THE PADERKICKER TEAM: AUTONOMY IN REALTIME ENVIRONMENTS

Willi Richert, Bernd Kleinjohann, Markus Koch, Alexander Bruder, Stefan Rose, and Philipp Adelt
Faculty of Computer Science, Electrical Engineering and Mathematics, University of Paderborn, Germany
richert@c-lab.de

Abstract: The Paderkickers are a robot soccer team that makes heavy use of automotive technology like C167 micro-controllers or communication over CAN bus. All sensor data is processed on these decentralized embedded nodes to yield a high degree of reliability and hardware layer abstraction. In this paper, we describe how the complex system copes with perception and action in real-time and integrates it in the higher strategy layer to achieve autonomous behavior.

1. INTRODUCTION

The Paderkicker team [8] consists of five robots (Fig. 1) that already participated successfully in the German Open competition in 2004 and 2005, and the Dutch Open 2006. Currently, last preparations for the RoboCup 2006 World Championships are in progress.

Our platform asks for the whole range of research issues needed for a successful deployment in the real world. This includes embedded real-time architectures [2, 5, 14–17], real-time vision [2, 14–17], learning and adaptation from limited sensor data, skill learning and methods to propagate learned skills and behaviors in the robot team [13, 12, 9]. However, our goal is not to carry out research for specific solutions in the robotic soccer domain, but to use and test advanced techniques from different research projects. The Paderkicker platform serves as a test bench for the collaborative research center 614 (funded by the Deutsche Forschungsgesellschaft). Furthermore, the knowledge in vision, motion and object tracking is currently used in the AR PDA (Bundesministerium für Bildung und Forschung) project [11].

Figure 1. The Paderkicker team.

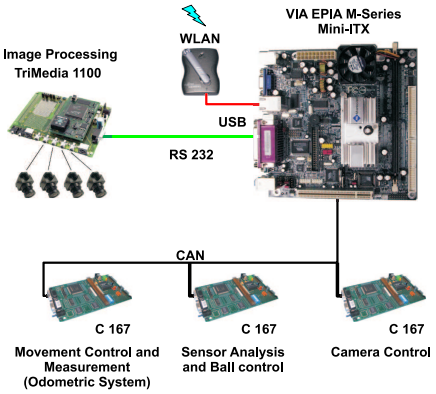

Figure 2. The Paderkicker architecture.

2. ROBOT OUTLINE

The robots (Fig. 1) have differential drive (2*75W) allowing for a maximum speed of 2.5 m/s. Instead of omni-vision we use four standard cameras for improved recognition of far away objects. As the colored marker objects at the corner will be removed in the near future the robots will have to distinguish far away line markers. We use a slide driven by an elastic band to shoot the ball. For the proper positioning of the ball prior to shooting the robot can use two side wise rolls and one above the ball, all of them electrically driven in both directions if needed. The central processing unit in our robot architecture (Fig. 2) is a PC compatible board which boots real-time Linux. The main process on the PC is Paderkicker's *Brain* module written in Java, which is the central instance to process the accumulated perception and choose the correct actions. This is possible, because all time-consuming processing is sourced out to the diverse distributed controllers. The system still has very low processor load. The Java process is assisted by a Particle Filter process (C++) which sits between the vision sensor and *Brain*, calculating the global robot position, relative ball position, and obstacles. Thereby, we have two ball positions the behavior system can operate with: a fast inaccurate one for approaching the near ball for better reactivity and a delayed, accurate and more stable one for moving towards the ball that is more than 2,5m away. Furthermore, *Brain* connects to a separate Jess [7] process, a rule engine similar to CLIPS but written in Java. The Jess instance has the task to switch between strategies (Attack, Defend,

etc.) dependent on team variables like e.g. "our team possesses the ball" or "enemy in goal area".

2.1 HARDWARE

Paderkicker robots make use of automotive technology like C167 16bit micro controllers and communication over CAN bus. Furthermore dedicated hardware is used for image processing. Self designed PCBs are integrated for voltage and temperature control. All incidental sensor data is processed on these decentralized embedded nodes for better reliability and hardware layer abstraction. The extracted data is reported to the central processing node, a Mini-ITX board with a VIA EPIA 1GHz CPU running real-time Linux.

Research is done mainly in the area of real-time image processing. An optimized algorithm for low latency real-time color segmentation [16] which even performs well on a PDA is implemented on a Trimedia TM 1100 video processing board running at only 100 MHz. Four rotatable analog cameras are used to cover the whole 360° view instead of omnivision resulting in an overall higher resolution. Hence, we are capable of seeing distant objects much better. Higher visual coverage of the environment will become more important in the near future when the field will likely be extended in size. All four cameras are connected to the Trimedia Board. The data stream of one camera is evaluated at a time and the extracted objects are delivered at a speed of up to 20 fps over RS232 to the Mini-ITX board. Additionally, object recognition and tracking [2] are a prerequisite for accurate self localization. Combining our real-time color segmentation algorithm and the implemented edge vectorizer [17] we can easily feed our real-time particle filter [10] with the crucial data needed to provide our world model with the absolute robot position.

The central node of our hardware layer is the Mini-ITX board running a Timesys Linux kernel (2.6 series) on a linux system built from scratch. The board is connected to the robot's actuators via the CAN bus. With three C167 boards we handle the drive control, odometry, ball control (rolls), and camera positioning. Voltage and temperature sensors are read out over USB. The base system is placed on a compact flash card, mutable configuration data resides on a USB stick. Finally, we use a WLAN USB module to communicate with a central server that routes the messages to the proper robot peer. With this hardware architecture, the system is able to provide the behavior and strategy level with all the necessary information it needs.

2.2 SOFTWARE

Our software behavior system breaks down into four independent modules (Fig. 3):

World model module We use particle filters for every robot and communication between the team members to update their world state, that includes the state of the robot and some features from the team members.

Strategy module Depending on the world model the strategy module decides the current strategy independently for every robot. Strategies include e.g. *Defend*, *Attack*, but also standard situations like *Kick off* or *Penalty*. This is done by the aforementioned rule engine (cf. Section 3.1).

Tactics module Every strategy is realized by a finite state machine that carries out the tactics. Its states are, e.g., *Ball facing*, or *Stay between goal and opponent*.

Behavior module This module executes the actual actions. It is behavior based in terms of Arkin's Motor Schemes [1]. Our behavior system [4] allows for a distinction between cooperative and competitive behaviors and behavior control through time excited evaluation functions. It consists of a set of low-level behaviors that have to be combined in order to result in a vector that can be sent to the actuators.

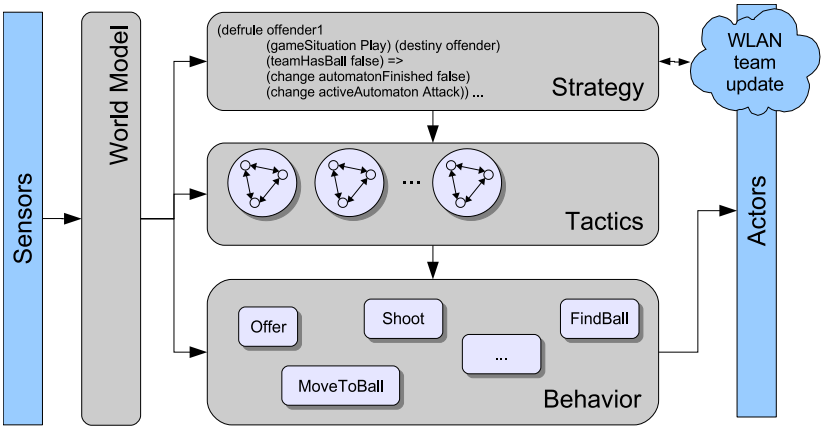

Figure 3. Information flow in the Paderkicker.

Every major tactic like attacking the opponent or defending the own goal has been modelled as a separate automaton. The selection of the

proper automaton is the task of a rule engine implemented in Jess that keeps track of robot and team state changes (Fig. 3). In this way, the RoboCup soccer rules could be implemented very quickly and we could concentrate on the behavior details.

2.3 THE MESSAGE FORMAT: CONNECTING HARDWARE AND SOFTWARE

With the decentralized approach the systems autonomy is delegated to the dedicated hardware boards, every board has one task to accomplish and is responsible for it. This leads to a more complex information exchange mechanism needed to provide all subsystems with needed information in time. In addition, there are different bus systems to pass information over — CAN bus, RS 232, USB and TCP/IP for communication between robots. For this domain, a special message format has been designed that can be used on all bus systems in the Paderkicker architecture (Fig. 4). This message format turned out to be robust and

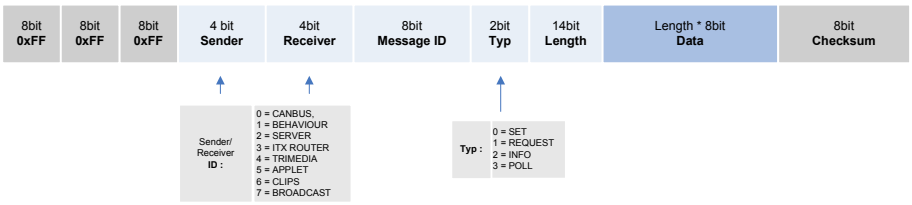

Figure 4. The message format.

computationally cheap. Every message is preceded by three bytes `0xFF` as a message start delimiter. Thereby, every subsystem is able to figure out the starting point of messages in continuous data streams with low processing overhead. With the *Message ID* field it is possible to extend the current message set by not yet foreseen message possibilities.

3. TEAM OUTLINE

The mission of the RoboCup project is to "develop a team of fully autonomous humanoid robots that can win against the human world champion team in soccer" by 2050 [3]. To decrease the gap of robots ability today and the mentioned honor future goal we have to focus on different substantial skills. A robot on one hand must be capable of acting fully autonomously under a dynamic environment and on the other hand needs to cooperate with other team members to have a chance to

win a soccer game. We will focus on these two important points on the layer of strategy. The complexity of such a project will spread quickly to different other problems like a union world model, exchange of perceptions between teammates, role negotiation, to play by the competition rules and to handle uncertainty. The strategy level of both bases on the rule-based system Jess [7] which is used as an expert system for RoboCup and interacts closely with other components of the robot and even over the whole team.

3.1 STRATEGY LAYER

For the strategy layer of the Paderkicker we use a rule-based system for tactics on a higher level of the robotic architecture. The planning on that layer is very intuitive by using expert system declarative programming methodology. So the expert knowledge is well human readable and acts in a manner of what's to be solved and not procedural how this should happen. We prefer to program like first mentioned and rely on the ability of the expert system to reason even under incomplete world information [6]. First we want to mention the interaction between the rule-based system and the other components of the robotic software architecture.

Architecture and Interfaces. The overall rule-based system architecture is shown in Fig. 5. Beginning at the left side of the figure the expert system core architecture is shown. It consists of different modules. The working memory holds the facts and variables of the actual world model (Here: It is unknown if the ball is in the perception range of the robot). The rule-base holds the domain specific expert knowledge coded into rules. (Here: If the ball is in range then change the role of the robot to offender.) The pattern matcher matches the rule premises against the facts in working memory and creates an agenda for execution of the activated rules. Now the interface to the rest of the robot comes into account on the right side of the figure. The rule-based system is fed for example by the perception module of the behavior system. The perceptions are preprocessed and appended into the working memory as facts independently of the inference machine. This is solved by an observer in Java. For example if the perception system recognizes a ball in range then the observer reflects this as a fact into the working memory. The way back to the behavior system and later on to the actuators is done by extending Jess with so called User-Functions which implement a Transmitter to the behavior system. In our example if the ball is in range the rule will fire and change the role of the robot via the change-transmitter to offender. This closes the circle of interaction.

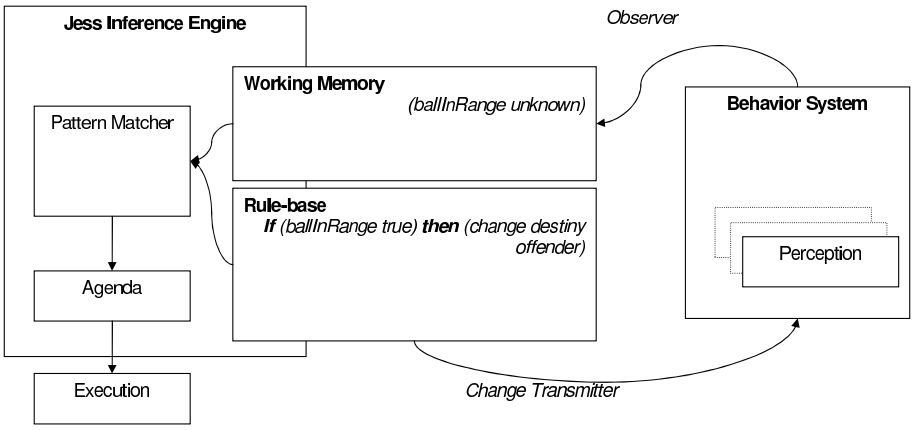

Figure 5. The rule engine allows for autonomy at the strategy level.

In future enhancements we think about using the capabilities of shadowed facts in Jess, which are automatically updated by core rule system mechanism. The way from the expert system to the lower layers is done by extending the Jess functionality by special user-functions which transmits the automation activation and other commands.

Teammate Knowledge. The teammate knowledge base holds the facts and rules which are relevant for one autonomous robot on its own. The rule-base reasons on states of the robot world that are mapped to facts of the expert system. In addition, it reasons on requirements which can be created inside the expert system itself or outside, like other team members or a team server. The interaction with a team server is particularly important for mixed teams like the mixed team of Paderkicker and Tech United participating in the world championship in Bremen this year. The teammate expert system activates and monitors the different tactics (finite state machines) which are required to handle typical situations in RoboCup.

Team Knowledge. There are different ways to form a team out of individual soccer robots. One way is to create one dedicated server process that distributes roles and commands to the teammates. Another way is to design the expert system in a distributed way, so that every robot has the same rule base and there are special rules for team decisions that every teammate makes an independent decision in a given situation. We started with the second solution, implementing our team

play as a distributed expert system. That way, we have to handle a lot of special rules for rule assignment and additionally have to hold every piece of information redundant in every database. For the RoboCup world championship in Bremen we need to build a mixed team where both teams — the Paderkickers and TechUnited Eindhoven — have completely different hard- and software. This led to the introduction of a team server that holds a separate Expert System for cooperation of robots from different universities. The team server acts on a coach level like in real soccer games. The different robots register at the team server and transmit facts which are important for a coach. Examples of these facts are the robots' positions and the position of the ball on the playground.

3.2 CONNECTING TEAMS

In order to be able to connect with other soccer robot teams to form one team, a team server serves as a central point, featuring a simple and easy-to-adopt team message format. This was necessary, since other teams should not have to bother with the message format peculiarities that stem from our robots' unified communication including CAN-bus, a serial line, etc. The new team message format thus only supports messages that are dedicated to inter-team communication, world model exchange und routing referee commands between the different teams. It is now being evaluated with TechUnited, the Robocup midsize league team of Eindhoven and Delft.

4. OUTLOOK

At the moment we are redesigning our platform (Fig. 6) having now omni-wheels, allowing for better vision capabilities and supporting inter-team connection for building mixed teams.

Concerning the demanding needs for line detection and higher frame rates the Trimedia processor and the RS232 connection are bottlenecks. Therefore current research takes place in the field of image processing on FPGA where color segmentation and line detection is to be integrated to deliver extracted 2D features easily at speeds of more than 50 fps. A self designed PCB with a FPGA and USB as well as the image processing circuit meet these needs and will be integrated in the near future.

Another purpose of our Paderkicker soccer team is the investigation of appropriate means to propagate learned knowledge in teams of robots [13, 12, 9]. Currently, we develop a framework that enables robots in a team to find their most natural skills. As every robot has a different perception stream the sensorimotorical couplings will be learned much

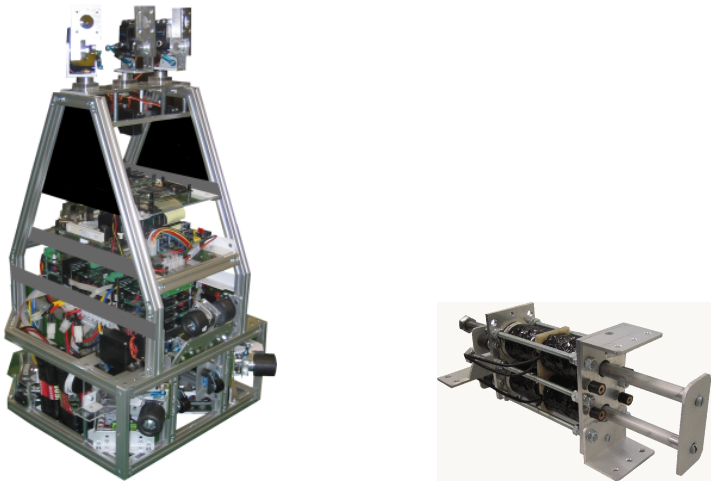

Figure 6. The new platform with a more robust chassis, better vision capabilities, omnidirectional drive, a stronger shooting device (right side), and a software architecture that supports inter-team cooperation.

faster by decentralizing the "babbling phase", in which they find out basic behaviors by trial and error. These can then be propagated to other team members.

5. CONCLUSION

With the platform described in this paper we achieve the two conflicting goals in the soccer domain needed to reach true autonomy: quick response rates for high reactivity and more complex but less frequent deliberation processes. Components that are subject to quick changes in the environment are arranged in a decentralized manner and are located on specialized hardware. Those processes that do not need that fast update cycles like team communication, planning or processing at the higher levels are located at the Mini-ITX, allowing for faster development cycles but slower execution time. This architecture has evolved naturally out of the needs to combine a fault tolerant embedded system with fast development cycles for behavior exploration.

REFERENCES

[1] R. C. Arkin. Motor schema based navigation for a mobile robot: An approach to programming by behavior. In *Proceedings of the IEEE Conference on Robotics and Automation*, 1987.

[2] D. Beier, R. Billert, B. Brüderlin, Bernd Kleinjohann, and Dirk Stichling. Marker-less vision based tracking for mobile augmented reality. In *Proceed-*

ings of the Second International Symposium on Mixed and Augmented Reality (ISMAR 2003), 2003.

[3] H.-D. Burkhard, D. Duhaut, M. Fujita, P. Lima, R. Murphy, and R. Rojas. The road to robocup 2050. *IEEE Robotics and Automation Magazine*, 9(2):31–38, 2002.

[4] Natascha Esau, Bernd Kleinjohann, Lisa Kleinjohann, and Dirk Stichling. MEXI - machine with emotionally extended intelligence: A software architecture for behavior based handling of emotions and drives. In *Proceedings of the 3rd International Conference on Hybrid and Intelligent Systems (HIS'03)*, 2003.

[5] Natascha Esau, Bernd Kleinjohann, Lisa Kleinjohann, and Dirk Stichling. Visitrack - video based incremental tracking in real-time. In *6th IEEE International Symposium on Object-oriented Real-time Computing (ISORC '03)*, 2003.

[6] Ernest Friedman-Hill. *Jess in Action : Java Rule-Based Systems (In Action series)*. Manning Publications, December 2002. ISBN 1930110898.

[7] Ernest Friedman-Hill. Web site for the software Jess, 2005. http://herzberg.ca.sandia.gov/jess/.

[8] Bernd Kleinjohann. The Paderkicker Team, 2006. http://paderkicker.upb.de.

[9] Markus Koch, Willi Richert, and Alexander Saskevic. A self-optimization approach for hybrid planning and socially inspired agents. In *Second NASA GSFC/IEEE Workshop on Radical Agent Concepts*, 2005.

[10] Cody C. T. Kwok, Dieter Fox, and Marina Meila. Real-time particle filters. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *NIPS*, pages 1057–1064. MIT Press, 2002. ISBN 0-262-02550-7.

[11] Christian Reimann. Kick-Real - a mobile mixed reality game. In *ACE2005, ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, 2005.

[12] Willi Richert, Bernd Kleinjohann, and Lisa Kleinjohann. Evolving agent societies through imitation controlled by artificial emotions. In M. Huang, X.-P. Zhang, and M. Huang, editors, *ICIC 2005*, number 3644 in LNCS, pages 1004–1013. Springer-Verlag Berlin, 2005.

[13] Willi Richert, Bernd Kleinjohann, and Lisa Kleinjohann. Learning action sequences through imitation in behavior based architectures. In *Systems Aspects in Organic and Pervasive Computing - ARCS 2005*, number 3432 in LNCS, pages 93–107. Springer-Verlag Berlin, 14 - 17 March 2005.

[14] Dirk Stichling. *VisiTrack - Inkrementelles Kameratracking für mobile Echtzeitsysteme*. PhD thesis, Universität Paderborn, Fakultät für Elektrotechnik, Informatik und Mathematik, 2004.

[15] Dirk Stichling and Bernd Kleinjohann. CV-SDF - a model for real-time computer vision applications. In *IEEE Workshop on Application of Computer Vision*. IEEE, December 2002.

[16] Dirk Stichling and Bernd Kleinjohann. Low latency color segmentation on embedded real-time systems. In Bernd Kleinjohann, K.H. Kim, Lisa Kleinjohann, and Achim Rettberg, editors, *Design and Analysis of Distributed Embedded Systems*. Kluwer Academic Publishers, 2002.

[17] Dirk Stichling and Bernd Kleinjohann. Edge vectorization for embedded real-time systems using the CV-SDF model. In *Proceedings of the 16th International Conference on Vision Interfaces (VI 2003)*, June 2003.