CS Freiburg 2001*

Thilo Weigel[1], Alexander Kleiner[1], Florian Diesch[1], Markus Dietl[1],
Jens-Steffen Gutmann[2], Bernhard Nebel[1], Patrick Stiegeler[1], and Boris Szerbakowski[3]

[1] Institut für Informatik, Universität Freiburg
79110 Freiburg, Germany
{last-name}@informatik.uni-freiburg.de
[2] Digital Creatures Laboratory, Sony Corporation
141-0001 Tokyo, Japan
gutmann@ieee.org
[3] Optik Elektronik, SICK AG
79183 Waldkirch, Germany
boris.szerbakowski@sick.de

Abstract. The CS Freiburg team has become F2000 champion the third time in the history of RoboCup. The success of our team can probably be attributed to its robust sensor interpretation and its team play. In this paper, we will focus on new developments in our vision system, in our path planner, and in the cooperation component.

1 Introduction

Although the general setup, the hardware and software architecture of CS Freiburg at the RoboCup 2000 competition [16] turned out to be quite satisfactory, there appeared to be room for improvements. First of all, the hardware was not as reliable as it used to be when the robots were bought in 1997. Secondly, there were quite a number of software components in the system that could be improved.

For this reason, we invested in new hardware, which led to a significant improvement in robustness. In fact, the final game against the *Osaka Trackies* was partially won because our hardware was very reliable.

On the software side, a number of improvements were made on the system level, such as integrating a new camera system. Furthermore, we worked on components that we considered as critical, such as the vision system, the path planning module, and the module for cooperative play. The improvements of these components, which are described below, did indeed give us the competitive edge. However, it also became clear that the reactiveness by the *Osaka Trackies* is hard to match with our team.

* This work has been partially supported by *Deutsche Forschungsgemeinschaft* (DFG), by *Medien- und Filmgesellschaft Baden-Württemberg mbH* (MFG), by *SICK AG* and *Sony Corporation*

A. Birk, S. Coradeschi, and S. Tadokoro (Eds.): RoboCup 2001, LNAI 2377, pp. 26–38, 2002.

2 Hardware

When we first participated at RoboCup in 1998 we used *Pioneer-1 robots* as manufactured by *ActivMedia*. To recognize the ball ball we employed the *Cognachrome* vision system manufactured by *Newton Labs* which was available installed in the Pioneer-1 robots. Our effort in hardware development was limited to mounting a *SICK laser range finder* for self localization and object recognition and building a simple kicking device with parts from the Märklin Metallbaukasten. For local information processing the robots were equipped with *Libretto 70CT* notebooks and they used the *Wavelan radio ethernet* from *Lucent Technologies* to communicate with each other [6].

Even though we maintained the same general hardware setup, we virtually replaced every part of this setup in the past four years. In 1999 we replaced the laser range finders with the *SICK LMS200* model. They provide depth information for a $180°$ field of view with an angular resolution of $0.5°$ and an accuracy of 1 cm [9]. For RoboCup 2000 we started to make major mechanical and electronical modifications to our robots. We installed nickel-cadmium batteries because of their light weight and high speed charging capability. To improve the ball handling and shooting capability of our robots *SICK AG* assisted us in building a new kicking device with movable ball steering flippers, which can be turned to an upright position and back. The kicking device is strained by a wind-screen wiper motor and released by a solenoid [16].

Fig. 1. A CS Freiburg Player

This year we made further hardware modifications in order to increase the performance of our team. Figure 1 shows one of our robots as it competed this year. We installed Pioneer-2 boards, which now allow the robots to move considerably faster with the same *Pittman* motors that we have been using during the past years. For more precise movements we substituted the rear caster wheel by a caster roller. To be able to develop our own vision software we replaced the old vision system by a *Sony DFW-V500 digital camera* and switched to *Sony Vaio PCG-C1VE notebooks* because of their IEEE 1394 interface. We also upgraded the *WaveLan* cards to the new 11 Mbit/s (2.4 GHz) cards. To get our laser range finders to work via the USB port we had to find a circuit diagram for a RS422-USB converter board, capable of a 500Mbaud rate[1], which *SICK AG* then manufactured for us. For an artificial intelligence research group hardware development and maintenance isn't a trivial task at all and would certainly not have been possible without the help of SICK AG. Even only adapting our software to the new hardware consumed a lot of our preparation time for this year's tournament.

[1] http://www.ftdichip.com/Files/usb-422.zip

3 Vision

The implementation of our new vision system consists of two major software parts, one for region segmentation by color and one for the mapping from image coordinates to positions of objects on the field. The region segmentation by color is carried out on 320x240 Pixels in YUV color space using the CMVision library.[2] Due to the fast algorithm [3], our system is capable to work with even two cameras at a rate of 30Hz. From the segmented regions the world coordinates are computed. For this purpose we implemented two alternative solutions, described below.

3.1 Tsai Camera Calibration

Tsai's camera model [15] is based on the pinhole model of perspective projection, and consists of six extrinsic and five intrinsic parameters. The extrinsic parameters describe the translation (T_x, T_y, T_z) and rotation (R_x, R_y, R_z) from the world coordinate frame **W** to the camera coordinate frame **C**. The intrinsic parameters include the effective focal length f, the radial lens distortion κ_1, a scale factor s_x and the image center (C_x, C_y), also known as principal point. Generally, the objective of calibration is to determine optimal values for these parameters from a set of known points in the world coordinate frame (x_w, y_w, z_w) and their corresponding pixels in the sensor plane (x_u, x_v). Once the intrinsic parameters are determined, they can be used for different positions and orientations of the camera. The extrinsic parameters, however, have to be re-calibrated when the camera moves relatively to the world coordinate frame origin.

With the calibrated parameters, one can predict the pixel (x_u, x_v) in the sensor plane from a given point (x_w, y_w, z_w) in **W** by three transformations:

1. **Rigid body transformation** Transformation from the world coordinate frame to the camera coordinate frame: $\begin{bmatrix} x_c\ y_c\ z_c \end{bmatrix}^T = R \begin{bmatrix} x_w\ y_w\ z_w \end{bmatrix}^T + \begin{bmatrix} T_x\ T_y\ T_z \end{bmatrix}^T$, where $R = Rot(R_x)Rot(R_y)Rot(R_z)$
2. **Perspective projection** Due to the pinhole model, the undistorted pixel coordinates can be calculated by the theorem of intersecting lines: $x_u = f\frac{x_c}{z_c}, x_v = f\frac{y_c}{z_c}$
3. **Distortion transformation** Transformation from undistorted pixels to distorted ones with the distortion factor κ_1: $x_u = x_d(1 + \kappa_1\rho^2), y_u = y_d(1 + \kappa_1\rho^2)$, where $\rho = \sqrt{x_d^2 + y_d^2}$

Since the ball is usually located on the field plane, a coplanar calibration was used. As mentioned above, the calibration process can be divided into two parts, which are intrinsic and extrinsic parameter calibration. For the intrinsic parameter calibration a small dot matrix was placed on different positions in front of the camera. Data for extrinsic parameter calibration was generated from a large dot matrix, which was placed on the soccer field plane in front of the robot.

Another possibility to generate calibration data is to take snapshots of the ball on the field directly. This comes with the advantage that the error from calculating the blob center on the image is reflected in the data. However, it turned out that the data was too noisy for the Tsai optimization procedure, which was indicated by a high calibration error.

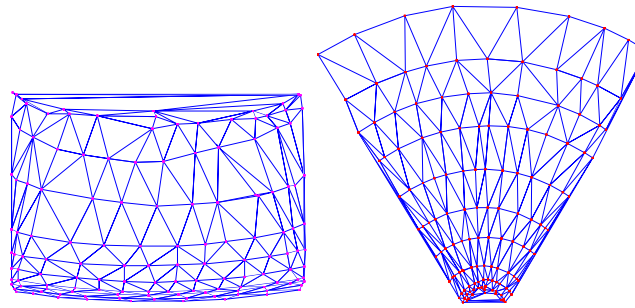
[2] http://parrotfish.coral.cs.cmu.edu/cmvision/

3.2 Interpolation

Besides the analytical Tsai method, we also used a method from the last years team for linear interpolation [13]. The method interpolates the position of the ball based on triangles generated from *a priori* collected world to image correspondences. The mapping takes place by identifying the triangle which encloses the considered pixel. This triangle can be mapped to a triangle on the field which then is used to interpolate the objects position.

For a uniform distribution of the triangles the *Delaunay* triangulation has been used. Figure 2 shows the generated triangles on the image and the field respectively. The dense occurrence of samples at the bottom of both pictures indicates that more samples have been collected from positions in the vicinity of the robot than from distant positions.

The collection of adequate samples has been carried out by assistance of our accurate self-localization. The ball has been placed on a fixed position on the field, which has been taken as reference. Subsequently, snapshots of the ball from different robot position have been taken and converted to samples in the camera coordinate frame. Finally, a triangulation algorithm produced a list of triangles, such as shown in figure 2.



(a) (b)

Fig. 2. Distribution of triangles on the camera image (a) and on the field (b) after the *Delaunay* triangulation

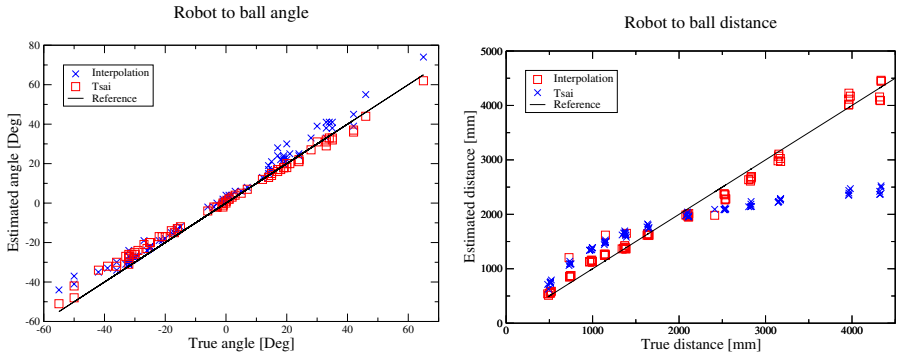
3.3 Evaluation

Figure 3 shows the measured accuracy in estimating the ball distance and ball angle by the two approaches. While both approaches perform well in estimating the direction of the ball, the Tsai method seems to be inaccurate in estimating larger distances. We assume that this is partially caused by a poor coverage of the field by calibration dots during the calibration of the extrinsic parameters. A larger matrix would probably improve the quality of the calibration. However, this is not always practicable during RoboCup competitions.

In contrast, data collection for the interpolation is handled easier. The robot is programmed to take samples at particular positions autonomously. Furthermore, the quality of the collected data can be evaluated directly by considering the triangle distribution

as shown in figure 2. Also the effect of noisy measurements is not as critical as it is for the analytical method, which uses non-linear optimization to estimate the model parameters.

For most demands in the RoboCup domain, however, the resulted accuracy of both methods suffice, since in most cases the direction of the ball is more important than an accurate distance estimation beyond three meters.



(a) (b)

Fig. 3. Accuracy in estimating the direction (a) and distance (b) of the ball by interpolation and the Tsai method

4 Path-Planning

The robotic players of our team are equipped with a rich set of basic soccer skills such as *GoToBall*, *GetBall*, *MoveShootFeint*, *ShootGoal*, etc. Details about these skills can be found in our last year's team report [16]. Some of these skills, e.g. *GoToBall* which enable a player to move around obstacles close to the ball, require the planning of a path from the robot's current position to a target position. In addition, more team oriented skills like moving to a strategic position on the field, or the automatic positioning procedure before game start need the ability to find collision-free paths, too.

A prerequisite for planning paths is the existence of a model of the world. Although, a crude world model can already be used for generating paths, the integrity and accuracy of the world model has a big influence on the quality of the computed paths. Fortunately, our robots have access to a comprehensive and accurate world model built from observations of the LRF and the vision system by the full team of robots [5, 6, 16].

Path planning has been well researched with a lot of interesting results found [8]. There are at least 3 basic categories approaches can be divided into: roadmaps, cell decomposition, and potential fields. Recently, probabilistic approaches became popular, in particular for finding paths in a high-dimensional configuration space [7]. As our robots only move in a 2 dimensional world and are able to turn on the spot, plus all objects can be reasonably represented as circles, we can restrict the configuration space to only two dimensions. Thus, a deterministic approach can be employed.

Roadmap algorithms are a popular way for finding paths in the RoboCup environment, e.g. the *All Botz* RoboCup team in the small size league developed a variant of the *extended visibility graph* for finding paths around other players [2]. We also implemented a version of the extended visibility graph for our first participation in 1998 [6]. However, this approach has the drawback that it depends on the chosen minimum distance the robot should keep to obstacles. If chosen small, the algorithm can generate paths that are too close to the robots. Especially for soccer this can be disadvantageous as obstacles like opponent robots are moving and collisions are likely to happen. On the other hand, a large value for the minimum distance might cause the planner to fail finding a path. Since it is difficult to find a good trade off we developed a new path planning approach based on a combination of potential fields and cell decomposition. We will show a comparison of the two approaches at the end of this section.

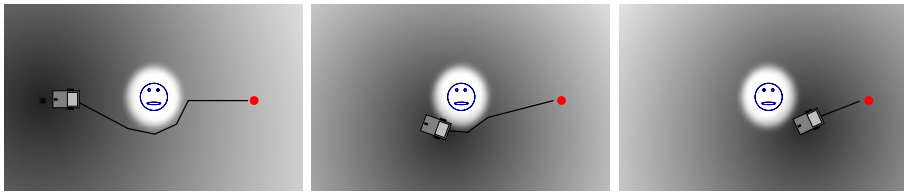
The basic algorithm is described in detail in Topor's work [14]. Due to limited space we will only give a brief overview and present extensions that improved the performance of the algorithm. The path planning algorithm uses a potential field for finding its way through the obstacles. This potential field consists of an attracting force towards the target position and repulsive forces arising from other players, the ball (optionally), and the field boundaries. An additional potential field directs the search into the robot's current heading. In our approach we approximate the world into a grid of 10×10 cm cells.

The search algorithm maintains a grid of the world where each cell holds a boolean indicating if the cell has already been visited. Furthermore, a priority queue is used with elements consisting of a cell index and its corresponding potential value. The algorithm starts with the target position and follows the negative gradient of the potential field to the robot. In each iteration the potential and gradient value of the cell referring to the current position are computed by summing up pre-computed potential and gradient fields of the individual forces. Cell index and potential value of cells that are not already visited are inserted into the priority queue. Note, that although we discretize the world into a grid, the algorithm still computes positions with floating point precision.

Figure 4(a) illustrates how this algorithm finds a path around an obstacle. Note that we search from the target towards the robot and not vice versa, since otherwise the robot would first move directly in direction of the obstacle before turning to the side. In each cycle of the robot control program (every 100 ms), a new path is computed and the robot smoothly moves around the obstacle (Figures 4(b) and (c)). The example also shows that in our algorithm re-using a previously computed path like in the approach of Baltes and Hildreth [2] is not possible without major modifications.

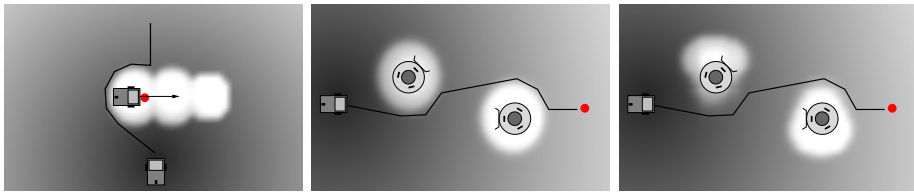
Figure 5 displays two features we added to the original algorithm. If a player is in possession of the ball and is heading towards the opponent goal, we surely do not want other robots to interfere with it. This can be achieved by adding additional, virtual potentials in front of the ball-owning robot (Figure 5(a)).

A problem in path planning in general is that of oscillation. A situation where an obstacle is in between robot and target can lead to such a problem (see Figure 4(a)). If the position of the obstacle is noisy, e.g. because the sensors delivers noisy data, the path planning algorithm might find paths around the obstacles on both sides with equal chance. Since paths are re-planned every 100 ms, the robot might continuously



(a) (b) (c)

Fig. 4. Path planning around an obstacle using potential fields: initial plan (a), plan after robot moved closer to obstacle (b), and final straight-line path to target (3). Potential values are indicated by grey-scale levels where lower potentials appear darker.



(a) (b) (c)

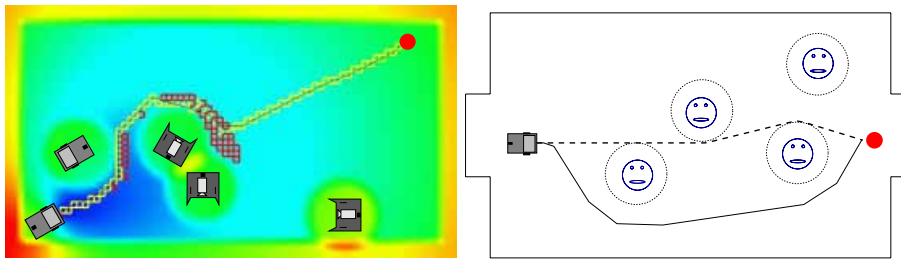
Fig. 5. Extra potential fields are added in front of the active player (a). To reduce oscillation the center of gravity of an obstacle is shifted according to the path computed in the last cycle: without (b) and with hysteresis shift (c).

choose the other way. This could be observed in experiments in our laboratory including the robot bumping into the obstacle because it wasn't able to decide which way to go around.

A directional potential field that prefers the robot's current heading helps avoiding such scenarios. However, if the robot is far from the obstacle, the algorithm still can oscillate. For RoboCup 2001 we added a new technique for directing the search into the right way. Based on the path computed in the previously cycle a hysteresis shift is applied to the individual potential fields of the obstacles. This shift moves the center of gravity of a potential field while still retaining the shape at the border of the obstacle, i.e. the center of the obstacle is shifted while the obstacle still occupies the same space. Due to this shift, the gradient changes and forces the search to go around the obstacle in the same way as in the previous cycle. Figures 5(b) and (c) give an example.

Potential field methods can be trapped in local minima. In our approach this can be detected by examining the visited flag of the cell that refers to the current position. If a cell is visited for the second time, a best first search is started beginning with the cell that has the lowest potential in the priority queue. In best first search, the current cell is removed from the queue and its 4 neighbors are examined and inserted into the priority queue if not already visited. This leads to a wave-front search in the potential field until the negative gradient of a cell points towards a non-visited cell where the algorithm can follow steepest decent again. Figure 6 shows an example.

For evaluating the path planner, we conducted several experiments using random start, target and object positions in a simulated environment of size 81×45. In average,



(a) (b)

Fig. 6. Evaluation of the path planner. Best first search in local minima (a). Dark cells indicate steps where best first search has been applied, light cells refer to steepest decent iterations. Comparison of path planning methods (b). Dashed line refers to shortest path as computed by the extended visibility graph, solid line reflects path generated by our new potential field method.

the running time was less than 1.5 msec on a Pentium MMX 233 MHz. In 35 out of 5000 cases, the algorithm needed more than 15 msec and the maximum run time was about 29 msec. Thus, the approach is well suited for continuous path planning.

A last experiment was conducted to compare the performance of the new algorithm to the extended visibility graph approach we employed for our first participation [6]. Figure 6 (b) shows the setup. The extended visibility approach has the property that it computes shortest paths and for the given scenario it returned the path that goes through the middle of the field. In contrast, our new approach leads the robot around the obstacles close to the wall. Although the length of the path at the wall is significantly longer than the shortest one, our robots need much less time following the longer path (13 s) than following the shorter one (24 s) because our motion controller allows for higher speeds if obstacles are far.

5 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 strategical positions on the field. In this section we describe our method for determining such positions depending on the current game situation and illustrate how our players decide which position they should occupy.

To express that a player has a task which is related to a position in the team formation we say a player pursues a certain *role* [12]. Distinguishing between different areas of responsibility we established the following roles:

- *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 our own half.

– *support*: the supporting player serves the team considering the current game situation. In defensive play it complements the teams' 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 balls position, direction and velocity.

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

5.1 Positions

Our approach to determine the target positions associated with the field player roles is similar to the SPAR method of the *CMU* team in the small size league [11]. 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.

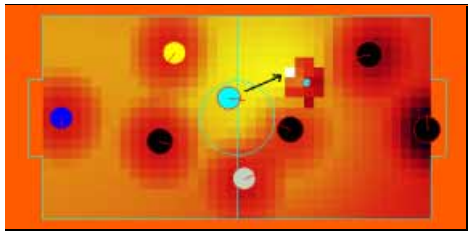


Fig. 7. Potential field for determining the active position

Figure 7 shows an example of a potential field for the desired position of the active player. Dark cells indicate very undesirable positions whereas light positions represent very desirable positions. The resulting position is marked white. We consider the ideal position for the active player 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 balls' position. A defending player should be placed between the ball and our own goal, but in offensive play the ball should be between the player and the opponent goal.

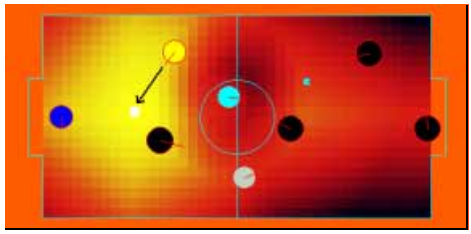


Fig. 8. Potential field for determining the strategic position

Figure 8 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. We want the strategic player to stay well behind all players and the ball and prefer positions close to the horizontal centerline of the field. 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.

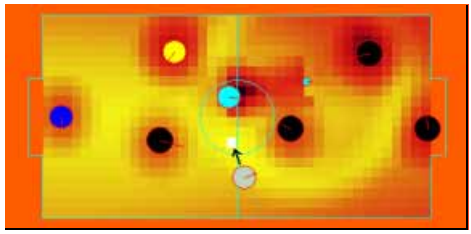


Fig. 9. Potential field for determining the support position

Figure 9 shows how in the same game situation as above the defensive support position is determined. We want the supporter to stay away from all other players and at a certain distance to the ball. As the supporting player should complement the teams' defensive formation, we additionally prefer positions behind and aside the active player.

Even though the resulting positions are in general similar to the ones a human observer would establish, we needed to make some restrictions in order to avoid "hysteric" behavior resulting from "overreacting" to the constantly changing environment. Because our robots are turning rather slowly they need a lot of time for even minor position changes. We therefore favor a players' current position with a persistence value and are quite tolerant regarding how close our defending players actually need to get to their positions. In order not to loose too much of precision either, we adjust this tolerance dynamically depending on the players current angle to its target position. By allowing large tolerances for large angles but requiring small tolerances for small angles, we achieve that a player only starts to update its position if the new position differs considerably from the old one. But once the player is moving towards that position the player intends to approach it as close as possible.

Statistics based on the log files of our Seattle games show that the balance between teams is reflected in the average distances between the target positions of our three field players. Against strong teams our players intended to stay in a rather compact formation with average distances of 2051mm against *Trackies* (1:0) or 2270mm against *Fusion* (2:0). When our team was clearly dominant, the players were spread wider over the field with average distances of 3000mm against *CMU* or 3657mm against *Robosix* (16:0).

The fact that in the Seattle games the active player was on average only 973mm away from its target position indicates that our players managed to maintain a formation where always at least one robot was close to the ball. Also the fact that the area for possible strategic positions is quite restricted is reflected in the respective average distance of only 892mm. As the support position differs considerably in offensive and defensive play, the corresponding player showed the largest offset to its target position (1857mm). Evaluating the log files also showed that there is still room for improvement. As it is a tedious task to test and evaluate team behavior online with real robots, we intend to rely more on the simulator that we are currently implementing.

5.2 Roles

After a field player has determined the best active, strategic and support position from its perspective, it calculates utilities which are basically a heuristic for the time it would take the player to reach the corresponding position. The following criteria are taken into account:

- The (euclidian) distance to the target position
- The angle necessary to turn the robot towards the target position
- The angle necessary for the robot to turn at the target position in order to face in direction of the ball
- The angle between the robot and the ball (it is more desirable, especially for the active player, to approach the target position already facing in direction of the ball)
- Objects between the player and the target position

The total utility is now computed as the weighted sum of all criteria. In order to decide which role to take a player sends these utilities to its teammates and compares them with the received ones.

Following a similar approach taken by the *ART* team [4],[3] each player's objective is to take a role so that the sum of the utilities of all players is maximized. In contrast to the *ART* approach a player doesn't take its desired role right away, but checks first if not another player currently pursues the same role and considers that role best for itself as well. As the world models of our players are not identical, the perspectives of our players 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 we reduce with only little extra communication effort the number of situations where more than one player owns 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 we ensure that a player only gives up a role if its real utility for that role is clearly worse than the one of its teammate.

	Goalkeeper	Active	Strategic	Support
Role kept	986.9 s	5.4 s	5.7 s	8.1 s
Role not unique % of playing time	0 %	2.17 %	1.23 %	1.06 %
Role not unique average time	0 s	0.203 s	0.218 s	0.206 s

Table 1. Evaluation of role assignment method

Table 1 displays a statistics evaluating our role assignment method. All values are averaged over the games played at RoboCup 2001 in Seattle. In the first line the times our players kept a certain role are listed. Interestingly the values for the field players are similar to the average role switch time of 7 seconds stated by the *ART* team [4]. The second line shows how much of the total playing time a role was pursued by more than one player. The fact that for the active player this happened in only 2.17% of the total playing time and in even less cases for the other roles, demonstrates, that our method to avoid such situations works successfully. The average values for the times, a role was not unique (in line three), gives further evidence for this.

When evaluating the log files of our Seattle games we also noted that the roles were usually quite equally distributed between our players. However, in games where we

[3] Note, however, that our approach was developed independently.

scored a lot of goals, the player with the most offensive starting position held the active role considerably longer than its teammates.

6 Conclusion

The development of robotic soccer during the last five years was quite impressive. In 1997 the robots hit the ball only occasionally – and often kicked it in the wrong direction (or even into the own goal). In 2001, the games looked much more interesting. The development of our team followed this general path. In 1998, our main advantage was that our robots knew their own position – which helped to avoid own goals. Over the last four years, we concentrated on the improvement of our hardware, on the sensor interpretation process, on cooperative sensing and on team play. As demonstrated, this gave CS Freiburg the chance to win the championship three times. However, having watched the final game against *Osaka Trackies*, we got the impression that a point is reached where it is hard to improve our robots so that they are able to dominate a game against a fast, reactive team such as the *Trackies*.

References

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

[2] Jacky Baltes and Nicholas Hildreth. Adaptive path planner for highly dynamic environments. In Stone et al. [10], pages 76–85.

[3] J. Bruce, Tucker Balch, and Maria Manuela Veloso. Fast and inexpensive color image segmentation for interactive robots. In *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '00)*, volume 3, pages 2061 – 2066, October 2000.

[4] Claudio Castelpietra, Luca Iocchi, Maurizio Piaggio, Alessandro Scalzo, and Antonio Sgorbissa. Communication and coordination among heterogeneous mid-size players: ART99. In Stone et al. [10], pages 149–158.

[5] Markus Dietl, Jens-Steffen Gutmann, and Bernhard Nebel. Cooperative sensing in dynamic environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'01)*, Maui, Hawaii, 2001.

[6] Jens-Steffen Gutmann, Wolfgang Hatzack, Immanuel Herrmann, Bernhard Nebel, Frank Rittinger, Augustinus Topor, Thilo Weigel, and Bruno 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.

[7] L.E. Kavraki and J.C. Latombe. Probabilistic roadmaps for robot path planning. In K. Gupta and A. del Pobil, editors, *Practical Motion Planning in Robotics: Current Approaches and Future Directions*, pages 33–53. John Wiley, 1998.

[8] Jean-Claude Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[9] Bernhard Nebel, Jens-Steffen Gutmann, and Wolfgang Hatzack. The CS Freiburg '99 team. In M. Veloso, E. Pagello, and H. Kitano, editors, *RoboCup-99: Robot Soccer World Cup III*, Lecture Notes in Artificial Intelligence, pages 703–706. Springer-Verlag, 2000.

[10] 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.

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

[12] Peter Stone and Maria Manuela Veloso. Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence*, 1999.

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

[14] Augustinus Topor. Pfadplanung in dynamischen Umgebungen. Diplomarbeit, Albert-Ludwigs-Universität Freiburg, Institut für Informatik, 1999.

[15] Roger Y. Tsai. An efficient and accurate camera calibration technique for 3D machine vision. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 364–374, 1986.

[16] Thilo Weigel, Willi Auerbach, Markus Dietl, Burhard Dümler, Jens-Steffen Gutmann, Kornel Marko, Klaus Müller, Bernhard Nebel, Boris Szerbakowski, and Maximiliam Thiel. CS Freiburg: Doing the right thing in a group. In Stone et al. [10].