The AGILO Autonomous Robot Soccer Team: Computational Principles, Experiences, and Perspectives

Michael Beetz, Sebastian Buck, Robert Hanek, Thorsten Schmitt, and Bernd Radig
Munich University of Technology
Department of Computer Science IX
Orleanstr. 34, D-81667 Munich, Germany

ABSTRACT

This paper describes the computational model underlying the AGILO autonomous robot soccer team, its implementation, and our experiences with it. The most salient aspects of the AGILO control software are that it includes (1) a cooperative probabilistic game state estimator working with a simple off-the-shelf camera system; (2) a situated action selection module that makes amble use of experience-based learning and produces coherent team behavior even if inter-robot communication is perturbed; and (3) a playbook executor that can perform preprogrammed complex soccer plays in appropriate situations by employing plan-based control techniques. The use of such sophisticated state estimation and control techniques distinguishes the AGILO software from many others applied to mid-size autonomous robot soccer. The paper discusses the computational techniques and necessary extensions based on experimental data from the 2001 robot soccer world championship.

1. INTRODUCTION

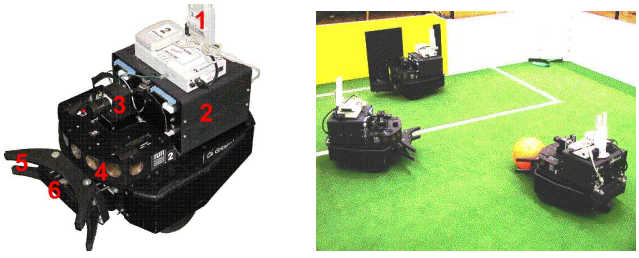
Robotic soccer has become a standard "real-world" testbed for autonomous multi robot control. In robot soccer (mid-size league) two teams of four autonomous robots — one goal keeper and three field players — play soccer against each other. The soccer field is four by nine meters big surrounded by walls. The key characteristics of mid-size robot soccer is that the robots are completely autonomous. Consequently, all sensing and all action selection is done onboard of the individual robots. Skillful play requires our robots to recognize objects, such as other robots, field lines, and goals, and even entire game situations. The robots also need to collaborate by coordinating and synchronizing their actions to achieve their objectives — winning games.

In this paper, we show how the AGILO autonomous robot soccer team meets these challenges. The AGILO robot controllers employ game state estimation, situated action selection, and playbook execution as their main control mechanisms. The game state estimator estimates the complete game situation, including the position of the opponent robots and the ball almost at frame rate using a cheap off-the-shelf camera system. The use of such a vision system yields considerable innaccuracies in the sensor data and highly incomplete information about the game situation. The AGILO game estimator deals with these problems by employing sophisticated probabilistic reasoning techniques and by exploiting the cooperation between the game state estimators of different robots. The second component of the individual robot controllers selects appropriate actions based on an assessment of the current estimated game state. The situated action selection mechanism is the default mechanism for choosing actions and must therefore propose reasonable actions for all possible situations. Because the AGILO robots are nonholonomic and difficult to steer, the AGILO soccer robots employ experience-based learning mechanisms to acquire competence in robot control. Finally, the playbook executor performs preprogrammed complex soccer plays in appropriate situations by employing plan-based control techniques. The use of plan-based action selection enables the robots to additionally exploit knowledge about the current intentions of the team mates.

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

2. OVERVIEW

The AGILO RoboCup team is realized using inexpensive, off the shelf, easily extendible hardware components and a standard software environment. The team consists of four Pioneer I robots; one of them is depicted in figure 1(a). The robot is equipped with a single onboard linux computer (2), a wireless ethernet (1) for communication, and several sonar sensors (4) for collision avoidance. A color CCD camera with an opening angle of 90^o (3) is mounted fix on the robot. The robot also has a guide rail (5) and a kicking device (6) that enable the robot to dribble and shoot the ball.



(a) (b)

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

Besides giving us a substantial handicap in the games, the hardware also confronts us with challenging research problems. The camera system with an opening angle of 90^o and pointed to the front gives an individual robot only a very restricted view of the game situation. Therefore, the robot needs to cooperate to get a

more complete picture of the game situation. Vibrations of the camera, spot light effects, and poor lighting effects cause substantial inaccuracies. Even small vibrations that cause jumps of only two or three pixel lines cause deviations of more than half a meter in the depth estimation, if the objects are several meters away. The robots are nonholonomic which implies that reaching many target positions requires complex and accurate driving maneuvers. As a consequence, simple heuristics for deciding which robot should get to the ball and how do not work well.

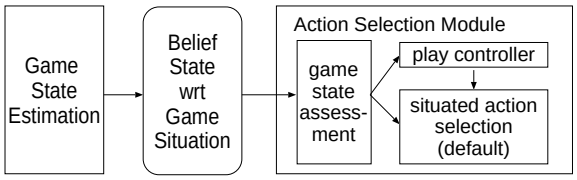


Figure 2: Software architecture of an AGILO robot controller.

The main software components of the AGILO robot controllers are depicted in figure 2. The incoming video streams are processed by the *vision-based cooperative state estimation module*, which computes the *belief state* of the robot with respect to the game situation. The action selection module then computes an abstract feature description of the estimated game state that can be used to recognize relevant game situations. There are two basic components for action selection. First, the situated action selection module selects action based on a limited horizon utility assessment. Second, a plan-based controller that enables the team to execute more complex and learned plays. The sections 3-5 detail the software design and the operation of these software components.

3. VISION-BASED, COOPERATIVE GAME STATE ESTIMATION

The game state estimators of the AGILO robots maintain a belief state that contains the respective robot's belief about the current game situation [16]. The belief state includes the estimated positions and orientations of the robot itself, its team mates, the ball, and the opponent robots and provides the information that is necessary for selecting the appropriate actions (see figure 3).

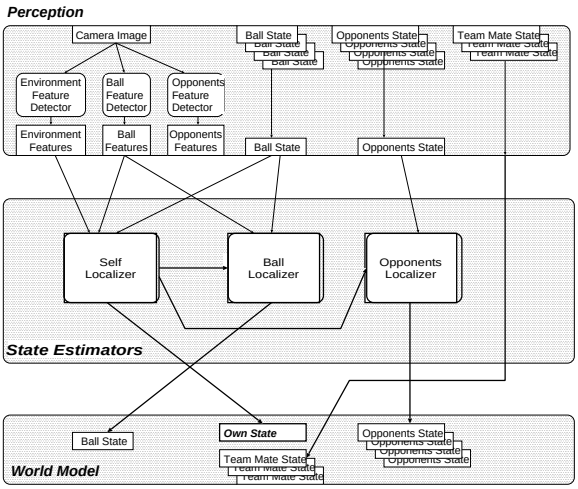


Figure 3: Software architecture of the state estimator.

Estimating the game state both accurately and reliably is very difficult because the state is to be estimated by multiple mobile inaccurate sensors with uncertain positions, the soccer field is only partly accessible for each sensor due to occlusion caused by other robots and a limited camera view, the robots change their direction and speed very abruptly, the models of the dynamic states of the robots of the other team are very crude and uncertain, and it requires the processing of vast amounts of data very quickly.

State estimation is an iterative process where each iteration is triggered by the arrival of a new piece of evidence, a captured image or a state estimate broadcasted by another robot. The state estimation subsystem consists of three interacting estimators: the self localization system, the ball estimator, and the opponents estimator. This decomposition of game state estimation into specialized estimation problems reduces the overall complexity of the state estimation process and enables the robots to exploit the structures and assumptions underlying the different subtasks of the complete estimation task. The remainder of this section describes the components of the game state estimator.

3.1 Perception

The information needed for game state estimation is provided by the perception system and includes the following kinds of information: (1) partial state estimates broadcasted by other robots, (2) feature maps extracted from captured images, and (3) odometric information. The estimates broadcasted by the team mates comprise the respective robot's location, the ball's location, and the locations of the opponents. From the captured camera images the feature detectors extract problem-specific feature maps that correspond to (1) static objects in the environment including the goal, the borders of the field, and the lines on the field, (2) a color blob corresponding to the ball, and (3) the visual features of the opponents.

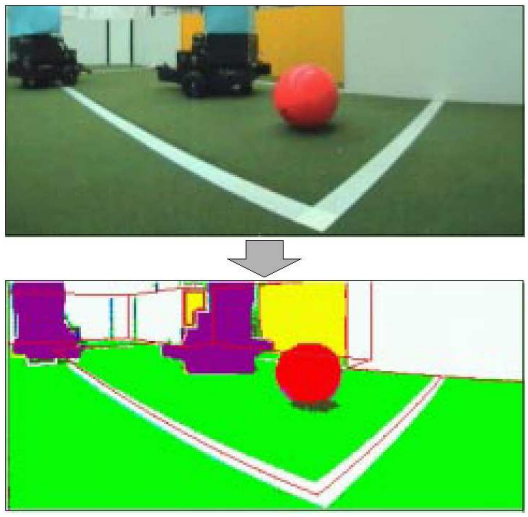


Figure 4: The figure shows an image captured by the robot and the feature map that is computed for self, ball, and opponent localization.

The working horse of the perception component are a color classification and segmentation algorithm that is used to segment a captured image into colored regions and blobs (see figure 4). The color classifier is learned in a training session before tournaments in order to adapt the vision system to specific lighting conditions and effects. We are currently working on the next version of the classifier, which will be capable of automatically adjusting itself to changing lighting conditions during the game.

3.2 Self Localization

Each robot's belief about its own position is maintained as a probability density function that maps the possible positions of the robot into the probability density that the respective position has generated the observations of the robot and is caused by the sequence of driving actions issued by the robot. The probability density is approximated by a multi-variate Gaussian density and represented using its mean vector $\vec{\Phi}$ and its covariance matrix $C_{\vec{\Phi}}$.

The robot's model of the static part of its environment that is used for self localization is composed of landmarks together with their positions and orientations. The landmarks include goals, field lines, and walls surrounding the pitch. Figure 5 depicts an excerpt of the environment model representing the neighborhood around a goal, which is used for self localization in the robot soccer domain. The goal is modeled as a set of 3D lines where each line is associated with a color transition. Using the world model and a position estimate, the robot can predict where in a captured image lines should be visible and which color transition they represent.

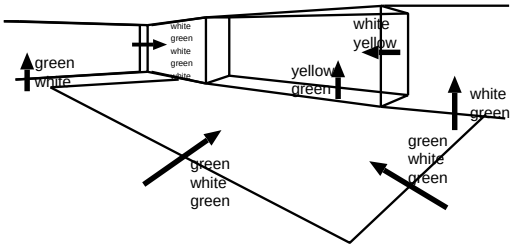


Figure 5: Model of the neighborhood of a goal. The model contains the edges of the objects and the color transition they are the borderline of.

The self-localization algorithm [6] itself consists the iteration of three steps. First, the visible 3D curve features are predicted and projected into the image plane given the current estimated position. Second, a local search is performed to establish correspondences between the predicted and detected 3D curve features. Third, a maximum a-posteriori (MAP) estimation step computes an estimate $\hat{\vec{\Phi}}$ of the robot pose which best fits to the position prediction $p(\vec{\Phi})$ and to the image observations $\tilde{P}_{i,j}$.

3.3 Opponent Tracking

The opponent tracker tracks the positions of the other team's robots by (1) detecting feature blobs in the captured image that might correspond to an opponent, (2) estimating the world coordinates and uncertainties of these blobs, and (3) associating them with the object hypotheses — tracked opponents.

Rough play ground, camera vibrations caused by accelerations, unknown shapes of the opponent robots, and unpredictable opponent movements cause a lot of uncertainty. Therefore, the state estimator has to deal with two different kinds of uncertainties. The first one is the inaccuracy of the robot's sensors. We represent this kind of uncertainty using a Gaussian probability density. The second kind of uncertainty is introduced by the data association problem, i.e. assigning feature blobs to object hypotheses. This uncertainty is represented by a hypotheses tree where nodes represent the association of a feature blob with an object hypothesis. A node $H_j(t)$ is a son of the node $H_i(t-1)$ if $H_j(t)$ results from the assignment of an observed feature blob with a predicted state of the hypothesis $H_i(t-1)$. In order to constrain the growth of the hypotheses tree, it is pruned to eliminate improbable branches with every iteration of the MHT. Our MHT Algorithm (an extension of Reid's Multiple Hypotheses Tracking (MHT) algorithm [13]) is capable of handling multiple mobile sensors with uncertain positions. Each iteration begins with the set of hypotheses $H(t-1)$ from the previous iteration $t-1$. Each hypothesis represents a different assignment of measurements to objects, which was performed in the past. The algorithm maintains a Kalman filter for each hypothesis. For each hypothesis a position of the dynamic objects is predicted $\hat{Z}_i(t)$ and compared with the next observed opponent performed by an arbitrary robot of the team. Assignments of measurements to objects are accomplished on the basis of a statistical distance measurement. Each subsequent child hypothesis represents one possible interpretation of the set of observed objects and, together with its parent hypothesis, represents a possible interpretation of past observations assigned to the object. With every iteration probabilities describing the validity of an hypothesis are calculated [4].

3.4 Cooperative State Estimation

The state estimation modules of different robots cooperate to increase the coverage, accuracy, and reliability of the estimation process. This is achieved by the individual robots broadcasting their own observations and their covariances to their team mates. The other robots use these broadcasted estimates as additional evidence for their own estimation processes. Of course, how to weigh the broadcasted estimates against the own perceived features is a subtle and delicate issue. For example, we are weighing the own estimates of the ball much higher than the broadcasted estimates. This implies that in behaviors such as going to the ball the control signals are much tighter coupled to the robot's own perception. The other robots' estimates of the ball are only considered as stronger evidence if the ball is far away from the robot itself or not visible at all. We are currently analyzing the advantages and disadvantages of different parameterizations using our log files of the robocup competition in this year.

Despite crude parameterizations, the cooperation between the robots enables them to track temporarily occluded objects and to faster recover their position after they have lost track of it. A representative result of our cooperative game state estimation process is shown in figure 10.

4. SITUATED ACTION SELECTION AND EXECUTION

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

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

To achieve a high degree of autonomy the AGILO robots per-

form the task assignment and execution distributedly on the individual robots. This makes the task assignment more robust against problems in inter robot communication. These problems can be caused by robots being sent off the field, computers being crashed after heavy collisions, and communication being corrupted due to interferences with other communication channels.

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

4.1 AGILO Simulator: a Tool for Learning

An important means for developing competent robot soccer skills is a robot simulator that allows for realistic, controllable, and repeatable experiments. For this reason we have developed a robot simulator that accurately simulates how the dynamic state of the robot changes as the robot's control system issues new driving commands such as setting the target translational and rotational velocities. The AGILO software development environment therefore provides a robot simulator that uses multi layer neural network [7] to simulate the dynamics of the AGILO soccer robots. We have used the RPROP algorithm [14] for supervised learning in order to teach the neural net the mapping dynamic states and control signals into the subsequent states.

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

4.2 Task Assignment

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

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

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

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

The AGILO task cost estimator performs three steps. First, the

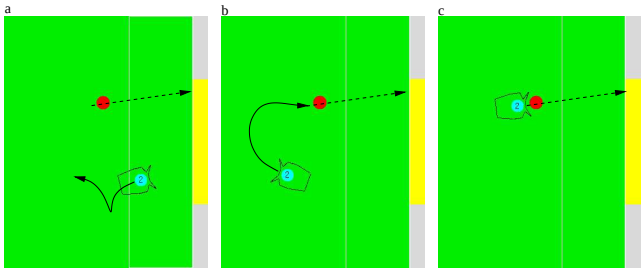


Figure 6: A training scenario in the multi robot simulation environment: To acquire training patterns for the neural projector $\mathcal{P}$ a robot is set to a randomly defined start state ζ_s (position of the robot in subfigure a) and has to drive to a randomly defined target state ζ_t indicated by the dashed arrow. The direction and length of this arrow indicate the target state's orientation and velocity. The time the robot needs to reach its target state (subfigure b & c) is taken to complete the training pattern $\langle\langle\zeta_s, \zeta_t\rangle$,*time*$\rangle$.

selection of the multi robot navigation method that matches the game state best. By taking the estimated game state into account the cost estimator can take even expectations about the movements of the opponents into account. Second, computing a path in the context of the navigation paths of the team mates. This is done by computing navigation paths that avoid negative interferences with the paths computed for the higher priority tasks (see section 4.3). Third, the proposed path is then decomposed into a sequence of simpler navigation tasks for which the time cost can be accurately predicted using a neural network. The mapping from navigation tasks, given by start points and destinations, into the time cost needed for the task completion is realized through a multi layer artificial neural network and learned through the backpropagation derivative RPROP [**?**]. We have trained the network using about 300.000 training patterns generated from accomplishing random navigation tasks in the learned simulator (see figure 6).

4.3 Multi Robot Navigation Planning

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

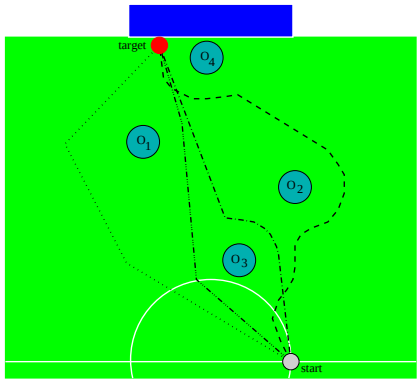


Figure 7: Navigation plans for a given single robot navigation task as proposed by different navigation planning methods.

But which navigation planning method should a robot apply to achieve its objectives? Contemplate figure 7. The figure depicts a single robot navigation task in a typical game situation and the navigation plans proposed by different navigation planning algo-

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

Rather than designing yet another multi robot navigation algorithm we have decided to equip the AGILO robots with a hybrid robot navigation planning system. The system [3] employs different single robot navigation and plan merging mechanisms and selects the appropriate methods based on an assessment of the given navigation task and game situation. This way the system can exploit the different properties of the individual methods by learning for which navigation tasks the methods are best suited.

The planning methods employed by the AGILO navigation system include the *Potential Field Method* [9], the *Shortest Path Method* [9], *Circumnavigating Obstacles* [9], and *Maximizing the Clearance* [9]. Plan merging and repair methods include methods for merging plans that add waiting steps in order to avoid interferences. Path replanning methods revise the individual plans such that no negative interferences will occur include the *Definition of Temporary Targets*, the *Hallucination of Obstacles at Critical Sections*, and the *Insertion of New Obstacles*. The first one modifies the path by introducing additional intermediate target points. The second one hallucinates additional obstacles at the positions where collisions might occur. The third one simply considers the other robot at its respective position as a static obstacle.

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

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

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

We have learned a set of 10 rules including the one above using the decision tree learning C4.5 algorithm [12] with standard parameterization and subsequent rule extraction. To do so we have collected a training set of 1000 data records, where each data record contained a description of a randomly generated navigation task and the time resources required to complete the task for each possible combination of navigation planning and plan repair method.

The language for characterizing navigation tasks uses 7 features (see figure 8): (1) the number of intersections between the line segments that represent the navigation tasks, (2) the size of the bounding box of the navigation tasks, (3) the minimal linear distance between different starting positions, (4) the minimal linear distance between different target positions, (5) the minimal distance between the line segments that represent the navigation tasks, (6) the maximum length of the linear distances of the individual navigation tasks, and (7) the number of obstacles in the bounding box of the joint navigation task.

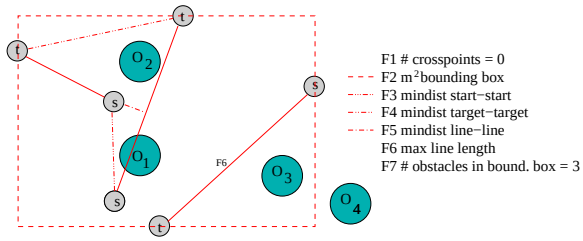


Figure 8: Visualization of navigation task features that are used for classifying navigation tasks.

To sum up, the AGILO multi robot navigation algorithm works as follws. First, the appropriate planning mechanism is selected based on the assessment of the given navigation task and the situation in which it is to be executed. In the second step, the joint navigation task is decomposed into single robot navigation problems. The individual problems are then solved using the selected planning methods. Then, the individual plans are repaired in order to avoid negative interferences with the higher priority plans. Finally, the algorithm extracts sequences of target states from the robot's own navigation plan and sends those sequences to the robot's neural network controller, which is further described in the next section.

4.4 Execution of Navigation Plans

The *robot motion controllers* are used for achieving given dynamic states as fast as possible. The motion controller receives the target state (for example, the next state on a planned path) of a robot and returns low level commands that transform the current state into the target state as fast as possible. To arrive at the target state different trajectories are possible. But how to set them to quickly reach the target state? The AGILO robot controllers learn a direct mapping from the robot's current state (ζ_0) and the robot's target state (ζ_{target}) to the next command to be executed (ξ_0) using multi layer artificial neural networks [7] and the RPROP [14] algorithm: *Net*: $\langle \zeta_0, \zeta_{target} \rangle \mapsto \xi_0$.

5. PLAN-BASED CONTROL

While our situated action selection aims at choosing actions that have the highest expected utility in the respective situation it does not take into account a *strategic* assessment of the alternative actions and the respective *intentions* of the team mates. This is the task of the plan-based action control. While situated action selection achieves an impressive level of performance it is still hampered by the requirement for small action and state spaces, a limited temporal horizon, and without explicitly taking the intentions of the team mates into account.

The goal of plan-based control in robotic soccer is therefore to improve the performance of the robot soccer team by adding the capability of learning and execute soccer plays. Soccer plays are properly synchronized, cooperative macro actions that can be executed in certain game contexts and have, in these contexts, a high success rate. Plans for soccer plays specify how the individual players of a team should respond to changing game situations in order to perform the play successfully.

The integration of soccer plays into the game strategies enables robot teams to consider play specific state spaces for action selection, parameterization, and synchronization. In addition, the state space can reflect the intentions of the other robots. An action that is typically bad might be very good if I know that my team mate intends to make a particular move. Further, action selection can consider a wider time horizont, and the robots can employ play

specific routines for recognizing relevant game situations.

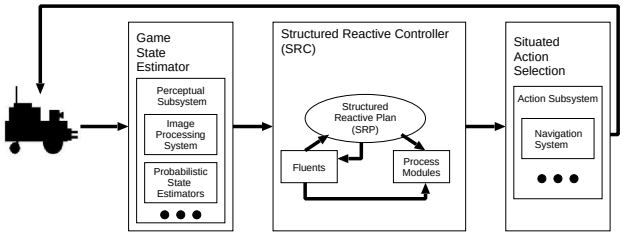


Figure 9: Block diagram of the high-level controller of the AGILO soccer robots.

In order to realize an action assessment based on strategic consideration and on considerations of the intentions of the teammates, we develop a robot soccer playbook, a library of plan schemata that specify how to perform individual team plays. The plans, or better plays, are triggered by opportunities, for example, the opponent team leaving one side open. The plays themselves specify highly reactive, conditional, and properly synchronized behavior for the individual players of the team.

The high-level controller of each soccer robot is realized as a structured reactive controller (SRC) [1] and implemented in an extended RPL plan language [10]. The high-level controller works as follows. It executes as the default strategy the situated action selection. At the same time, the controller continually monitors the estimated game situation in order to detect opportunities for making plays. If an opportunity is detected, the controller decides based on circumstances including the score and the estimated success probability of the intended play whether or not to perform the play.

6. DEMONSTRATION

The AGILO robot soccer team, described in this paper, has participated in the fifth robot soccer world championship in Seattle (2001). The team has played six games for a total of about 120 minutes. The team advanced to the quarter finals playing games against Sharif CE (7:0), SPQR (7:0), Eigen (0:4), Ulm Sparrows (7:0), GMD Robots (1:1), and COPS Stuttgart (0:1). In the tournament none of the AGILO players was sent off the field because of causing collisions with opponent players or not leaving penalty area in time. Most of the occasions in which the AGILO players had to be taken off to be restarted seemed to be caused by hardware problems. Unfortunately, in midsize robot soccer there is no external sensing device which records a global view of the game and can be used as the ground truth for experiments. Thus for the experimental results in this section we can only use the subjective information of our robots and argue for the plausibility of their behavior and belief states.

6.1 Game State Estimation

A typical result of the AGILO game state estimator is shown in figure 10. The upper picture shows the tracked positions of the AGILO players and the ball. The lower picture shows the individual observations of the opponent robots. The colors of the observations indicate which AGILO robot the observation made. The dashed lines show the tracks of the opponent robots. You have to look at the color figures in order to see the individual observations of the different robots and how they are merged into a consistent track. Qualitatively, we can estimate the accuracy of the game state estimation by looking for the jumps in the tracked lines. We can see that the own tracks are smooth and can therefore be expected to be accurate. Also, the tracks of the ball look very reasonable. The tracks of the opponents are less accurate and only partial. This must be expected due to the high inaccuracy and incompleteness of the sensory data and the lack of odometry data. One can see that the blue AGILO robot occasionally perceived the AGILO goalie as an opponent player.

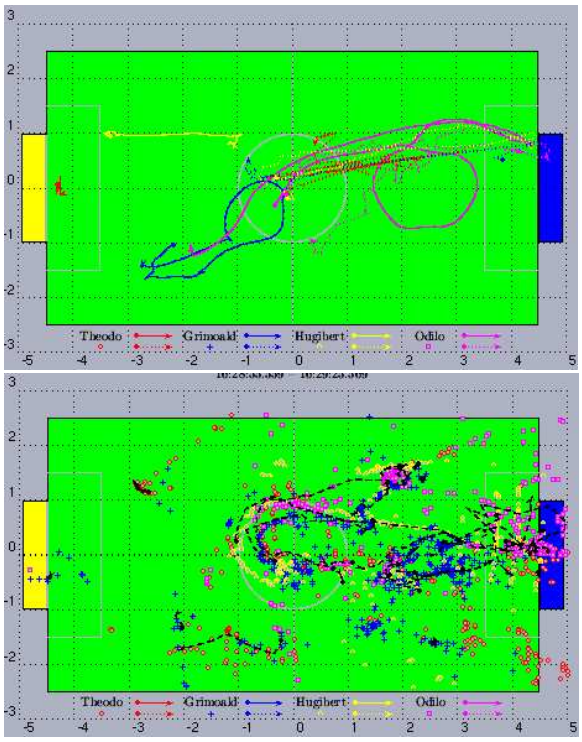


Figure 10: Results of the game state estimation process during a thirty seconds game episode against Ulm Sparrows in which the AGILO team scores a goal. The upper picture shows the results of the self localization processes of the individual robots and the tracking results for the ball (dotted lines) The lower picture shows the results of tracking the opponent players.

We are very pleased with the performance of our game state estimation module. The self localizations of the robots seems to work very accurately and reliably. There are very few jumps in the tracks which would indicate localization errors. The same holds for the ball track. As can be expected, tracking the opponent players is much more difficult. It is very hard to extract the tracks out of the clouds of different observations. However, we see that several tracks resulted from merging the observations of different robots. In addition, the merging of the different observations results in fewer hallucinated obstacles and therefore allows for more effcient navigation paths.

Our preliminary results suggest that purely image-based probabilistic estimation of complex game states is feasible in real time even in complex and fast changing environments. We have also seen that maintaining trees of possible tracks is particularly useful for estimating a global state based on multiple mobile sensors with position uncertainty. Finally, we have seen how the state estimation modules of individual robots can cooperate in order to produce more accurate and reliable state estimation.

There are several ways in which we intend to advance our game estimation methods. First of all, we want to better understand how the action models for the opponent players and the parameters of

the algorithms should be set to produce better results. It is also an open question how we could use prior information such that there are at most four opponent robots on the field or there is most of the time an opponent in its own goal area can be used to obtain better results. Another important aspect is that effective play requires the team to always know where the ball is. We want to develop a method for active ball localization. Thus whenever the team does not know where the ball is it should look for it in a goal directed way. One possible way to do this is to choose actions such that the actions reduce the entropy in the probability distribution over the ball's position [5]. Yet another extension is to enable state estimation to deal with ambiguous states. This could be achieved by running a Markov localization process in parallel. Finally, we want to improve the perceptual capabilities of the robots through an online mechanism for the adaptation of color classification.

6.2 Action Selection

The action selection is even more difficult to evaluate. A weak indication of the coherence of coordination is the number of robots performing *go2ball* at the same time. Ideally there should always be exactly one robot going for the ball if the team knows where the ball is. The statistics extracted from the log files of the Seattle tell us that 98.64% of the cycles exactly one robot was going to the ball, in 0.34% no robot, and in 1.02% of the cycles more than one. The average duration that a robot performs *go2ball* or handles the ball without being interrupted by a decision of a fellow robot is 3.35 seconds. In only 0.25% of the time a robot that is stuck is determined to go for the ball by the other robots. Thiese results suggest that the task assignment algorithm together with our task cost estimation mechanism works well.

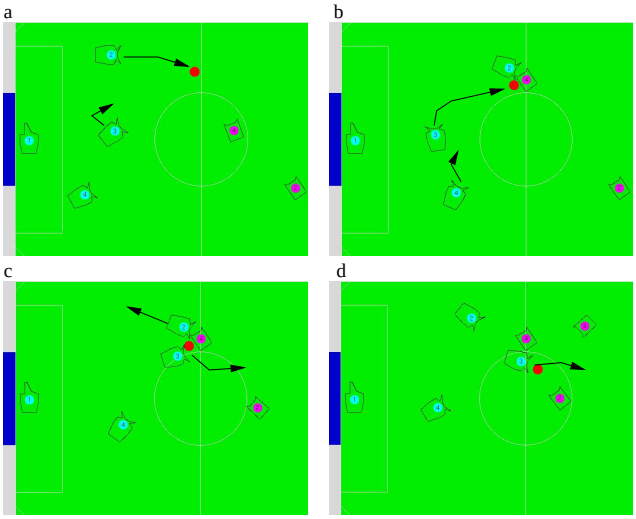


Figure 11: An example for intelligent cooperation in a real robot soccer environment. Robot 2 approaches the ball (subfigure a) and thereby collides with a robot of the opponent team (b). As the opponent robot constantly pushes robot 2 is stuck and temporary not regarded by the other robots. Thus robot 3 moves towards the ball while robot 2 tries to get unstuck (c). Finally robot 3 dribbles towards the opponent goal while robot 2 is staying back in its own half (d).

Figure 11 shows a kind of situation that has occurred several times during the Robo Cup and is replayed in the AGILO simulator. Robot number 2 is supposed to be the fastest to get the ball and therefore approaches the ball (fig. 11a). Near the ball robot 2 collides with an opponent robot. Robot 2 is in a deadlock situation and cannot move forward anymore. The only action feasible to execute remains *get_unstuck*. Thus robot 3 approaches the ball now (fig. 11b) Having reached the ball robot 3 dribbles towards the opponent goal while robot 2 is moving backwards (fig. 11c): Further on robot 2 is no more stuck and robot 3 is still dribbling. Buck et al. [3] present more conclusive results obtained in the Robo Cup simulation league that show that the team performance using the task assignment algorithm degrades gracefully as the corruption level for communication is increased. They also show how the task assignment algorithm achieves more complex patterns of cooperation such as double passes.

We have also evaluated our learned hybrid multi robot navigation system in the AGILO robot simulator. To do so, we have compared its performance with the performance obtained by the individual navigation methods. The results are shown in table 1. We have performed a bootstrapping t-test based on 1000 different joint navigation tasks to empirically validate that the hybrid navigation planner performs better than the individual planning methods. Based on these experiments we obtained a 99.9% confidence in the test set (99.9% in the training set) that the hybrid method outperforms the potential field method (with its respective parameterization). The respective probabilities for the shortest path method are 99.9% (99.9%), for the maximum clearance method 99.84% (99.71%), and for the viapoint method 96.25% (94.62%). This validates our hypothesis that the hybrid planner dominates the other planning methods with statistical significance ($\geq 95\%$).

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

Table 1: Results of four evaluated algorithms and the trained decision tree. The significance level is based on a t-test.

Besides the performance the frequency with which the AGILO system selects actions is also impressive. There are on average 4-5 action selection cycles per second despite the sophistication of task cost prediction and multi robot navigation. This speed can be reached because the results of complex computations are estimated through neural networks and decision trees that have been trained using experience-based learning mechanisms.

So far our plan-based control mechanisms have only be applied in the AGILO robocup simulator and not yet in extensive experiments. Our next steps in the advancement of our action selection mechanisms are the autonomous learning of more complex learning tasks, such as dribbling towards the opponent goal and shooting in the right moment in the right corner and getting the ball away from the wall. Another important issue is action selection under uncertainty. The skills that we have learned so far were all acquired with the simulator using a perfect world model. We believe that we can learn much better skills if our simulator can also learn probabilistic models of the AGILO state estimation processes. To learn such a probabilistic perception model we need however a ceiling camera that records the ground truth that the estimated states can be compared to. Finally, we believe that in order to acquire more skills more autonomously it is crucial to better integrate learning mechanisms into robot control languages. To this end we extend our plan-based control language RPL such that it is capable of declara-

tively specifying learning problems within the control routines.

7. RELATED WORK

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

In the mid-size league most competing teams apply behavior-based control techniques and avoid the problem of estimating the complete game state with the CS Freiburg team being a notable exception [11]. However, because the Freiburg team is using Laser range finders as their primary sensors, which are very accurate in depth estimation, they get away with a simpler state estimation mechanism in which can assume almost perfect sensors with known positions. Most other mid-size teams coordinate the play of their team mates by negotiating or assigning roles to the different players [8]. In contrast, in the AGILO team the coordination is implicit and based on a sophisticated cost estimate for task assignment. The AGILO team is also distinguished in the mid-size league with respect to its extensive use of learning and plan-based control mechanisms. Technologically, the AGILO software shares control mechanisms with teams in the simulator league. In particular, it applies similar learning techniques as the Karlsruhe Brainstormers [15]. The use of such techniques in **autonomous** robot soccer is much more difficult due to the difficulties in obtaining sufficient training data, high variances in physical effects, extremely noisy sensors, and the incompleteness of available information.

With respect to the software architecture and employed software techniques the AGILO control software shares commonalities with autonomous robotic agents such as the extended RHINO control system [2]. The RHINO system, too, makes extensive use of probabilistic state estimation, has a default mechanism for action selection, and plan-based control mechanism. The AGILO software extends this work in that it applies these techniques to a multi robot control problem in a very dynamic and adversary environment.

8. CONCLUSION

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

We have shown that the application of probabilistic state estimation techniques together with information exchange between the robots results in game state estimators that are capable of estimating complete states including robots with surprising accuracy and robustness even with restrictive and noisy camera systems. We have also seen that the amble use of experience-based learning has resulted in powerful control mechanisms, including competent coordination, with little runtime computational cost. Finally, we have explained how plan-based control mechanisms will enhance the robot's playing skills by enabling the robots to perform complex soccer plays. The results of the 2001 robot soccer world championship have shown that these techniques allow for competitive soccer play despite an inferior hardware equipment.

9. REFERENCES

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

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

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

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

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

[6] R. Hanek and T. Schmitt. Vision-based localization and data fusion in a system of cooperating mobile robots. In *International Conference on Intelligent Robots and Systems (IROS)*, 2000.

[7] J. Hertz, A. Krogh, and R.G. Palmer. *Introduction to the Theory of Neural Computation*. Addison-Wesley, 1991.

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

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

[10] D. McDermott. A reactive plan language. Research Report YALEU/DCS/RR-864, Yale University, 1991.

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

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

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

[14] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: the rprop algorithm. In *Proceedings of the ICNN 1993*, 1993.

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

[16] T. Schmitt, R. Hanek, S. Buck, and M. Beetz. Cooperative probabilistic state estimation for vision-based autonomous mobile robots. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2001.