about a player which can't be associated with an already existing track, i.e., if the distance to all existing tracks exceeds a certain threshold, a new track is initiated. Tracks are modeled as Gaussian variables $\mathbf{x}_r$ with mean $\hat{\mathbf{x}}_r$ and covariance Σ_r. Thus, when initiating a new track, it is set to

$$\hat{\mathbf{x}}_r = \hat{\mathbf{x}}_s, \qquad \Sigma_r = \Sigma_s. \qquad (1)$$

For predicting the state of a track over time, a simple motion model is used with the assumption that the object moves and rotates with constant speed. Given a certain time interval t, the

track is projected according to

$$\hat{\mathbf{x}}_r \quad \leftarrow \quad F_s(\hat{\mathbf{x}}_r, t) = \begin{pmatrix} \hat{x}_r + \cos(\hat{\theta}_r)\hat{v}_r t \\ \hat{y}_r + \sin(\hat{\theta}_r)\hat{v}_r t \\ \hat{\theta}_r + \hat{\omega}_r t \\ \hat{v}_r \\ \hat{\omega}_r \end{pmatrix}, \qquad (2)$$

$$\Sigma_r \quad \leftarrow \quad \nabla F_s \Sigma_r \nabla F_s^T + \Sigma_a(t), \qquad (3)$$

where ∇F_s is the Jacobian of F_s and $\Sigma_a(t)$ is the covariance of some additive Gaussian noise with zero mean:

$$\Sigma_a(t) \quad = \quad diag(\sigma_{x_a}^2 t, \sigma_{y_a}^2 t, \sigma_{\theta_a}^2 t, \sigma_{v_a}^2 t, \sigma_{\omega_a}^2 t) \qquad (4)$$

with σ_{x_a}, σ_{y_a}, σ_{θ_a}, σ_{v_a} and σ_{ω_a} being some constant standard deviations estimated through experiments.

Now, when a new measurement $\hat{\mathbf{x}}_s$ arrives from one of the robots which corresponds to a track $\mathbf{x}_r$, observation and track are fused according to:

$$\hat{\mathbf{x}}_r \quad \leftarrow \quad (\Sigma_r^{-1} + \Sigma_s^{-1})^{-1}(\Sigma_r^{-1}\hat{\mathbf{x}}_r + \Sigma_s^{-1}\hat{\mathbf{x}}_s), \qquad (5)$$

$$\Sigma_r \quad \leftarrow \quad (\Sigma_r^{-1} + \Sigma_s^{-1})^{-1}. \qquad (6)$$

Note, that since the sensor model does directly observe the system state, the simplified Kalman filter equations found in Maybeck [24] can be utilized.

The success of a Kalman filter depends on a reliable data association method. In the current implementation a geometric method developed by Veloso *et al.* [36] is used. The method assigns measurements to tracks by minimizing the sum of squared error distances between observations and tracks. Although this geometric method already yields reasonable results in practice, it should be noted that the application of a probabilistic method such as joint probabilistic data association filters (JPDAF) [2], [8], [27] might still improve the results.

D. Ball Tracking from Noisy Data

Tracking the ball from observations of multiple robots is similar to tracking different players but there are some notable differences. Data association is easier as there can be only one ball in the field during a game. However, since ball recognition usually employs vision, measurements are less accurate and can be in few cases unreliable, e.g., when observing false positives due to reflections or poorly trained colors.

Since the vision sensor is only able to determine the heading to the ball with good accuracy but fails to provide accurate range data, especially if the ball is far away from the robot, the covariance Σ_b of a ball observation $\hat{\mathbf{x}}_b$ depends on the distance to the ball. Given the range $\hat{r}_b = \sqrt{(\hat{x}_b - \hat{x}_{rob})^2 + (\hat{y}_b - \hat{y}_{rob})^2}$ and heading $\hat{\phi}_b = \tan^{-1}((\hat{y}_b - \hat{y}_{rob})/(\hat{x}_b - \hat{x}_{rob})) - \hat{\theta}_{rob}$ of the ball with respect to the robot located at pose $(\hat{x}_{rob}, \hat{y}_{rob}, \hat{\theta}_{rob})^T$, the uncertainty $\Sigma_{r\phi}$ of the ball position is modeled as

$$\Sigma_{r\phi} \quad = \quad diag(\hat{r}_b \sigma_{r_b}^2, \sigma_{\phi_b}^2), \qquad (7)$$

where σ_{r_b} and σ_{ϕ_b} are some constant standard deviations determined empirically. From this, the observation error can be computed as

$$\Sigma_b \quad = \quad \nabla P \Sigma_p \nabla P^T, \qquad (8)$$

where

$$P(\hat{r}_b, \hat{\phi}_b, \hat{\theta}_b, \hat{v}_b, \hat{\omega}_b) = \begin{pmatrix} \hat{x}_{rob} + \hat{r}_b \cos(\hat{\theta}_{rob} + \hat{\phi}_b) \\ \hat{y}_{rob} + \hat{r}_b \sin(\hat{\theta}_{rob} + \hat{\phi}_b) \\ \hat{\theta}_b \\ \hat{v}_b \\ \hat{\omega}_b \end{pmatrix},$$

$$\Sigma_p = diag(\hat{r}_b \sigma_{r_b}^2, \sigma_{\phi_b}^2, \sigma_{\theta_b}^2, \sigma_{v_b}^2, \sigma_{\omega_b}^2),$$

and σ_{θ_b}, σ_{v_b} and σ_{ω_b} are further constant standard deviations estimated by experiments.

Initiation of a new track $\mathbf{x}_r$ from a ball observation is performed according to:

$$\hat{\mathbf{x}}_r = \hat{\mathbf{x}}_b, \qquad \Sigma_r = \Sigma_b. \tag{9}$$

For predicting the ball state over time a similar function as for player movements is used but the assumption is taken that the ball rolls on a straight line and slows down with deceleration a_b,

$$\hat{\mathbf{x}}_r \leftarrow F_b(\hat{\mathbf{x}}_r, t) = \begin{pmatrix} \hat{x}_r + \cos(\hat{\theta}_r)(\hat{v}_r - a_b t')t' \\ \hat{y}_r + \sin(\hat{\theta}_r)(\hat{v}_r - a_b t')t' \\ \hat{\theta}_r \\ \hat{v}_r - a_b t' \\ \hat{\omega}_r \end{pmatrix}, \tag{10}$$

$$\Sigma_r \leftarrow \nabla F_b \Sigma_r \nabla F_b^T + \Sigma_a'(t), \tag{11}$$

where $t' = \min(t, \hat{v}/a_b)$ and $\Sigma_a'(t)$ is a similar constant covariance matrix as $\Sigma_a(t)$ to flatten the Gaussian distribution over time. Finally, fusing new observations to this track is analogous to equations (5) and (6).

The Kalman filter for ball tracking presented in this section assumes noisy but reliable data, that is, no outliers are integrated. However, sometimes completely wrong ball measurements were observed by one of the robots due to reflections on walls or poorly trained colors. One possibility to filter out such outliers is to use a validation gate that discards measurements whose Mahalanobis distance is larger than a certain threshold d, where d is chosen from a χ^2 distribution.

Such a validation gate, however, has the problem that when one robot is constantly sending out wrong observations and the Kalman filter for some reason is tracking these wrong observations and filters out all others, the global ball position becomes unusable. Furthermore, when the robot stops sending wrong observations it takes several cycles until other observations are taken into account again. For these reasons a more sophisticated filter method is employed which is described in the next section.

E. Markov Localization as Observation Filter

In localization experiments carried out on the mobile robot *Rhino* [33], it became evident that Markov localization is more robust, while Kalman filtering is more accurate when compared to each other [14]. A combination of both methods is likely to provide a maximum robust and accurate system.

For ball localization, this result is utilized by employing a Markov process as an observation filter for the Kalman filter. A grid-based approach with a 2-dimensional (x, y) grid is used where each cell z is associated with the probability $p(z)$ that the ball is in this cell. The grid is initialized with a uniform distribution before any observation is processed. The integration of new ball measurements is then done in two steps: prediction and update.

In the prediction step, ball motion is modeled by a conditional probability $p(z \mid z')$ which denotes the probability that the ball is at position z given that it was at position z'. Upon ball motion, the new ball position is calculated as:

$$p(z) \leftarrow \sum_{z'} p(z \mid z')p(z'). \tag{12}$$

Grid-based Markov localization can be computational expensive if the size and especially the dimension of the grid is large. For efficiency, only a 2-dimensional grid is used that does not store any heading or velocity information of the ball. This means that the position cannot accurately be estimated when the ball is moving. For the motion model $p(z \mid z')$ it is assumed that all directions are equally possible and velocities are normally distributed with zero mean and covariance σ_v^2. Therefore, $p(z \mid z')$ can be expressed as a Gaussian distribution around z':

$$p(z \mid z') \sim N(z', diag(\sigma_v^2 t, \sigma_v^2 t)), \tag{13}$$

where t is the time passed during ball motion.

In the update step, a new ball observation z_b is fused into the probability distribution according to Bayes' law:

$$p(z) \leftarrow \frac{p(z_b \mid z)p(z)}{\sum_{z'} p(z_b \mid z')p(z'),} \tag{14}$$

The sensor model $p(z_b \mid z)$ determines the likelihood of observing z_b given the ball is at position z. It is modeled according to:

$$p(z_b \mid z) \sim N(\hat{z}_b, \Sigma_b'), \tag{15}$$

where $\hat{z}_b$ are the (x, y) components of ball observation $\hat{\mathbf{x}}_b$ as defined in Section III-D and Σ_b' is the upper left 2×2 sub matrix of Σ_b as calculated in equation (8).

Maintaining the multi-modal probability grid makes it very easy to distinguish which ball measurement should be integrated by the Kalman filter and which not. After updating the grid with a new measurement the most likely ball position is determined, that is, the cell with the highest probability. Only measurements that are close to the most likely position are fused into the Kalman filter and all others are considered as outliers. Furthermore, if the current state of the Kalman filter does not correspond to the most likely ball position in the grid, the Kalman filter is re-initialized using this position.

By using this dual probabilistic localization method high accuracy is achieved through Kalman filtering together with high robustness through Markov localization. One might argue that Markov localization alone could be sufficient for localizing the ball. However, since positions are discretized into a grid and the ball position cannot be accurately estimated on ball motion due to the 2 dimensionality of the grid, the resulting position would be less accurate than the one from the combined method.

F. Results

In order to demonstrate the performance of the ball tracking algorithm, let us consider an ambiguous situation such as the one in Figure 7. Two robots, player 1 and 3, observe the ball at the true location in front of the goal but one robot, player 2, gets it all wrong and thinks the ball is somewhere on the center line.
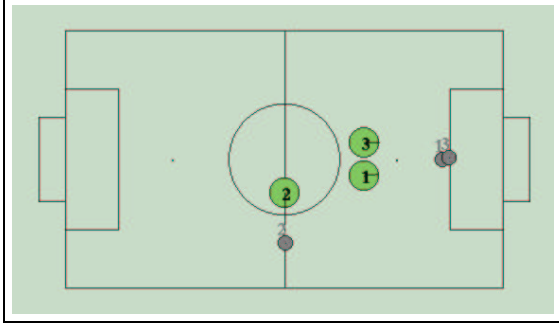

Fig. 7. Player 2 observes a false positive on the center line.

Assuming that all three players send their ball observations to the global sensor integration module on a regular basis, the probability distributions as shown in Figure 8 result.
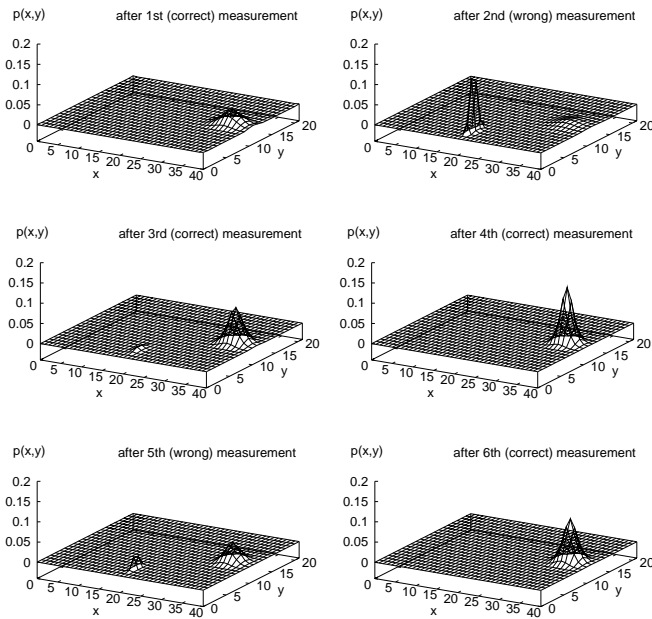

Fig. 8. Evolution of the position probability grid.

When integrating the first three measurements, all of them are fused by the Kalman filter since none of them has been detected to be an outlier yet. Note that after updating the grid with the second measurement, the probability distribution has a sharp peak at the center line caused by the low uncertain measurement of player 2 which thinks the ball is close. After integrating more measurements, the probability distribution concentrates more and more on the true ball location and further measurements from player 2 (left graph on bottom row in Figure 8) cannot out-weigh the true location anymore. Thus, after the first integration of observations from all players, subsequent read-

ings from player 2 are filtered out and the Kalman filter tracks the ball based on observations from player 1 and 3 only.

In order to verify that the ambiguous situation as described above is not an academic case, data recorded from competition games have been reviewed. From the 118,388 ball observations during RoboCup 2000, 938 of these observations (= 0.8%) were excluded using the described kind of filtering. The implications from such a filtering and more information about the accuracy of the tracking methods can be found in our paper on cooperative sensing [9].

IV. TEAM COORDINATION

Soccer is a complex game where a team usually has to meet several requirements at the same time. To ensure that in any game situation a team is prepared to defend its own goal, but also ready to attack the opponent goal, the various team players have to carry out different tasks and need to position themselves at appropriate strategic positions on the field.

To express that a player has a task which is related to a position in the team formation a player is said to pursue a certain *role* [29]. Distinguishing between different areas of responsibility the following roles are established:

- *active*: the active player is in charge of dealing with the ball. The player with this role has various possible actions to approach the ball or to bring the ball forward towards the opponent goal.
- *strategic*: the task of the strategic player is to secure the defense. It maintains a position well back in its own half.
- *support*: the supporting player serves the team considering the current game situation. In defensive play it complements the team's defensive formation and in offensive play it presents itself to receive a pass close to the opponents goal.
- *goalkeeper*: the goalkeeper stays on its goal line and moves depending on the ball's position, direction and velocity.

As the goalkeeper has a special hardware setup for its task, it never changes its role. The three field players, however, are mechanically identical and switch their roles dynamically whenever necessary.

A. Preferred Poses

The approach of *CS Freiburg* for determining a *preferred pose* for each field player *role* is similar to the SPAR method of the *CMU* team in the small size league [30]. From the current situation as observed by the players, a potential field is constructed which includes repulsive forces arising from undesirable positions and attracting forces from desirable ones.

Figure 9 shows an example of a potential field for the preferred position of the active player. Dark cells indicate very undesirable positions whereas light positions represent very desirable positions. The resulting position is marked white. The ideal position for the active player is considered to be at least a certain distance away from other players and at an optimum distance and angle to the ball. While the optimum distance is fixed, the optimum angle is determined by interpolating between a defensive and an offensive variant depending on the ball's position. A defending player should be placed between the ball and the own goal, but in offensive play the ball should be between the player

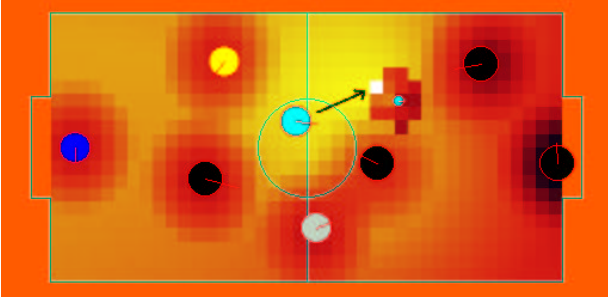and the opponent goal and the player should face the opponent goal.

Fig. 9. Potential field for determining the active position.

Figure 10 shows the potential field for the desired position of the strategic player. It is based on the same game situation and uses the same colors as the example for the active player. The strategic player is supposed to stay well behind all players and the ball and should prefer central positions with regard to its own goal. Only the active player is assigned a repulsive force explicitly in order to enforce staying out of its way. Other players are avoided implicitly by the path planner which finds an appropriate position close to the desired one.
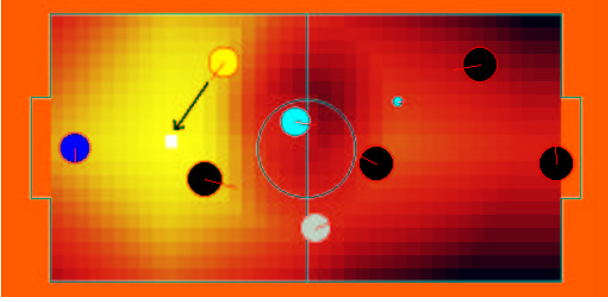

Fig. 10. Potential field for determining the strategic position.

Figure 11 shows how in the same game situation as above the defensive support position is determined. The supporter should stay away from all other players and at a certain distance to the ball. As the supporting player should complement the team's defensive formation, additionally positions behind and aside the active player are preferred.
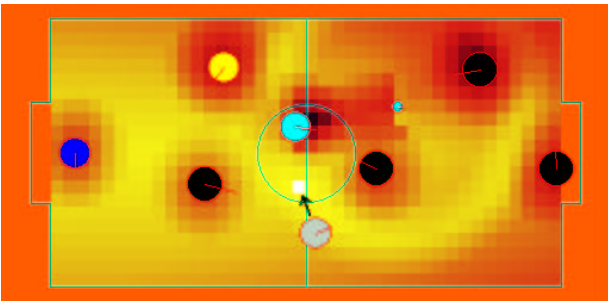

Fig. 11. Potential field for determining the support position.

To avoid "overreacting" to the constantly changing environment a player's current pose is favored with a persistence value.

Additionally, the tolerances for reaching the preferred pose of the defending players are determined dynamically depending on how much turning is required to move to these poses. By allowing large tolerances for large angles but requiring small tolerances for small angles it is achieved that a player only starts to update its pose if the new pose differs considerably from the old one. But once the player is moving towards that preferred pose it intends to approach it accurately.

To reach a preferred pose, a player follows a collision-free trajectory generated by a path planning system which constantly (re)plans paths based on the player's perception of the world. The system is based on potential fields and uses A^* search for finding its way out of local minima [35], [40]. In order to avoid interfering with the active player, the strategic and supporting player are adding extra sources of repulsive forces ahead of the active's way. Furthermore, to reduce collisions between teammates the *prioritized path coordination method* [21], [3] was adopted. The players communicate their current paths to their teammates and check for potential collisions in the future. Whenever a collision is detected the robot with the lower priority reduces its velocity such that collisions with higher prioritized robots are avoided. The player's priorities are derived from their current role, assigning the highest priority to the active and the lowest to the supporting player [12].

B. Roles

After a field player has determined the best active, strategic and support poses from its perspective, it estimates utilities for each role, which are based on the role itself and on an approximation for the time it would take the player to reach the corresponding preferred pose. The utilities are determined by taking into account the robot pose $(x_r, y_r, \theta_r)^T$, the ball position $(x_b, y_b)^T$, the preferred pose $(x_p, y_p, \theta_p)^T$, and positions $(x_o, y_o)^T$ of other objects from the set of objects O in the player's world model. In order to simplify notation, positions (x_i, y_i) are abbreviated by vectors $\vec{v_i}$. $\phi(\vec{v_i}) = tan^{-1}(y_i/x_i)$ denotes the orientation of a vector and $|\vec{v_i}| = \sqrt{x_i^2 + y_i^2}$ its length. The utility for a preferred pose p is calculated from the following constituents:

- Distance to target position: $u_d = |\vec{v_p} - \vec{v_r}|$
- Angle necessary for the robot to turn towards the preferred position: $u_{t_r} = |\phi(\vec{v_p} - \vec{v_r}) - \theta_r|$
- Objects between the player and the target position: $u_o = \min_{o \in O}(|\vec{v_o} - \vec{v_r}||\sin(\phi(\vec{v_o} - \vec{v_r}) - \phi(\vec{v_p} - \vec{v_r}))|)$
- Angle necessary to turn the robot at the preferred position into the orientation of the preferred pose. The target orientation is either the bearing to the opponent's goal (role *active*) or the bearing to the ball (roles *strategic* and *support*): $u_{t_p} = |\phi(\vec{v_p} - \vec{v_r}) - \theta_p|$

The total utility $U(p)$ for the preferred pose p is now computed as the weighted sum of all criteria

$$U(p) = w_d f_d(u_d) + w_{t_r} f_{t_r}(u_{t_r}) + \\ w_o f_o(u_o) + w_{t_p} f_{t_p}(u_{t_p}), \quad (16)$$

where the weights w_i sum up to 1 and f_i are fuzzy functions yielding values in the interval $[0, 1]$. The value returned by a function f_i is the higher, the better the value of its argument is

considered to be for the player. Finally, the utilities are weighted by the importance of the role, i.e., the *active* role is more important than the *strategic* role, which in turn is more important than the *supporter* role.

In order to decide which role to take a player sends the utilities estimated for each role to its teammates and compares them with the received ones. Following a similar approach taken by the *ART* team [7], each player's objective is to take a role so that the sum of the utilities of all players is maximized - under the assumption that all other team players do the same. In contrast to the *ART* approach a player doesn't take its desired role right away, but checks first if no other player is currently pursuing the same role and considering that role best for itself as well. As the world models of the players are not identical their perspectives can in fact differ. Therefore a player only takes a role if either no other player is currently pursuing that role or the pursuing player signals that it actually wants to change its role. That way with only little extra communication effort the number of situations is reduced where more than one player have the same role.

A problem for this approach are situations where different players come up with very similar utilities for a certain role and the roles might oscillate. However, by adding a hysteresis factor to the utility of a player's current role it is ensured that a player only gives up a role if its real utility for that role is clearly worse than the one of its teammate.

As this approach depends heavily on communication, a "fall-back" strategy is implemented for situations in which communication is not possible or not working. A CS Freiburg player detects these situations by monitoring the time it last received a message from the global world model. If the last message is too old, it assumes a communication malfunction and limits its area of play to a predefined *competence area* [16]. The competence areas of the players are designed in a way that important areas on the field are covered by at least one player and ensuring that the players (which don't recognize their teammates anymore) don't disturb each other.

Figure 12 shows a screenshot of the global view during a game. While the active player dribbles the ball around an opponent the supporting player moves to its preferred position and the strategic player observes the ball.
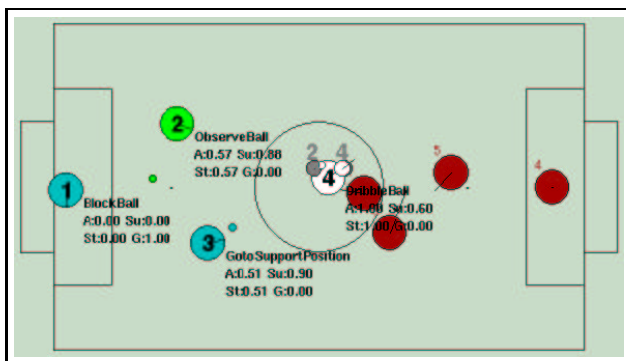

Fig. 12.   Visualization of the results of the global sensor integration together with a player's utilities for taking a certain role, its current role and its current action. The small white circle denotes the position of the ball as determined by the global sensor integration, whereas the small grey ones correspond to the ball as perceived by the individual players.

C. Results

In order to verify the effectiveness of the team coordination approach, log-files of games the *CS Freiburg* team played at RoboCup 2001 were evaluated.

Table I displays statistics evaluating the role assignment method. All values are averaged over the games played at RoboCup 2001 in Seattle. In the first line the length of the time intervals our players kept a certain role are listed. Interestingly the values for the field players are similar to the average between role switches of 7 seconds reported by the *ART* team [7]. The second line shows how long (on average) a role was not unique when there was a role change. This time is different from zero because of two reasons. Firstly, the communication of the robots is not synchronized, so one robot might already take on a role even if the other one will only give it up the next time it sends an information package. Since the cycle time of our robots is 100 msec, one would expect an average delay of 50 msec at this point. Secondly, there is the problem that there are glitches in the wireless communication, leading to the loss of some information packages, which means that a role change is not recorded immediately, which explains the remaining 25–55 msec.

Role	Active	Support	Strategic
Role kept	5.4 s	8.1 s	5.7 s
Role not unique (per change)	0.106 s	0.075 s	0.076 s

TABLE I

EVALUATION OF THE ROLE ASSIGNMENT METHOD

V. BASIC SKILLS AND ACTION SELECTION

In order to play an effective and successful game of robotic soccer, a powerful set of basic skills is needed.

A. Goalkeeper

As in most other robot soccer teams in the middle size league the hardware configuration of the goalkeeper differs from the one of the field players which is mainly because of the different tasks the goalkeeper and field players are designed for. The *CS Freiburg* goalkeeper has a special hardware setup where the top part of the robot, containing laser range finder and vision camera, is mounted 90° to one side allowing the robot to move quickly parallel to the goal line (see Figure 13). This kind of setup is quite popular in the middle size league and used by other teams, too.

The goalkeeper uses six skills for defending its goal as sketched in Figure 13. If the goalkeeper doesn't know where the ball is, it alternately moves a little bit closer towards the goal center and rotates left and right searching for the ball. When the ball is moving slowly the goalkeeper blocks in order to minimize the area of attack. However, if the ball is moving fast, the goalkeeper intercepts at the point where the ball is expected to pass the goal line. In practice a "smooth" behavior is achieved by interpolating between the block and intercept position considering the ball's velocity. If the ball is on the side of the goal, the robot turns towards the corners to maintain the ball in its field of view and to make it more difficult for an opponent to score a
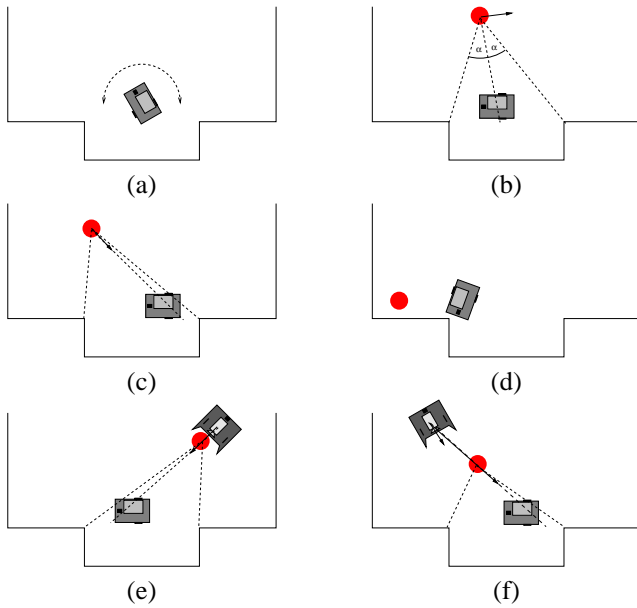
Fig. 13. Goalkeeper's tactics for saving the *CS Freiburg*'s goal: (a) ball searching, (b) minimizing the area of attack, (c) intercepting the ball, (d) turning to corner, (e) interception using opponent heading, and (f) interception using opponent to ball heading.

direct goal. The last two and more sophisticated tactics for intercepting the ball are based on the heading of an opponent owning the ball (Figure 13(e)), or the heading of an opponent to the ball (Figure 13(f)). They assume that the attacking robot will kick the ball straight ahead which is true for most robot teams participating in the middle size league but is not true for a few teams (e.g., *Golem* or *Alpha++*). For this reason these two tactics can be turned on or off at game start.

B. Basic Skills of Field Players

To get hold of the ball a player moves to a position behind the ball following a collision-free trajectory generated by the path planning system which constantly (re)plans paths based on the player's perception of the world (*GoToBall*). If close to the ball a player approaches the ball in a reactive manner to get it precisely between the flippers while still avoiding obstacles (*GetBall*). Once in ball possession, a player turns and moves the ball carefully until facing in a direction which allows for an attack (*TurnBall*). If the player is right in front of the opponent goal, it kicks the ball in a direction where no obstacles block the direct way to the goal (*ShootGoal*). Otherwise it first heads towards a clear area in the goal and turns sharply just before kicking in case the opponent goalkeeper moved in its way (*MoveShootFeint*). However, if obstacles are in the way to the goal, the player tries to dribble around them (*DribbleBall*) unless there is not enough room. In this case the ball is kicked to a position close to the opponent goal (*ShootToPos*). In the event of being too close to an opponent or to the field border the ball is propelled away by turning quickly in an appropriate direction (*TurnAwayBall*). If a player gets stuck close to an obstacle it tries to free itself by first moving away slowly and (if this doesn't help) then trying random moves (*FreeFromStall*). However a player doesn't give way if the ball is stuck between

himself and an opponent to avoid being pushed with the ball towards his own goal (*WaitAndBlock*).

Against fast playing teams the *CS Freiburg* players are often outperformed in the race for the ball when approaching it carefully. Therefore two variants of a skill for situations in which speed is crucial were developed. Both let the robot rush to the ball and hit it forwards while still avoiding obstacles. In offensive play *BumpShootOffense* is employed to hit the ball into the opponents goal when very close to it. In defensive play the use of *BumpShootDefense* can be switched on or off according to the strength of the opponent.

Players fulfilling strategic tasks compute their positions and follow collision-free paths to dynamically determined positions (*GoToPos*). From these positions the players either search the ball if not visible (*SearchBall*) by rotating constantly or observe it by turning until facing it (*ObserveBall*). In offensive play a supporting player may also take a position from where it should be able to score a goal directly (*WaitForPass*). Once in such a position, it signals to its teammates that it is waiting to get the ball passed. The decision is then up to the ball owning player whether to pass the ball (*PassBall*) or to try to score a goal by itself.

To comply with the "10-seconds rule"[5] a player keeps track of the time it is spending in a penalty area. Whenever it come close to violating the 10-seconds rule, it leaves the area following a collision-free path generated by the same path planning system as employed in the GoToBall skill (*LeavePenaltyArea*).

The *CS Freiburg* players are capable of effectively dribbling with the ball around obstacles and exploiting deliberately the possibility of rebound shots using the walls[6]. In the following these two skills are described in more detail.

Figure 14(a) shows a screenshot of a player's local view while dribbling. In every cycle, potential continuations of the current play are considered. Such continuations are lines to points closer to the opponent's goal within a certain angle range around the robot's heading.
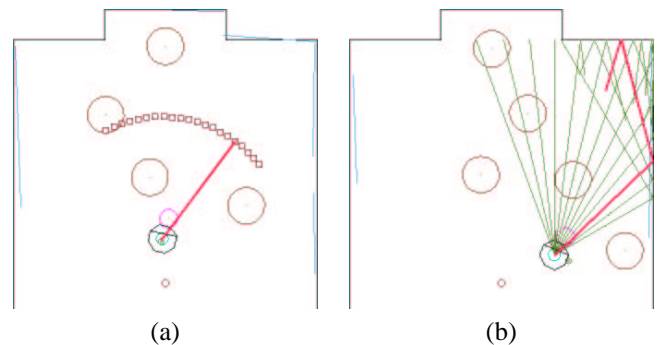

Fig. 14. A *CS Freiburg* player's view of the world while (a) dribbling and (b) ball-shooting. Circles denote other robots and the small circle in front of the player corresponds to the ball. Lines almost parallel to the field borders are perceived by the laser range finder. The other lines leading away from the player are evaluated by the skills.

All the possible lines are evaluated and the direction of the

---

[5] A player is allowed to enter a goal area for no more than 10 seconds. Visit www.robocup.org for a complete description of the rules.

[6] Of course, rebound shots only make sense when games are played according to the "old rules" which envision walls as field borders

best line sample is taken as the new desired heading of the robot. A line is evaluated by assigning a value to it, which is the higher the further it is away from objects, the less turning is necessary for the player and the closer its heading is to the opponent's goal center. Determining the robots heading this way and adjusting the wheel velocities appropriately in every cycle lets the robot smoothly and safely dribble around obstacles without loosing the ball. The *CS Freiburg* team scored some beautiful goals after a player had dribbled the ball over the field around opponents along an S-like trajectory. Of course, all this only works because the ball steering mechanism allows for a tight ball control.

Figure 14(b) shows a screenshot of a player during ball-shooting. For this skill the lines are reflected at the walls and are evaluated to find the best direction where to kick the ball. A line's value is the higher the further away from obstacles it is, the closer its endpoint is to the opponent's goal and the less turning is required for the player to face in the same direction. Taking into account that the ball doesn't rebound at the field borders in a perfect billiard-like manner the correlation between the angles of reflection is calibrated manually. Using the shooting skill the players were able to play the ball effectively to favorable positions and even to score goals directly.

C. Action Selection for Field Players

The *CS Freiburg*'s action selection module is based on extended behavior networks developed by Dorer [11]. They are a revised and extended version of the behavior networks introduced by Maes [23] and can be viewed as a particular form of decision-theoretic planning. The main structural element in extended behavior networks is the *competence module*, which consists of preconditions, effects, and a certain behavior that has to be executed. *Goals* can be explicitly specified and can have a situation-dependent relevance, reflecting the agent's current motivations. The state of the environment, as it is perceived by the agent, is described via a number of continuously valued propositions $p_i \in [0..1]$. Competence modules are connected with goals if they are able to influence goal conditions and also with each other, if a competence module has an effect that is a precondition of another. Along the resulting network connections an *activation spreading mechanism* is defined, with goals being the source of activation. An action is selected by considering each competence module's executability and received activation.

Figure 15 shows a part of the extended behavior network [25] used for the *CS Freiburg* players. The ellipses represent the competence modules with their preconditions below and their effects on top of them. The players have two goals: Shoot a *soccer goal* or *cooperate* with teammates. The relevance condition *role_active* (player has *active* role) ensures that only one of these goals is relevant at a time depending on the player's current role. The strength of the effect connections (indicated by the numbers next to arrows) are set manually and are related to the probability of success.

D. Learning Basic Skills and Action Selection

One difficulty in robotics (and in particular in robotic soccer) is to adapt skills and the overall behavior to changes in the environment. An upgrade of the robot's kicking device, for example, influences the overall performance drastically and parameters of
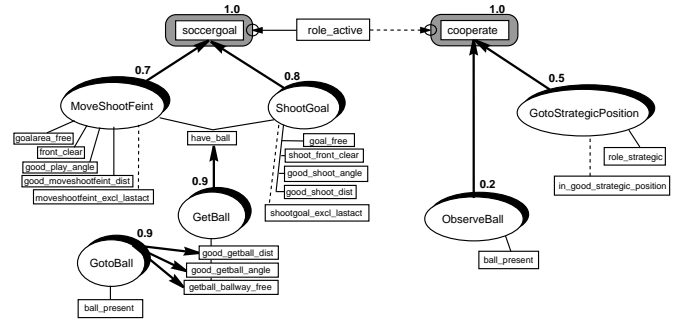

Fig. 15. A part of *CS Freiburg*'s extended behavior network.

the skills and the action selection mechanism have to be revised. In order to address this issue, learning methods for learning unsupervised and online have been evaluated. The ultimate goal in this context is to build robotic soccer agents that improve their skills during their whole life as efficiently and quickly as possible [20].

We applied hierarchical *Reinforcement Learning (RL)* based on Semi Markov Decision Processes (SMDPs) as proposed by Bradtke and Duff [5]. In contrast to Markov Decision Processes (MDPs), which are defined for an action execution at discrete time steps, SMDPs are providing a basis for learning to choose among *temporally abstract actions*. Temporally abstract actions are considered in standard SMDPs as *black box skills*, which execute a sequence of actions in a defined partition of the state space for an arbitrary amount of time.

In contrast to other implementations based on SMDPs the described implementation has the capability of learning simultaneously and online on all levels of the hierarchy, similar as Dietterich's MAXQ method [10]. Hence skills were modelled as MDPs with rewards defined according to the specific task.

The learner's state space consists of position and velocity (x_b, y_b, v_b) of the ball, the pose and velocity $(x_r^k, y_r^k, \theta_r^k, v_r^k)$, with $0 < k < n$, of other robots in the field of view, and pose and velocity $(x_r, y_r, \theta_r, v_r)$ of the robot itself. The learner's action space is given by the set values for translational and rotational velocity (v_t, v_r) and a binary value for triggering the kicking device. Since the state space, though based on features, is still large, the method of *Tile Coding* has been utilized to gain the effect of generalization [31].

Depending on the skill's natural goal, terminal states have been defined that indicate a successful execution by a reward of 100 and a non-successful execution by a reward of -100. Furthermore, in order to foster near optimal and thus fast solutions, a negative reward of -1 was emitted for each selected action.

The SMDP and the MDPs have been learned with $Q(\lambda)$ and Sarsa(λ), respectively. At this, Q-learning is used with *Eligibility Traces* in order to improve the learning process. Experiences are taken for an update of the whole trace rather than for the last transition only. As the execution of a skill usually envolves many perception-action cycles, *Eligibility Traces* accelerate the distribution of information within the value function. [7]

---

[7] The influence of experiences on former states is set by the parameter $\lambda = 0.8$ (a common value when learning with n-step updates). The other learning parameters have been choosen as follows: Learning rate $\alpha = 0.1$ (small, due
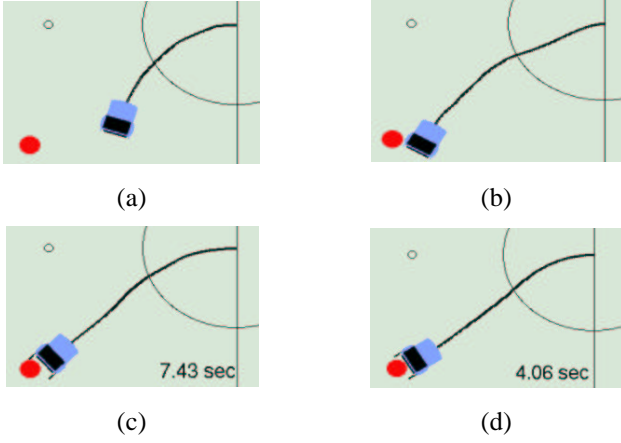
Fig. 16. Learning the skill *ApproachBall* after (a) 10, (b) 100, (c) 500 and (d) 15000 episodes. After 500 episodes the goal of the task could continuously be achieved. The time indicates the average duration of one episode.

	Active	Strategic	Support
BumpShootDefense	7	0	0
BumpShootOffense	1	0	0
DribbleBall	4	0	0
FreeFromStall	3	0	0
GetBall	32	0	0
GotoBall	32	0	0
GoToPos	0	16	40
LeavePenaltyArea	1	0	0
MoveShootFeint	1	0	0
ObserveBall	0	75	50
SearchBall	5	9	10
ShootGoal	3	0	0
ShootToPos	2	0	0
TurnBall	2	0	0
TurnAwayBall	4	0	0
WaitAndBlock	3	0	0

TABLE II

TIME IN PERCENT A SKILL WAS PERFORMED IN THE FINAL GAMES AT ROBOCUP 2000 BROKEN DOWN FOR THE DIFFERENT ROLES

The training process has been carried out in a straightforward manner, similar to the way humans would train soccer. Firstly, basic skills, as described in section V-B,[8] have been trained in simple, static scenarios. Secondly, the appropriate selection of these skills and their embedding into the task has been trained in a realistic scenario which was a game against a hand-crafted *CS Freiburg* player. Due to the complexity of the task it turned out to be necessary to develop a simulation for a faster pre-training of the robots. Figure 16 gives an impression of the evolution of the skill *ApproachBall* during learning in the simulation.

E. Results

Table II shows statistics of how often a skill was performed in the final games at RoboCup 2000 broken down for the different roles. It seems quite surprising that the players were searching for the ball up to 10 % of the total playing time. However analysis of the log-files showed that the high numbers can be either attributed to communication problems or to one of the few situations where none of the players was seeing the ball. The supporting and strategic player spent most of their time observing the ball. This was intended since moving more often or more accurately to a target positions would result in a very nervous and less effective behavior.

At a first glance it seems surprising that the active player was most of the time (64 %) occupied with getting hold of the ball. However the fact that after kicking the ball the active player usually starts to follow the ball again explains the high numbers for *GoTo* and *GetBall*. Nevertheless it made use of all the skills available to it demonstrating that it distinguished between a large number of different game situations.

The learning of skills and their selection has been evaluated with respect to the systems capability of adaption to the task. Figure 17 documents the rewards received during learning by the robot with the reduced kicking device (normal kicker) and

Fig. 17. Learning to improve pre-learned skills and their selection while competing with a hand-crafted player. If equipped with a different kicking device (normal kicker), the robot adapts its strateg.

the robot with the kicking device used for pre-learning (strong kicker). The baseline indicates the average of the accumulated rewards a CS Freiburg Player with normal velocity achieves during one episode. Due to the fact that skills were pre-trained with a strong kicker the learner using the normal kicker reaches less rewards during the first 1000 episodes. After 6000 episodes, however, playing with a normal kicker turns out to be more successful than playing with the strong one. The learner with the normal kicker develops a different way of playing: He is dribbling more often to the front of the goal and performs a rather precise shoot from small distance.

Finally the trained system has been run on a real robot. While playing for one hour the learner was able to score 0.75 goals per minute. In contrast, the hand-crafted CS Freiburg player scores 0.94 goals per minute. Note, that compared to the hand-crafted player far less time was needed for design and parameterization.

---

to the non-determinism of the environment), exploration rate $\epsilon = 0.05$ (small, since high exploration could lead to failures) and discounting parameter $\gamma = 1.0$ (due to the presence of an absorbing state).

[8]Note, that the skills are learned from scratch without *a priori* knowledge expect the rewards.

VI. CONCLUSION AND DISCUSSION

The combination of methods described above as well as the implementation of the methods themselves led to a successful robotic soccer team, as demonstrated by the performance of *CS Freiburg* at the RoboCup competitions. Nevertheless, there are two important questions. Firstly, there is the question whether there are alternative successful designs for robotic soccer teams. Secondly, there is the question in how far RoboCup can influence the multi-robot and/or multi-agent research areas.

A. Designs for Robotic Soccer Teams

Reconsidering the history of the last five years of RoboCup, there are, of course, other quite successful robotic soccer team designs. Our team represents just one point in the space of possible designs. It can be characterized by the following salient properties:

1. accurate and reliable self-localization
2. a large set of basic skills
3. deliberation and reactive planning
4. group coordination for sensing and acting
5. restricted perception (only to the front)
6. restricted mobility (two differential drives and one caster)

(1)–(4) are properties that definitely help to win games. In particular, without self-localization it can often happen that a team scores own goals (and the history of RoboCup is full of examples of that). Furthermore, accurate and reliable self-localization allows the *CS Freiburg* players to go to their kick-off positions automatically (while other teams place their robots manually). In addition, accurate self-localization is needed when one wants to comply with the 10-seconds rule (see Section V-B). However, an uncertain but good enough pose estimation might often be enough. As a matter of fact, in the two final games at RoboCup 2000 and 2001, the other teams did not use accurate self-localization, but were nevertheless very strong.

At RoboCup 2000, the *Golem* team demonstrated that robots with omni-directional vision and an omni-directional drive can be very agile and reactive, enabling them to be faster at the ball than the *CS Freiburg* players. The strength of the *Golem* team can, for instance, be judged from Figure 18, which shows the traces of the positions of the ball and the *CS Freiburg* players as recorded in the quarter final game and the final game.

While in the quarter final against *CMU Hammerheads* the *CS Freiburg* players spent most of the time in the opponent's half, they were forced into their own half most of the time during the final game against *Golem*. The game was finally won by *CS Freiburg* by penalty kicks. All in all, it was the overall impression that the mechanical design and the sensor setup of the *Golem* team was superior, but that *CS Freiburg* was able to compensate for this by accurate self-localization, team play, and a good goalie.

Similarly, during the final game at RoboCup 2001, the *Osaka Trackies* put *CS Freiburg* under a lot of pressure. They also used an omni-directional camera, but no omni-directional drive. Nevertheless, the platform had high agility and the players of *Osaka Trackies* were much faster at the ball than the *CS Freiburg* players. This game was decided because one *Osaka* robot was removed from the field because of a red card, one was removed
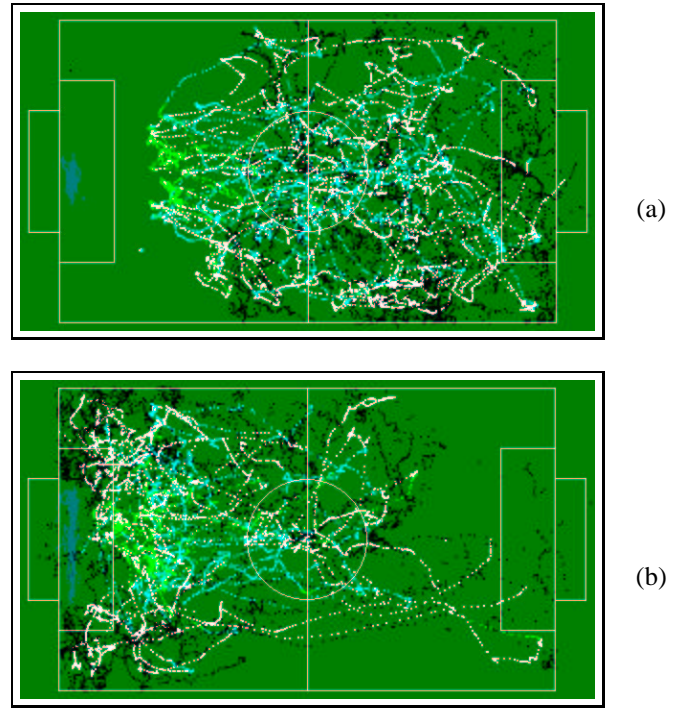
Fig. 18. Trace of the positions of the ball (black) and the *CS Freiburg*'s player's (light color) at RoboCup 2000 during (a) the quarter final and (b) the final – both times playing from left to right.

because of a hardware problem and the goalie had a severe problem as well. In this situation, *CS Freiburg* scored the only goal during the game. From this game one got the impression that *CS Freiburg* was able to compensate the speed of the other team by a very reliable goalie and robustness of the hard- and software.

Summarizing the brief comparison with designs of other strong teams, it is clear that the *CS Freiburg* design is quite competitive, but there are alternative designs that are as competitive as *CS Freiburg*.

B. The Relevance for Multi-Robot- and Multi-Agent-Systems

As should have become obvious, robotic soccer is a rich source of inspiration for multi-robot and multi-agent research. It has already led to the development of interesting methods, e.g., our fast scan-based localization method in polygonal environments [17], our global object tracking technique described in Section III-B, the SPAR method for deciding the placement of players on the field [30], and the dynamic role assignment technique developed by Castelpietra *et al.* [7]. And this list is by far not complete. Although some of these methods are specific to robotic soccer, they may nevertheless serve as inspiration for similar problems in other multi-robot applications.

Furthermore, RoboCup is an attractive testbed for comparing different methods under "hostile" conditions. The most interesting aspect of the RoboCup, however, is the need to design and implement systems that "close the loop" from observation over action planing and action selection to action execution and monitoring. It is not enough to come up with methods that work in isolation under ideal conditions, but one has to use methods that can be integrated into a large system and that show graceful degradation when something goes wrong.

Of course, the RoboCup environment has a number of properties which may limit transferability to other multi-robot applications. For example, on a robotic soccer field one can assume that communication radius is not a limiting factor in establishing successful communication links between the agents, while in larger scale applications one might be very well forced to consider local, temporary, ad-hoc communcation links [18]. Also, the robots can almost always observe the rest of the group, which may not be true when operating in larger environments. However, the necessity to be responsive to dynamic changes, to substitute roles of broken robots and to be robust in general, are properties found in almost all multi-robot applications.

ACKNOWLEDGMENTS

REFERENCES

[1] M. Asada and H. Kitano, editors. *RoboCup-98: Robot Soccer World Cup II*. Lecture Notes in Artificial Intelligence. Springer-Verlag, 1999.

[2] Y. Bar-Shalom and T.E. Fortmann. *Tracking and Data Association*. Academic Press, 1988.

[3] M. Bennewitz, W. Burgard, and S. Thrun. Exploiting constraints during prioritized path planning for teams of mobile robots. In *Proc. Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 393–398, Maui, Hawaii, 2001.

[4] A. Birk, S. Coradeschi, and S. Tadokoro, editors. *RoboCup-2001: Robot Soccer World Cup V*. Lecture Notes in Artificial Intelligence. Springer-Verlag, Berlin, Heidelberg, New York, 2002.

[5] S. J. Bradtke and M. O. Duff. Reinforcement learning methods for continuous-time Markov decision problems. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 393–400. The MIT Press, 1995.

[6] J. Bruce, T. Balch, and M. Veloso. Fast and inexpensive color image segmentation for interactive robots. In *Proc. Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 2061 – 2066, Takamatsu, Japan, 2000.

[7] C. Castelpietra, L. Iocchi, M. Piaggio, A. Scalzo, and A. Sgorbissa. Communication and coordination among heterogeneous mid-size players: ART99. In Stone et al. [28], pages 86–95.

[8] I.J. Cox. A review of statistical data association techniques for motion correspondence. *Int. Journal of Computer Vision*, 10(1):53–66, 1993.

[9] M. Dietl, J.S. Gutmann, and B. Nebel. Cooperative sensing in dynamic environments. In *Proc. Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 1706 – 1713, Maui, Hawaii, 2001.

[10] T. G. Diettrich. The MAXQ method for hierarchical reinforcement learning. In *Proc. 15th Conf. on Machine Learning (ICML)*, pages 118 – 126. Morgan Kaufmann, 1998.

[11] K. Dorer. Behavior networks for continuous domains using situation-dependent motivations. In *Proc. 16th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 1233–1238, Stockholm, Sweden, 1999.

[12] B. Dümler. Kooperative Pfadplanung (in German). Diplomarbeit, Albert-Ludwigs-Universität Universität Freiburg, Institut für Informatik, 2001.

[13] B. Fischer. Robuste Positionsschätzung mittels Monte-Carlo-Lokalisierung in der RoboCup-Umgebung (in German). Diplomarbeit, Albert-Ludwigs-Universität Universität Freiburg, Institut für Informatik, 2002.

[14] J.S. Gutmann, W. Burgard, D. Fox, and K. Konolige. An experimental comparison of localization methods. In *Proc. Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 736 – 743, Victoria, Canada, October 1998.

[15] J.S. Gutmann, W. Hatzack, I. Herrmann, B. Nebel, F. Rittinger, A. Topor, and T. Weigel. The CS Freiburg team: Playing robotic soccer on an explicit world model. *AI Magazine*, 21(1):37–46, 2000.

[16] J.S. Gutmann, W. Hatzack, I. Herrmann, B. Nebel, F. Rittinger, A. Topor, T. Weigel, and B. Welsch. The CS Freiburg robotic soccer team: Reliable self-localization, multirobot sensor integration, and basic soccer skills. In Asada and Kitano [1], pages 93–108.

[17] 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, 2001.

[18] M. Jäger and B. Nebel. Decentralized collision avoidance, deadlock detection, and deadlock resolution for multiple mobile robots. In *Proc. Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 1213 – 1219, Maui, Hawaii, 2001.

[19] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, E. Osawa, and H. Matsubara. RoboCup: A challenge problem for AI. *AI Magazine*, 18(1):73–85, 1997.

[20] A. Kleiner, M. Dietl, and B. Nebel. Towards a life-long learning soccer agent. In *Proc. Int. RoboCup Symposium '02*, pages 119 – 127. Fukuoka, Japan, 2002.

[21] S. Leroy, J.P. Laumond, and T. Simeon. Multiple path coordination for mobile robots: A geometric algorithm. In *Proc. 16th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 1118 – 1123, Stockholm, Sweden, 1999.

[22] A. K. Mackworth. On seeing robots. In A. Basu and X. Li, editors, *Computer Vision: Systems, Theory, and Applications*, pages 1–13. World Scientific Press, Singapore, 1993.

[23] P. Maes. Situated agents can have goals. In Pattie Maes, editor, *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, pages 49–70. MIT Press, Cambridge, MA, 1990.

[24] P. S. Maybeck. The Kalman filter: An introduction to concepts. In I.J. Cox and G.T. Wilfong, editors, *Autonomous Robot Vehicles*, pages 194–204. Springer-Verlag, 1990.

[25] K. Müller. Roboterfußball: Multiagentensystem CS Freiburg (in German). Diplomarbeit, Albert-Ludwigs-Universität Freiburg, Institut für Informatik, 2000.

[26] B. Nebel, J.S. Gutmann, and W. Hatzack. The CS Freiburg '99 team. In Veloso et al. [37], pages 703–706.

[27] D. Schulz, W. Burgard, D. Fox, and A.B. Cremers. Tracking multiple moving targets with a mobile robot using particle filters and statistical data association. In *Int. Conf. on Robotics and Automation (ICRA)*, pages 1665 – 1670, Seoul, Korea, 2001.

[28] P. Stone, G. Kraetzschmar, T. Balch, and H. Kitano, editors. *RoboCup-2000: Robot Soccer World Cup IV*. Lecture Notes in Artificial Intelligence. Springer-Verlag, Berlin, Heidelberg, New York, 2001.

[29] P. Stone and M. Veloso. Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence*, 110(2):241 – 273, jun 1999.

[30] P. Stone, M. Veloso, and P. Riley. CMUnited-98: Robocup-98 simulator world champion team. In Asada and Kitano [1], pages 61–76.

[31] Richard S. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 1038–1044. The MIT Press, 1996.

[32] M. Thiel. Zuverlässige Ballerkennung und Positionsschätzung (in German). Diplomarbeit, Albert-Ludwigs-Universität Freiburg, Institut für Informatik, 1999.

[33] S. Thrun, A. Bücken, W. Burgard, D. Fox, T. Fröhlinghaus, D. Hennig, T. Hofmann, M. Krell, and T. Schimdt. Map learning and high-speed navigation in RHINO. In D. Kortenkamp, R.P. Bonasso, and R. Murphy, editors, *AI-based Mobile Robots: Case studies of successful robot systems*. MIT Press, Cambridge, MA, 1998.

[34] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust monte carlo localization for mobile robots. *Artificial Intelligence*, 128(1-2):99–141, 2001.

[35] A. Topor. Pfadplanung in dynamischen Umgebungen (in German). Diplomarbeit, Albert-Ludwigs-Universität Freiburg, Institut für Informatik, 1999.

[36] M. Veloso, M. Bowling, S. Achim, K. Han, and P. Stone. The CMUnited-98 champion small-robot team. In Asada and Kitano [1].

[37] M. Veloso, E. Pagello, and H. Kitano, editors. *RoboCup-99: Robot Soccer World Cup III*. Lecture Notes in Artificial Intelligence. Springer-Verlag, Berlin, Heidelberg, New York, 2000.

[38] T. Weigel. Roboter-Fußball: Selbstlokalisierung, Weltmodellierung, Pfadplanung und verhaltensbasierte Kontrolle (in German). Diplomarbeit, Albert-Ludwigs-Universität Freiburg, Institut für Informatik, 1999.

[39] T. Weigel, W. Auerbach, M. Dietl, B. Dümler, J.S. Gutmann, K. Marko, K. Müller, B. Nebel, B. Szerbakowski, and M. Thiel. CS Freiburg: Doing the right thing in a group. In Stone et al. [28].

[40] T. Weigel, A. Kleiner, F. Diesch, M. Dietl, J.S. Gutmann, B. Nebel, P. Stiegeler, and B. Szerbakowski. CS Freiburg 2001. In Birk et al. [4].