AGILO RoboCuppers 2004

Freek Stulp, Alexandra Kirsch, Suat Gedikli, and Michael Beetz

Munich University of Technology, Germany
`agilo-teamleader@mail9.in.tum.de`
`http://www9.in.tum.de/agilo/`

1 System Overview

The AGILO RoboCup team is the primary platform for our research on the semi-automatic acquisition of visuo-motoric plans. It is realized using inexpensive, off the shelf, easily extendible hardware components and a standard software environment.

The control system of an autonomous soccer robot consists of a probabilistic game state estimator and a situated action selection module [2, 4]. The game state estimator computes the robot's belief state with respect to the current game situation [16, 15]. The action selection module selects actions according to specified goals as well as learned experiences. Automatic learning techniques made it possible to develop fast and skillful routines for approaching the ball, assigning roles, and performing coordinated plays [10, 3].

The control program and the software development environment of the AGILO RoboCuppers 2004 advance the computational mechanisms of the earlier versions of the system in several important ways:

1. For the first time our probabilistic state estimation routines will employ *learned* observation and environment models and *active* state estimation routines.
2. The action selection is rationally reconstructed and completely re-implemented using ***ROLL*** (***RO***bot ***L***earning ***L***anguage). Using ROLL, learning and control tasks are stated explicitly *as part of the code*.
3. The control system employs *model-based transformational learning and planning* mechanisms in order to improve the system performance.
4. Finally, we have substantially extended the development environment for programming the control system. These extensions include a system for recording ground truth data and a software workbench for empirically analyzing robot behavior and perception.

2 Hardware Platform

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 customized Pioneer I robots (1); one of which is depicted in figure 1. The robot uses the original Pioneer I controller-board (2) and differential drive (3). For ball handling the robot has a passive ball guide rail (4) and a spring-based kicking device (5).

The only sensor apart from the odometry is a fixed, forward-facing color CCD Firewire camera with a lens opening angle of of 90^o (6). All computation is done on a standard 900 Mhz laptop with Linux operating system (7). The robot uses a wireless Wireless LAN device (8) for communication with teammates.

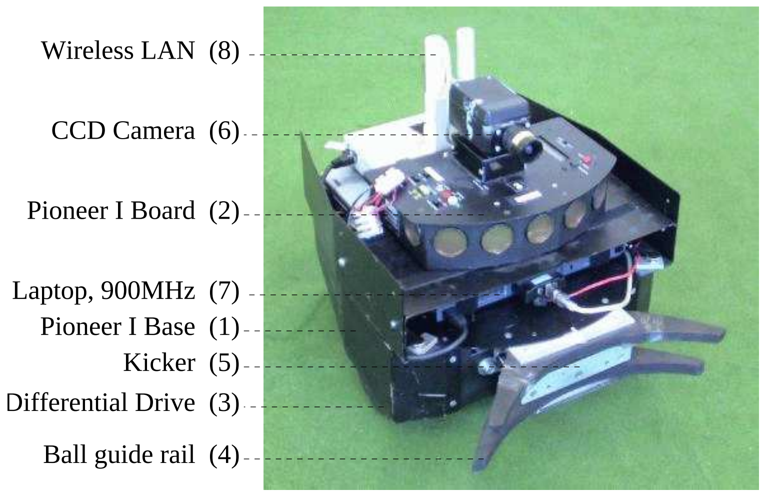

Fig. 1. An AGILO soccer robot

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

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. 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. Employing sophisticated probabilistic reasoning techniques and exploiting the cooperation between team mates, the robot can estimate complex game states reliably and accurately despite incomplete and inaccurate state information.

The state estimation system [16, 15] consists of the perception subsystem, the state estimator itself, and the belief state. The perception subsystem itself consists of a camera system with several feature detectors and a communication link that enables the robot to receive information from other robots. The belief state contains a position estimate for each dynamic task-relevant object. The estimated positions are also associated with a measure of accuracy, a covariance matrix. The state estimation subsystem consists of three interacting estimators: (1) the self localization system [11], (2) the ball estimator [12, 13], and (3) the opponents estimator. State estimation is an iterative process where each iteration is triggered by the arrival of a new piece of evidence, a captured image or a state estimate broadcasted by another robot. Detailed information about the computer vision and probabilistic state estimation techniques can be found in [15].

The game state estimator that will be employed by the AGILO RoboCuppers 2004 control systems features several substantial improvements and novel capabilities. For the first time we employ probabilistic observation models that are learned from ground truth data. Second, the robots will be more cognizant about the properties of the belief state, whether their self-localization is inaccurate or ambiguous, whether distinctive landmarks are in view, etc. Inferring these properties of belief states will result in more useful information for robot control and better performance of cooperative state estimation. The third extension will be novel methods for active state estimation. Here, the information about the belief states are used for selecting actions that reduce the entropy of the belief state along different dimensions. For example, if the situation allows for the localization of a lost robot by looking at this robot, the state estimation proposes that the other two robots face the lost robot in order to reinitialize the lost robot's self localization. Other active state estimation routines will intelligently search for the ball, etc.

Our medium term goals in state estimation that we will tackle after this year's competition are methods for perceiving and estimating game situations rather than belief states with accurate robot positions. An example of such a situation is an opponent robot in ball possession directly threatening our goal or an open area in front of the opponents' goal that is appropriate for goal shots. We believe that the perception of such system will provide better information to the action selection and will be more robust to compute because lower data accuracy is required.

Another direction of further research is to add capabilities for partial state estimation. Instead of relying on global position estimates, the robot system should be capable of acting based on partial state estimation such as only the ball is seen directly in front of me, etc. How to switch from action selection based on global state information to the one based on partial information and how to exploit partial state estimates as evidence for a global estimate are open questions on our research agenda.

4 The Robot Learning Language (ROLL)

Although learning techniques have become powerful and successful tools for the development and adaptation of robot control programs, a control task such as "win a soccer game in mid-size robot soccer" is far too complex to be solved as a monolithic learning task. Therefore the action selection component of the AGILO RoboCuppers employs a variety of smaller and diverse learned routines that solve, among other ones, the following reasoning tasks: 1. What happens if the voltage of the left motor is set to 2 volt? 2. Which control signal must be issued to reach the position (-2.0, 1.0) with an orientation of 96°? 3. How long will it take to get to the ball? 4. How promising is an attempt to dribble the ball into the goal in the current situation?

The action selection module of the AGILO RoboCuppers 2004 is a rational reconstruction and complete re-implementation of the action selection module used in earlier versions (see [9, 1, 10]). In the earlier versions of the action selection subsystem the learning tasks were solved in isolation and only the shared object files of the learned routines became part of the system. This yielded a number of severe drawbacks that made the development and improvement of the system very tedious and fragile.

Thus in the hot phases of preparation for RoboCup competitions, the learning processes were not properly documented. So afterwards it was impossible to find out the parameterization of the learning algorithms or which experiences had been employed. Thus, the solutions of a learning problem was irreproducible. This caused tremendous waste of resources when hardware components in the robots were replaced. The previously learned routine could not be used any more, a new one had to be learned from scratch and the same difficulties had to be solved anew.

To avoid these problems, in the new version of the action selection module the control tasks and learning problems are stated explicitly and declaratively. This is possible because we have added new execution mechanisms to the interpreter of our robot control system that make learning problems executable. Providing declarative and explicit specifications of learning problems and making these specifications operational supports the rigorous design of learning mechanisms and their transparent integration into the robot control systems.

Let us introduce the key concepts of ROLL using our second task from above as an example, that is the task of reaching a given position and orientation as fast as possible. A vital concept in our system is the notion of “experience”. Experiences are represented as first class objects and form the basis of a learning process. The first thing a robot has to do is to make “raw experiences”. He achieves this by following certain guidelines provided by a “problem generator”, how to explore the field. These raw experiences are then transformed into abstract experiences to make them suitable for learning. Furthermore the experiences are filtered in order to eliminate contradictory or undesired experiences.

In our example the raw experiences are composed of the x- and y-coordinates and the orientation angle with respect to the x-axis. As this representation is not suitable for learning. Therefore we chose an abstract experience representation containing the distance of the start and goal points and the turning angle of the robot with respect to the connection line of the two points. When two similar experiences exist, only the one is maintained where the robot reaches its goal faster.

The abstract experiences are now given to a learning algorithm that generates an executable function which can be integrated into the control system of the robot.

For the implementation of our learning constructs we extended the robot control language RPL. The whole system is constructed in an object oriented style, so that certain parts of a learning problem specification can be reused for other problems by exploiting inheritance mechanisms. Our approach has been described in more detail in [6] and [3].

Because of the complexity of our problems we learn them in simulation. For this purpose we have a simulator [8] that models the dynamic of the environment very accurately. Soon we are going to extend its functionality by adding models of our state estimation module (see section 3).

5 Using Models of Skills to Optimize Plans

The core of the action selection module is a set of skills common to the RoboCup domain. In our previous system, these skills were selected based on a hand-coded rule-based policy. We are converting these skill selection rules into more generic goal selection rules. Based on the current belief state (*"ball in own goal area"*, *"free path to goal"*), a goal is chosen (*"get ball out of own goal area"*, *"score goal"*). The robot then selects the appropriate skill(s) to achieve this goal. In order to do this we now use more complex models of our skills. They consist of procedural knowledge, declarative knowledge, and projection functions.

The procedural knowledge specifies how the skill should be executed. It can be hand-coded, a set of if-the-else rules, a state chart, learned through Reinforcement Learning, or any other method, as long as it provides a primitive action for each belief state. It is also known as the policy. In many approaches, a skill consists only of this component.

Declarative knowledge about skills consist of pre- and postconditions. In the action selection module we do not specify high-level goals such as *"win the game"*, or *"defend"*, but intermediate goals such as *"get ball out of own goal area"*, *"score goal"*. A skill that has this goal as its postcondition will be selected, as it can achieve this goal. If there is no such skill, a light-weight planning module constructs a plan that combines skills to achieve this goal (*"go to ball"*, then *"dribble ball into goal"*), using the pre- and postconditions specified in the declaritive component of the skills. The plan is executed as long as the preconditions for the plan still hold.

Projection functions allow us to make predictions about what will happen if we execute a skill. For instance, how long it will take to execute the task given a initial and goal-state? What is the chance of succeeding? This kind of knowledge is acquired by extensively executing the skill in a simulator, and recording the results. Neural networks are used to learn the mapping from $state_{init} \times state_{goal} \rightarrow time$. We have used this approach before [9], and it has been very successful in coordinating our robots, by letting every robot predict how long it will take it to get the ball. ROLL, described in section 4, automates this learning process.

This approach combines the benefits of planning and learning. Pre- and postconditions allows the programmer to express action selection constraints in a declarative and transparent way. A light-weight planner then generates a plan using these conditions. As valid plans must not always be optimal, we use learned projection functions of skills to optimize them. Combining planning and projection functions has been researched in the context of Reinforcement Learning [14], and hall-way navigation [7], but as far as we know, not in the RoboCup domain.

6 Empirical Evaluation

Realizing an autonomous robot control system is a very peculiar programming task that differs enormously from programming tasks in other application domains. While in most programming tasks we aim at programs that *compute the correct results* or *do the right thing*, in robot control we aim at programs that produce the *optimal perception driven behavior*. This difference is so big because the behavior does not only depend on the control program but also on the perception capabilities, the physical dynamics of the robot, system parameterization and the assignment of computational resources to computational tasks. Thus the very same programs may exhibit very different behaviors even if the influencing conditions are varied slightly.

For example, in the AGILO RoboCuppers control system the behavior of the robots critically depends on how computational resources are assigned to state estimation tasks and the remaining tasks. Or a variation of the camera model might cause inaccuracies in self localization and thereby cause problems in the role assignment within the team. Such interactions cannot be found and understood by looking at the program code. Rather, we have to identify the control parameters of our system and the context conditions that system performance depends, build a causal model of system behavior, and then learn the parameters of the causal model from empirical studies.

For this year's competition we have extended the software development environment for the AGILO RoboCuppers in two important ways. First, we have developed a camera system consisting of two cameras mounted at the ceiling, a visual marker system that allows us to recognize the individual robots on the field, and a global state estimation routine for tracking the positions and orientations of the ball and each robot on the field. This system provides us with ground truth data about the physical behavior of each robot where the expected accuracy of position data is about two centimeter in the center and about 4-5 centimeters at the boarders of the playing field. In addition, we have substantially extended our on-board logging mechanisms that give use very detailed information about the beliefs of the robot and the evidence the beliefs are based on at any time instance during the game. This setup is a huge source of data for the empirical study of our system.

The second extension is new workbench for the empirical study of system and agent behavior in simulated robot soccer, mid-size robot soccer, and human soccer [5] This workbench consists of a relational database system that stores the empirical data, an interpreter for recognizing and classifying behaviors based on observable features, and for data mining and visualization of empirical findings.

7 Sharing Belief within a Mixed Team

Due to scientific as well as pragmatic reasons, there is a growing interest in the robotics field to join the efforts of different labs to form mixed teams of autonomous mobile robots. As most robots in the F2000 league are custom built, or at least customized commercial research platforms with unique configurations of actuator and sensor configurations, mixed teams from different laboratories are extremely heterogeneous. This makes the unification of the software of the different robots of a potential mixed team almost impossible without substantial rewriting of at least one of the team's software.

Together with the teams of the University of Ulm (Ulm Sparrows) and the Technical University of Graz (Mostly Harmless) we have designed and implemented a communication framework for sharing information within a team of extremely heterogeneous autonomous robots, allowing us to play interchangeably with the different robot platforms in a mixed team [18].

The communication framework is hardware and software independent, and can extend existing software architectures in a transparent way. Major code rewrites are not necessary to use this framework. The team communication uses a message-based, type safe high-level communications protocol that is transfered by IP-multicast. The implementation uses the notify multicast module (NMC) of the Middleware for Robots (MIRO) framework [17]. MIRO is a CORBA based middleware architecture for autonomous mobile robots.

For cooperation between robots, the sharing of information about beliefs concerning the state estimation is initially sufficient for successful cooperation. At the moment we therefore communicate only an expressive belief state using this framework; explicit role assignment or coordination is not part of the communication.

8 Summary

In this paper we have given an overview of the AGILO RoboCuppers team. After having discussed the main research directions in section 1, the hardware platform was presented in section 2. In the next three sections on vision-based, cooperative game state estimation (3), the robot learning language (4), and using models of skills to optimize plans (5), we have explained how we aim to implement semi-autonomous acquisition of visuo-motoric skills. In section 6 we have briefly discussed our ground-truth system, and our workbench for the empirical study of system and agent behavior in simulated robot soccer, mid-size robot soccer, and human soccer. Our cooperation with two other universities to design and implement a communication framework that allows platform independent exchange of belief states, and therefore mixed team coordination, was presented in section 7.

References

1. M. Beetz, S. Buck, R. Hanek, A. Hofhauser, and T. Schmitt. AGILO RoboCuppers 2002: Applying Cooperative Game State Estimation Experience-based Learning, and Plan-based Control to Autonomous Robot Soccer. In *RoboCup International Symposium 2002*, Fukuoka, 2002.
2. M. Beetz, S. Buck, R. Hanek, T. Schmitt, and B. Radig. The AGILO Autonomous Robot Soccer Team: Computational Principles, Experiences, and Perspectives. In *First International Joint Conference on Autonomous Agents and Multiagent Systems*, Bologna,Italy, 2002.
3. M. Beetz, A. Kirsch, and A. Müller. Rpl-learn: Extending an autonomous robot control language to perform experience-based learning. In *submitted to 3rd International Joint Conference on Autonomous Agents & Multi Agent Systems (AAMAS)*, 2004.
4. Michael Beetz, Thorsten Schmitt, Robert Hanek, Sebastian Buck, Freek Stulp, Derik Schröter, and Bernd Radig. The AGILO 2001 robot soccer team: Experience-based learning and probabilistic reasoning in autonomous robot control. *accepted for publication in*

Autonomous Robots, special issue on Analysis and Experiments in Distributed Multi-Robot Systems, 2004.

5. Michael Beetz, Thomas Stammeier, and Sven Flossmann. Motion and episode models (simulated) football games: Acquisition, representation, and use. submitted to 3rd International Joint Conference on Autonomous Agents & Multi Agent Systems (AAMAS).
6. Michael Beetz, Freek Stulp, Alexandra Kirsch, Armin Müeller, and Sebastian Buck. Autonomous robot controllers capable of acquiring repertoires of complex skills. In *RoboCup International Symposium 2003*, Padova, July 2003.
7. T. Belker, M. Beetz, and A.B. Cremers. Learning action models for the improved execution of navigation plans. *Robotics and Autonomous Systems*, 38(3-4):137–148, 2002.
8. S. Buck, M. Beetz, and T. Schmitt. M-ROSE: A Multi Robot Simulation Environment for Learning Cooperative Behavior. In *H. Asama, T. Arai, T. Fukuda, and T. Hasegawa (eds.): Distributed Autonomous Robotic Systems 5, Springer*, 2002.
9. S. Buck, M. Beetz, and T. Schmitt. Reliable Multi Robot Coordination Using Minimal Communication and Neural Prediction. In *M. Beetz, J. Hertzberg, M. Ghallab, and M. Pollack (eds.): Advances in Plan-based Control of Autonomous Robots. Selected Contributions of the Dagstuhl Seminar "Plan-based Control of Robotic Agents", Lecture Notes in Artificial Intelligence, Springer*, 2002.
10. S. Buck, U. Weber, M. Beetz, and T. Schmitt. Multi Robot Path Planning for Dynamic Environments: A Case Study . In *Proc. of the IEEE Intl. Conf. on Intelligent Robots and Systems*, 2001.
11. R. Hanek and T. Schmitt. Vision-Based Localization and Data Fusion in a System of Cooperating Mobile Robots. In *Proc. of the IEEE Intl. Conf. on Intelligent Robots and Systems*, pages 1199–1204. IEEE/RSJ, 2000.
12. R. Hanek, T. Schmitt, S. Buck, and M. Beetz. Towards RoboCup without Color Labeling. In *RoboCup International Symposium 2002*, ¡B¿award-winning paper¡/B¿, Fukuoka, Japan, 2002.
13. Robert Hanek, Thorsten Schmitt, Sebastian Buck, and Michael Beetz. Towards RoboCup without color labeling. *AI Magazine*, 2002. to appear.
14. Malcom R.K. Ryan. Using abstract models of behaviours to automatically generate reinforcement learning hierarchies. In *Proceedings of The 19th International Conference on Machine Learning, Sydney, Australia*, July 2002.
15. T. Schmitt and M. Beetz. Designing Probabilistic State Estimators for Autonomous Robot Control. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2003.
16. Thorsten Schmitt, Robert Hanek, Michael Beetz, Sebastian Buck, and Bernd Radig. Cooperative probabilistic state estimation for vision-based autonomous mobile robots. *IEEE Transactions on Robotics and Automation*, 18(5):670–684, October 2002.
17. Hans Utz, Stefan Sablatng, Stefan Enderle, and Gerhard K. Kraetzschmar. Miro – middleware for mobile robot applications. *IEEE Transactions on Robotics and Automation, Special Issue on Object-Oriented Distributed Control Architectures*, 18(4):493–497, August 2002.
18. Hans Utz, Freek Stulp, and Arndt Mühlenfeld. Sharing belief in teams of heterogeneous robots. submitted to RoboCup International Symposium 2004, July 2004.