Hierarchically Distributing Embedded Systems for Improved Autonomy

Claudius Stern, Philipp Adelt, Willi Richert, and Bernd Kleinjohann

Abstract Distribution of functionality among nodes is a contemporary research issue for embedded systems, e.g. in the field of autonomous mobile robot groups. In such groups, the concept of distribution is mainly used to achieve flexibility and robustness that could not be reached by a single robot. Here it will be used as a design-paradigm for a robot's *internal* architecture. In this paper, a hierarchically distributed robot architecture will be introduced which leads to an improved autonomy of the overall system.

Key words: distributed embedded systems, autonomous systems, robot control

1 Introduction

Distribution is a contemporary research issue for embedded systems. Currently, in the field of distributing a task to a group of nodes much research work is done (e.g. [2, 15]). By distributing a task, the probability for a system-wide failure or for continuously incorrect execution decreases because of the diversity and independency of the distributed system parts.

In robotic soccer, which is a challenging research and application field for the combination of real time embedded systems design with intelligent autonomous behavior, distribution is applied at two levels. At the multi-robot level the paradigm of distribution is mainly used to build homogeneous cooperating teams of robots. At the level of single robots this paradigm can considerably support robustness and flexibility. That will be shown in this paper describing the Paderkicker robots.

The Paderkicker team [9] consists of five robots that already participated successfully in several international competitions including the RoboCup 2006 World

Claudius Stern · Philipp Adelt · Willi Richert · Bernd Kleinjohann
Universität Paderborn, C-LAB
e-mail: claudis, padelt, richert, bernd@c-lab.de

Championship. Our platform asks for the whole range of research areas needed for a successful deployment in the real world. This includes embedded real-time architectures [3, 5, 10], realtime vision [11, 12, 13], learning and adaptation from limited sensor data, skill learning [7] and methods to propagate learned skills and behaviours in the robot team [8].

In this paper we focus on the hierarchically distributed architecture of the Paderkicker robots. In Section 2 we describe the current system with its components. Section 3 then focuses on the modular system design of the Paderkicker robot, covering the functional design as well as the architecture of hardware. Section 5 gives an overview of the vision module which includes three individual real-time image processing modules whose outputs then are merged. The behavior module is described in Section 4. Section 8 concludes the paper and a short survey of future development directions and research fields is given regarding the architecture described in this paper.

2 Robot outline

The current generation of the Paderkicker robots is equipped with an omnidirectional drive which enables the robot to do translational and rotational movements simultaneously. This is a great advantage over the prior generation described in [9] that featured a differential drive with two driven wheels. Here a four wheel omnidirectional drive is used instead of a three wheel one. The construction of the wheel suspension ensures that all four wheels are pressed onto the ground which leads to enhanced stability.

Besides the driving system, the ball handling system has been redesigned from scratch. The ball handling system consists of two main components: the ball kicking system and the dribbling system. The previously used mechanical kicking system has been replaced by an electromagnetic solenoid which provides more control over the kicking power and reduces the actuation latency. The ball dribbling system has been redesigned to be more robust concerning collisions. All servo motors have been mechanically decoupled with rubber blocks so that even hard collisions will not harm the servos with excessive mechanical shocks.

The same mechanical decoupling has been applied to the servos of the active vision system to tolerate collisions with high kicked balls. In contrast to omnivision systems that are currently used by many other teams, three individual pan-tilt cameras are used in the vision system. Each camera may independently focus and track a different object of interest like ball, goal or other robots.

3 System design

In this section the structure of the Paderkicker robot will be shown. First, the functional architecture will be described. Then we will show how this logical structure maps onto a hardware structure. After the description of the underlying structures, the behavior system as well as the vision system will be introduced.

3.1 Functional architecture

During the system design process, four main functional units were identified (vision, driving, ball handling and behavior) and designed in a modular way. A robot of the Paderkicker team consists of a behavior module, the vision module, the driving module, and the ball handling module. The function of the last three is self-explaining by their respective names. The behavior module is the topmost module in a robot's hierarchy. It controls the robot's overall behavior.

The different modules are realized in a distributed way as described below. All components communicate with a message format which is used in the entire system independent of the respective medium for communication.

The functional units were further divided into sub-modules as depicted in Figure 1. This structure allows the independent development of the different functional units. Furthermore, the functional units were designed to work autonomously on their own presenting an already abstracted interface to the rest of the system. A dedicated interface sub-module manages the communication and merges data. This hierarchical structure enables the functional unit ”Behavior module” to act on a very high level of abstraction.

As an example, the driving module is designed to work autonomously and part of the robot's low-level behavior has been mapped to it. Distributing the drive-control task to a group of sub-modules instead of using only one monolithic module leads to more flexibility and robustness. The sub-modules within are realized on individual microcontroller boards working as a distributed system. Each microcontroller board realizes an individual motor controller and odometry data logger with a short measurement-control latency and therefore can react very fast. Each board also incorporates an emergency handling unit which leads to a more robust behavior of the whole driving module.

Other teams hardly describe their overall software architecture. Often they describe in detail the behavior system and its mechanisms but not the underlying overall structure. Nevertheless, a common approach is a layered structure, e.g. used by the 5dpo–2000 team [6] or the AIS–BIT Robots team [4]. The AIS–BIT Robots team uses two layers at different abstraction levels. The first layer deals with low-level processing of sensor data and image data. The second layer then deals with abstract behaviors. Both layers contain different modules but the modules within one layer are not further hierarchically arranged.

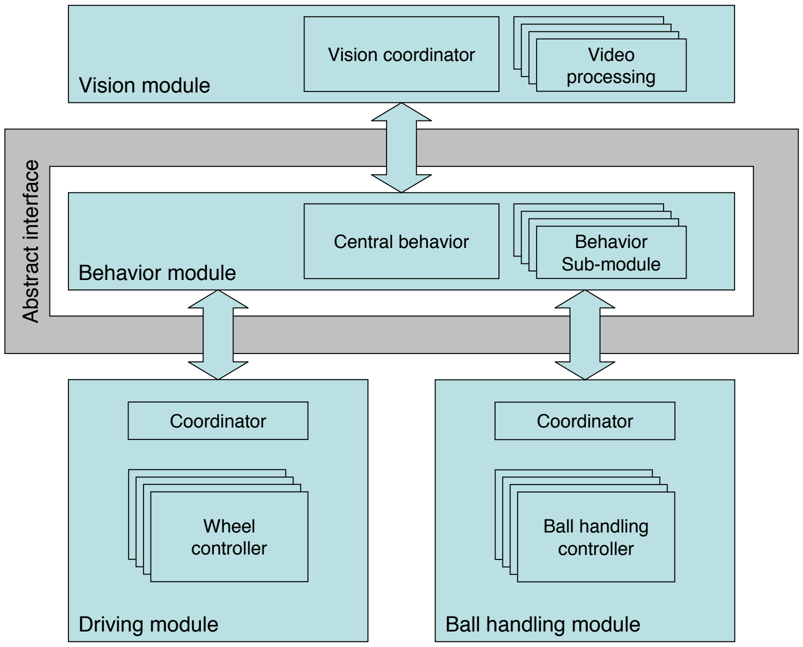

Fig. 1 Paderkicker hierarchical module structure

3.2 Hardware architecture

The functional structure described above is mapped onto a hardware architecture as depicted in Figure 2. The central processing unit is a Pentium M ULV PC board running Linux. The vision algorithms and the behavior system are realized here. The Mini-ITX board is equipped with a Mini PCI wireless LAN card and handles team communication.

As described above, the modules for ball handling and driving are divided into sub-modules. These sub-modules are realized on microcontroller boards equipped with an Atmel microcontroller which comes with an on-chip CAN bus interface. Groups of microcontroller boards communicate over CAN with the members of the according group. One dedicated microcontroller board in each group manages the communication with the central Mini-ITX board over a USB connection.

4 Behavior based system

The actual version of the behavior system is realized as a parallel distributed software system (Figure 3), where parallel running processes are responsible for the dedicated functional hardware units vision, driving, and ball handling. In addition,

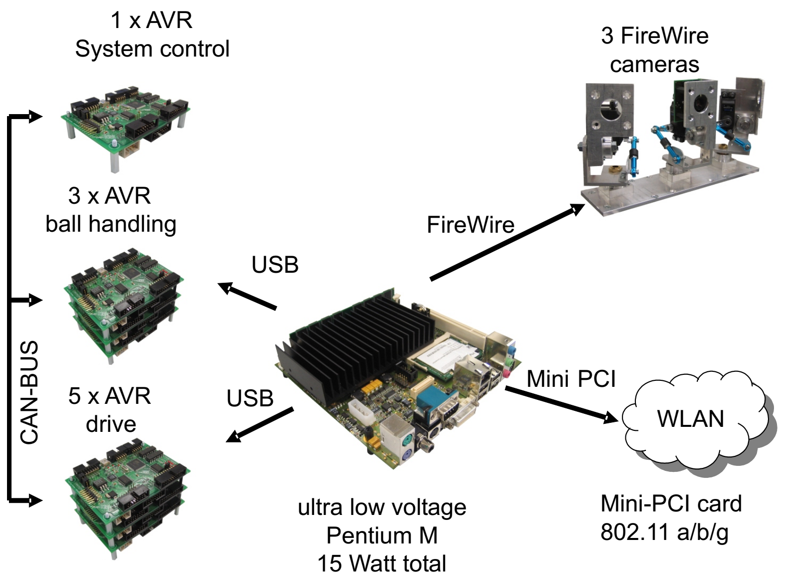

Fig. 2 Paderkicker hardware architecture

a new timing concept now allows the different subsystems like the above mentioned to run at different cycle duration. Using a double buffered shared memory approach it is no problem if e.g. the cycle time of the vision system increases because the analyzed image contains more detectable objects than usual or if the ball handling component has to run at a higher frequency than the driving component.

The architecture's design is driven by the need of the sub-modules *Active Vision*, *Driving*, and *Ball Handling* to run at different sample rates. In the former architecture all the functionality was done in the same module at the same speed. The problem was that functionality that needs to run at a high speed got at some point corrupted data from modules running at slower speed, which lead to unpredictable behavior in some cases. To avoid this, at first the different functionality was identified and regrouped in separate sub-modules. Then we introduced a double-buffered communication mechanism that separates the actual data on which the individual modules are working on from the communication process.

Each sub-module has its own *Sense-Plan-Act* cycle. The *Act* part is of course no real action but rather new data for the other modules or part of the final action which first has to be sent to the hardware via the *Router*. All sub-modules are running concurrently. While they are implemented at the moment as Java Threads it is no difficulty to let them reside on even different processors.

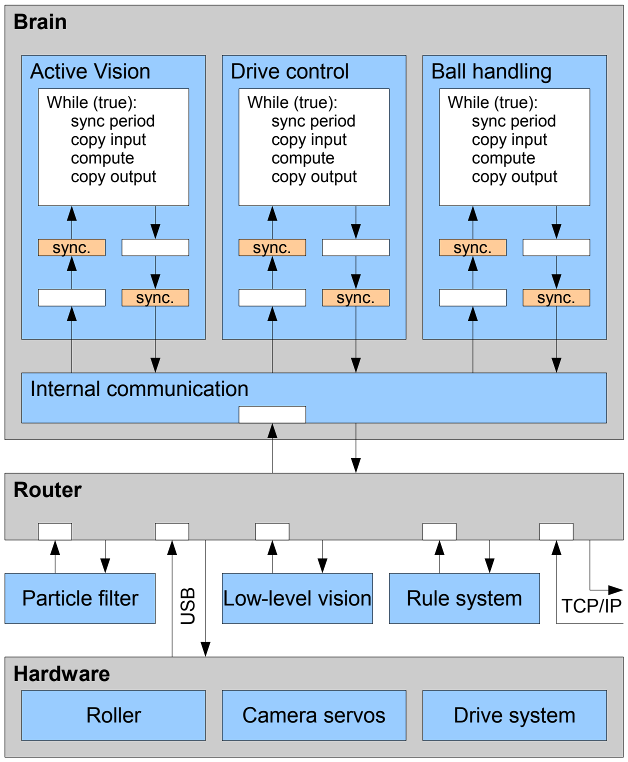

Fig. 3 Asynchronous architecture for the behavior module

5 Vision system

The vision system also has been designed using the paradigm of hierarchical distribution. The vision systems span four levels of abstraction, beginning with the low-level vision based on an optimized algorithm for low latency real-time color segmentation [12]. The original algorithm has been adapted to run on a PC under an ordinary Linux system. Three digital FireWire cameras are mounted on pan-tilt units to cover the whole 360° view. Each camera is handled by an independent task doing the low-level image processing. On the next level of hierarchy the outputs of these tasks are merged into a robot-centric view of the surrounding objects and landmarks. Figure 4 shows a visualization of the particle filter. Each triangle indicates a hypothesis of the robot's position with the hollow triangle being the resulting position estimation of the robot in the world coordinate system. An abstract interface is presented to the next level of hierarchy enabling the user of the interface to specify e.g. scan modes of the cameras.

The next level of abstraction includes two particle filters [1] and a specialized control module. One particle filter estimates the robot's position relative to known landmarks. The second particle filter estimates the position of the ball relative to the

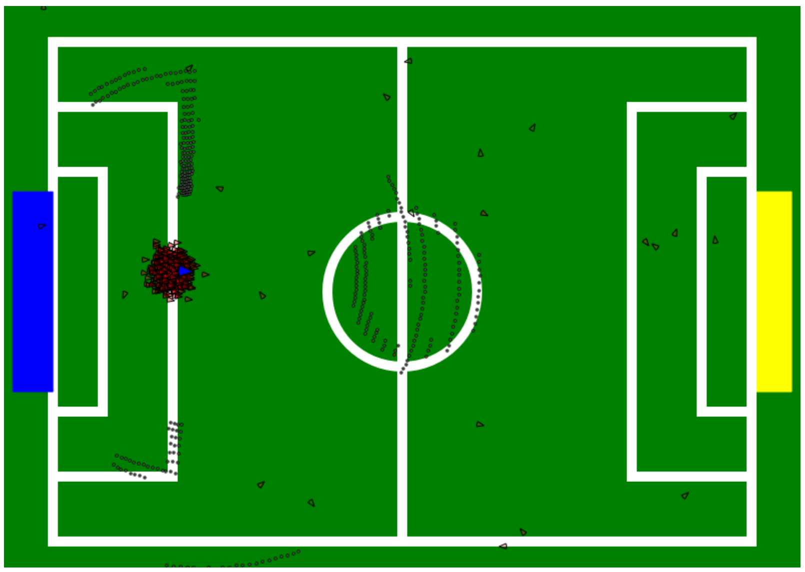

Fig. 4 Visualization of the particle filter and the robot's perceived artefacts (dots).

robot. The control module again presents an abstract interface to the next level of abstraction. Using this interface two views are accessible. One "global view" with global world coordinates including all absolute coordinates of objects and landmarks. However, the second view is robot-centric using relative coordinates.

The behavior based system descibed in Section 4 is located on the highest level of abstraction. A dedicated module within this system takes care about the behavior of the underlying vision system, e.g. which part of the field is to be examined or whether the ball has to be tracked.

Compared to systems using an omnivision camera [14], on our system the resolution is higher for a given viewing direction. Furthermore the system allows the over-sampling of a specified region of interest. Due to the constant usage of abstraction throughout the system this is done autonomously, e.g. for the position of the ball. This enables the system to recognize even distant objects that would be indistinguishable in a typical omnivision setup with only one fixed camera.

6 Coordination of functional units

The architecture does not impose limits upon the way data is exchanged between functional units. Most units will work asynchronously regarding each other and can work in a time-triggered or event-triggered manner. An example for an asynchronous time-triggered operation are the cameras attached to the vision system that will deliver data in periodic intervals that cannot practically be synchronized with the rest of the system. The high-level behavior system is running at a different rate unsynchronized to the cameras. In contrast, sensors like a ball detection sensor can trigger event processing and event messages that are non-deterministic in their timing.

To bridge the gap between such different execution semantics, an abstraction layer is introduced. It decouples the communication of the unit. Double buffering with atomic copying is used to ensure integrity for data transfers. Depending on the type of data, new data either is queued or overwrites an old value for a last-recently-received type of information.

7 Inter-robot strategy

This work focuses on the hierarchical distribution of functional units over embedded systems onboard a robot. Nevertheless, integrating the robot with its surrounding is an important task, too. Conceiving multiple autonomous mobile robots as a team brings up the question of task distribution as well.

Different from onboard the robot, task distribution is dynamic and depends on external non-deterministic parameters like the amount of robots available. Merging multiple robots of different types into a heterogeneous team allows for specialized task fulfillment but further complicates task allocation.

In general a strategy is needed to conquer a given objective with the available resources. In the existing homogeneous Paderkicker robot team the external architecture comprises a central dedicated server that oversees availability and state of the robots. It holds the strategy to be executed and dynamically decides which robot executes what task. This design has several drawbacks. The central component is a single point of failure that can render a complete team inoperable when it fails. Since communication reliability is of major concern in almost all situations, the team was designed to complete a task autonomously once the role is assigned.

To enhance this situation further, in the future the task decision process will also be distributed among the robots. This allows for decentralized strategic components that can be locally implemented on a robot. Integrating new and yet unknown robots with unique features does not imply having to change a central server rule-set anymore. Instead the robot specific parts of task assignment strategies can be implemented locally and therefore be distributed among the set of robots.

8 Outlook and conclusion

The future direction clearly indicates an even further distributed approach internal as well as external of a single robot. A distributed communication framework will act as a framework towards autonomous decision making in teams. By using a modular design and distributing functional units of the system onboard a robot among embedded systems, the stability of the whole system increases. Distributing the driving low-level behavior over multiple microcontrollers leads to faster reactions, e.g. regarding the compensation of transmission slip.

References

1. M.S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on]*, 50(2):174–188, Feb 2002.
2. Krishnakumar Balasubramanian, Jaiganesh Balasubramanian, Jeff Parsons, Aniruddha Gokhale, and Douglas C. Schmidt. A platform-independent component modeling language for distributed real-time and embedded systems. *J. Comput. Syst. Sci.*, 73(2):171–185, 2007.
3. D. Beier, R. Billert, B. Brüderlin, Bernd Kleinjohann, and Dirk Stichling. Marker-less vision based tracking for mobile augmented reality. In *Proceedings of the Second International Symposium on Mixed and Augmented Reality (ISMAR 2003)*, Tokyo, Japan, October 2003.
4. Stefan Christen, Ronny Hartanto, Benjamin Maus, Walter Nowak, Sven Olufs, Paul G. Ploger, Michael Reckhaus, Christian Rempis, Azamat Shakhimardanov, and Lars Weber. AIS–BIT Robots Team Description 2006. Technical report, FH Bonn-Rhein-Sieg and Fraunhofer AIS, 2006.
5. Natascha Esau, Bernd Kleinjohann, Lisa Kleinjohann, and Dirk Stichling. Visitrack – video based incremental tracking in real-time. In *Proceedings of the 6th IEEE International Symposium on Object-oriented Real-time Computing (ISORC '03)*, Hakodate, Japan, May 2003.
6. Antnio Paulo Moreira, Paulo Costa, Andr Scolari, Armando Sousa, and Paulo Marques. 5dpo–2000 Team Description for RoboCup 2006. Technical report, FEUP - Faculdade de Engenharia da Universidade do Porto, 2006.
7. Willi Richert and Bernd Kleinjohann. Towards robust layered learning. In *IEEE International Conference on Autonomic and Autonomous Systems (ICAS'07)*, June 2007.
8. Willi Richert, Bernd Kleinjohann, and Lisa Kleinjohann. Evolving agent societies through imitation controlled by artificial emotions. In *International Conference on Intelligent Computing, ICIC 2005*, number 3644 in LNCS, pages 1004–1013. Springer-Verlag Berlin, June 2005.
9. Willi Richert, Bernd Kleinjohann, Markus Koch, and Philipp Adelt. The paderkicker team: Autonomy in realtime environments. In *Proceedings of the Working Conference on Distributed and Parallel Embedded Systems (DIPES 2006)*, October 2006.
10. Dirk Stichling. *VisiTrack - Inkrementelles Kameratracking fr mobile Echtzeitsysteme*. PhD thesis, Universitt Paderborn, Fakultt fr Elektrotechnik, Informatik und Mathematik, 2004.
11. Dirk Stichling and Bernd Kleinjohann. CV-SDF – a model for real-time computer vision applications. In *IEEE Workshop on Application of Computer Vision*, Orlando, Florida, December 2002. IEEE.
12. Dirk Stichling and Bernd Kleinjohann. Low latency color segmentation on embedded real-time systems. In *Design and Analysis of Distributed Embedded Systems*. Kluwer Academic Publishers, November 2002.
13. 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)*, Halifax, Canada, June 2003.
14. Felix v. Hundelshausen, Sven Behnke, and Raúl Rojas. An omnidirectional vision system that finds and tracks color edges and blobs. *Lecture Notes In Computer Science*, 2377:374–379, 2002.
15. Jules White and Douglas C. Schmidt. Automated configuration of component-based distributed real-time and embedded systems from feature models. *Proceedings of the 17th Annual Conference of the International Federation of Automatic Control*, 2008.