Heterogeneity and On-Board Control in the Small Robots League

Andreas Birk and Holger Kenn

Vrije Universiteit Brussel, Artificial Intelligence Laboratory, Belgium
c/o birk@ieee.org, http://arti.vub.ac.be/~cyrano

Abstract. Versatile physical and behavioral features as well as their exploitation through computation-power onboard the robot-players are feasible and necessary goals for the RoboCup small robots league. We substantiate this claim in this paper by classifying different approaches and by discussing their potentials and limitations for research on AI and robotics. Furthermore, we present the most recent results of our approach to these goals in form of the so-called CubeSystem, a kind of construction-kit for robots and other autonomous systems. It is based on a very compact embedded computer, the so-called RoboCube, a set of sensor- and motor-modules, and software support in form of a special operating system and highlevel languages.

1 Introduction

The Small Robots League of RoboCup [KAK+97, KTS+97] allows global sensing, especially bird's view vision from an overhead camera, and restricts the size of the physical players to a rather extreme minimum. These two, most significant features of the small robots league bear an immense potential, but as well some major pitfalls for future research within the RoboCup framework.

First of all, it is tempting to exploit the set-up with an overhead camera for the mere sake of trying to win, reducing the robot-players to RF-controlled toy-cars within a minimal, but very fast vision-based closed-loop. The severe size limitations of the players in addition encourage the use of such "string-puppets" with off-board sensing and control instead of real robots. The Mirosot competition gives an example for this type of approach [Mir]. This framework would lead to dedicated solutions, which are very efficient and competitive, but only of very limited scientific interest from both a basic research as well as from an application-oriented viewpoint. If the teams in the small robots league would follow that road, this league could degenerate to a completely competition-oriented race of scientifically meaningless, specialized engineering efforts.

Though the two major properties of the small robots league, global sensing and severe size restrictions, discourage the important investigation of on-board control, they also have positive effects. First of all, the global sensing eases quite some perception problems, allowing to focus on other important scientific issues, especially team behavior. An indication for this hypothesis is the apparent difference in team-skills between the small robots league and the midsize league, where global sensing is banned.

The size restrictions as a second point also have a beneficial aspect for the investigation of team-behavior. The play-field of a ping-pong-table can easily be allocated in a standard academic environment, facilitating games throughout the year. It is in contrast difficult to embed a regular field of the midsize league into an academic environment, thus the possibilities for continuous research on the complete team are here limited. The severe size restriction of the small robots league has another advantage. These robots can be much cheaper as costs of electro-mechanical parts significantly increase with size. Therefore, it is more feasible to build even two teams and to play real games throughout the year, plus to include the team(s) in educational activities.

The rest of this article is structured as follows. In section two, different team-approaches are classified and possible implications are discussed. Section three presents the hardware aspects of the CubeSystem, i.e., the RoboCube V2.0 and the mechanical components used in our approach to the RoboCup small robots league. In the fourth section, the software aspects of the CubeSystem are discussed. First, its operating system CubeOS and highlevel language support are shortly presented. Then, it is shown with the example of path-planning that the RoboCube is indeed capable of quite powerful computations within realtime constraints. Last but not least the implications for team-coordination when using very heterogeneous systems are discussed. Section five concludes the paper.

2 Classification of Team-Approaches

For a more detailed discussion of the role of heterogeneity and on-board control in the small robots league, it is useful to have a classification of different types of teams and players.

Minoru Asada for example proposed in the RoboCup mailing-list to use a classification of approaches based on the type of vision (local, global or combined) and the number of CPUs (one or multi). He also mentioned that in the case of multiple CPUs a difference between systems with and without explicit communication between players can be made. Though this scheme is useful, it is still a first, quite rough classification. Therefore, we propose here to make finer distinctions, based on a set of crucial components for the players.

In general, a RoboCup team consists of a (possibly empty) set of host-computers and off-board sensors, and a non-empty set of players, each of which consist of a combination of the following components:

1. minimal components
   (a) mobile platform
   (b) energy supply
   (c) communication module
2. optional components
   (a) computation power
   (b) shooting-mechanism and other effectors
   (c) basic sensors

(d) vision hardware

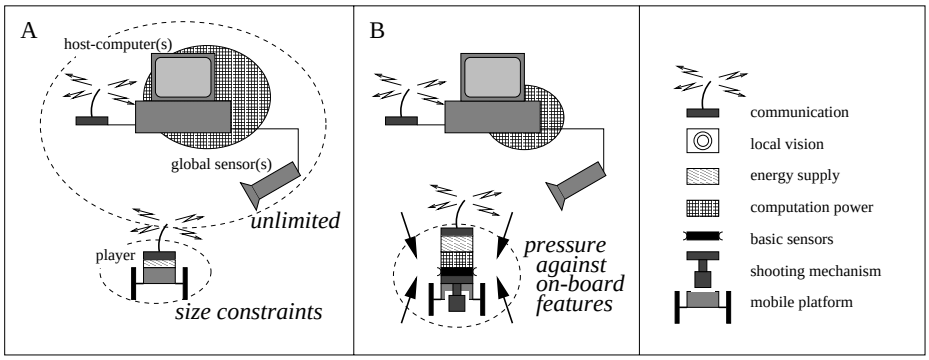

Fig. 1. There are several basic components which can be, except the minimal ones, freely combined to form a player. Situation A shows the most simple type of player, a radio-controlled toy-car, which can hardly be called a robot. Situation B shows a much more elaborated player. Unfortunately, the size-constraints of the small robots league put a strong negative pressure against the important implementation of on-board features for the players.

Note, that the most simple type of player, consisting of only minimal components, is hardly a robot. It is more like a "string-puppet" in form of a radio-controlled toy-car without even any on-board sensors or computation power (though it could well be possible that this type of device has an on-board micro-controller for handling the communication protocol and the pulse-width-modulation of the drive motors). The actual control of this type of players completely takes place on the off-board host(s).

Based on this minimal type of player, the optional components can be freely combined and added. In doing so, there is a trade-off between

- on-board sensor/motor components,
- on-board computation power, and
- communication bandwidth.

A player can for example be built without any on-board computation power at the cost of communication bandwidth by transmitting all sensor/motor-data to the host and back. So, increasing on-board computation power facilitates the use of a smaller communication bandwidth and vice versa. Increasing sensor/motor channels on the other hand increases the need of on-board computation power and/or communication bandwidth.

On-board features are important for research in robotics as well as AI and related disciplines for several reasons. Mainly, they allow research on important aspects which are otherwise impossible to investigate, especially in the field of sensor/motor capabilities. For effector-systems for example, it is quite obvious that they have to be on-board to be within the rules of soccer-playing. Here, the possibilities of systems with many degrees of freedom, as for example demonstrated in the SONY pet dog [FK97], should not only be encouraged in special leagues as e.g. in the one for legged players, but also within the small robots league. In general, a further splitting of the RoboCup activities into too many leagues seems not to be beneficial and it also seems not to be practical. Too many classifications which would justify just another new league would be possible. In addition, the direct competition and comparison of different approaches together with the scientific dialogue are one of the main features of RoboCup.

In the case of sensors and perception, the situation is similar to the one of effector-systems, i.e., certain important types of research can only be done with on-board devices. This holds especially for local vision. It might be useful to clarify here the often confused notions of local/global and on-/off-board. The terms on- and off-board are easy to distinguish, general properties. They refer to a piece of hardware or software, which is physically or logically present on the player (on-board) or not (off-board). The notions of local and global in contrast only refer to sensors, i.e., particular types of hardware, or to perception, i.e., particular types of software dealing with sensor-data. Global sensors and perception tell a player absolute information about the world, typically information about its position and maybe the positions of other objects on the playfield. Local sensors and perception in contrast tell a player information about the world, which is relative to its own position in the world. Unlike in the case of on- and off-board, the distinction between local and global is fuzzy and often debatable. Nevertheless, it is quite clear that the important issue of local vision can only be investigated if the related feature is present on-board of the player.

Hand in hand with an increased use of sensor and motor systems on a player, the amount of on-board computation power must increase. Otherwise, the scarce resource of communication bandwidth will be used up very quickly. Note, that there are many systems using RF-communication at the same time during a RoboCup tournament. Especially in the small robots league, were only few and very limited off-the-shelf products suited for communication exist, transmission of large amount of data is impossible. It is for example quite infeasible to transmit high-resolution local camera images from every player to a host for processing.

3 Towards a Robot Construction-Kit

3.1 The Motivation

Existing commercial construction-kits with some computational power like Lego Mindstorms^{TM} [Min] or Fischertechnik Computing^{TM} [Fis] are still much too limited to be used for serious robotics education or even research. Therefore, we decided to develop our own so-to-say robot construction-kit.

3.2 RoboCube V2.0

For RoboCup'98, the VUB AI-lab team focused on the development of a suited hardware architecture, which allows to implement a wide range of different robots. The basic features of this so-called RoboCube-system are described in [BKW98]. For RoboCup'99, the system was further improved and extended. A more detailed description is given in [BKW00].

The most recent version of the RoboCube boots out of a 1 MByte Flash-EPROM which holds a basic input/output operating system (BIOS) and offers space for a small file system. A huge part of the BIOS is dedicated to the efficient handling of different actuators and sensors. In the basic configuration the main memory consists of a 1 MByte low power SRAM, which can be extended by additional 12 MByte.

In its basic version, one I/O subsystem board of the RoboCube features

- 24 analog input
- 6 analog output
- 16 binary Input/Output (binI/O)
- 4 timer channels (TPC)
- 4 DC-motor controller with quadrature-encoding

The number of ports can simply be doubled by stacking a second I/O subsystem board on top of the first one. All sensor-motor-interfaces come with proper software support allowing an easy high-level usage.

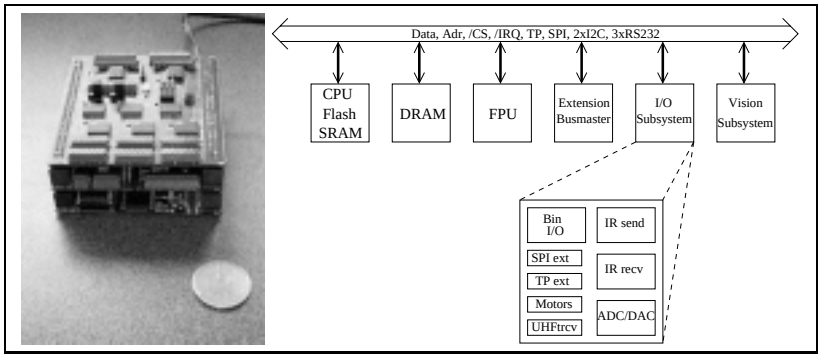

Fig. 2. A picture of the RoboCube (left) and the layout of its internal bus structure (right).

The RoboCube-system is constantly further improved, on the software as well as on the hardware side. At the moment for example, several options for inexpensive high-resolution color-vision are investigated.

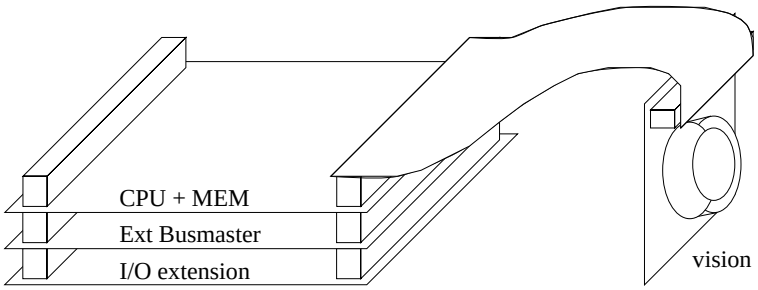

Fig. 3. Physical layout of the complete RoboCube

3.3 Mechanical Components for RoboCup

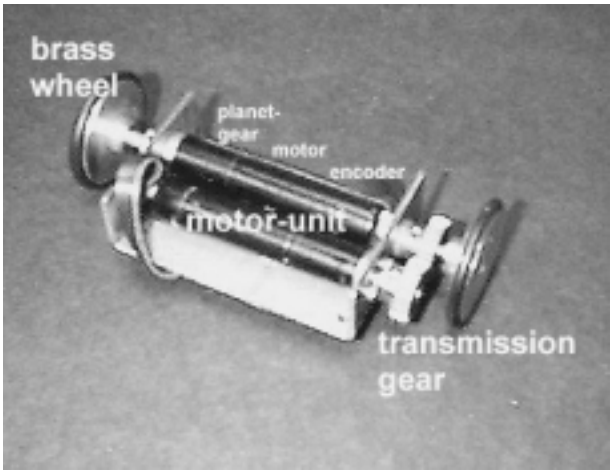

Fig. 4. The drive unit as a mechanical building-block, which can be mounted on differently shaped bottom-plates, forming the mechanical basis for diverse body-forms. Different ratios for the planetary gears in the motor-units are available, such that several trade-offs for speed versus torque are possible.

In some of our education and research activities, the RoboCube-system is combined with Lego^{TM} or Fischertechnik^{TM} components on the mechanical side. For RoboCup competitions, we developed a solid but still flexible solution based on metal components.

Keeping the basic philosophy of construction-kits, a "universal" building block is used for the drive (figure 4) of the robots. The drive can be easily

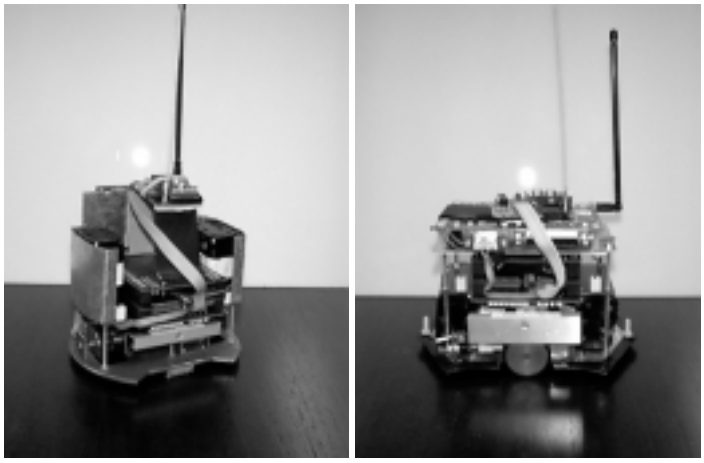

Fig. 5. A forward- (left) and a defender-type (right) robot. The mechanical set-up of the robot-players is based on a piled-stack approach such that different components, such as shooting-mechanisms and the RoboCube, can easily be added.

mounted onto differently shaped metal bottom-plates, forming the basis for different body-forms like the ones shown in figure 5. The motor-units in the drive exist with different ratios for the planetary gears, such that several trade-offs for speed versus torque are possible.

Other components, like e.g. shooting-mechanisms and the RoboCube, are added to the bottom-plate in a piled-stack-approach, i.e., four threaded rods allow to attach several layers of supporting plates.

4 Powerful On-Board Control

4.1 Operating System and Programming Languages

For the RoboCube, an embedded operating system has been developed, *CubeOS*. It provides the usual features like threads, semaphores, realtime clock, communication and I/O drivers in a small core of about 30 Kbytes. Additionally, it provides functionality that set it apart from other OS kernels and make it especially useful for robotics and autonomous systems in general:

- Drivers that handle access to the various sensor and actuator devices of the RoboCube
- Support for hardware-assisted realtime processing through the MC68332's onboard TPU
- A low-latency communication protocol engine for radio communication
- hardware-independent data encoding as defined in the *External Data Representation Standard* [SM]

CubeOS is written in C and makes use of the Free Software Foundation's Gnu C-Cross-Compiler. The host system for development is a Unix or Linux Workstation, the code is downloaded into the target via a wireless serial communication link.

On top of the CubeOS API, a simple software framework has been implemented to provide easy access within normal C-programs. In additional, there is highlevel language support suited even for novice programmers. This framework, named *NPDL* (for New Process Description Language) provides several simple constructs to create control programs for robots. Within education projects, NPDL has already been mastered by students studying economics, philosophy and architecture.

4.2 Using the RoboCube for Highlevel Control

Though the RoboCube has quite some computation power for its size, its capabilities are nevertheless far from those of desktop machines. So, it is not obvious that interesting behaviors in addition to controlling the drive-motors and shooting can actually be implemented on the RoboCube, i.e., on board of the robots. Therefore, we demonstrate in this section that for example path-planning with obstacle avoidance is feasible.

24	23	22	21	20	19	18	17	16	15	14	15	16	17	18
23	22	21	20	19	18	17	16	15	14	13	14	15	16	17
22	21	20	19	18	17	16	15	14	13	12	13	14	15	16
21	20	19	18	17	16	15	14	13	12	11	12	13	14	15
20	19	18	17	16	15	14	13	12	11	10	11	12	13	14
19	18	17	16	15	14	13	12	11	10	9	10	11	12	13
18	17	16	15	14	13	12	11	10	9	8	9	10	11	12
19	18	17	16	[X]	[X]	[X]	[X]	[X]	8	7	8	9	10	11
18	17	16	17	[X]	[X]	[X]	[X]	[X]	7	6	7	8	9	10
17	16	15	16	[X]	[X]	[X]	[X]	[X]	6	5	6	7	8	9
16	15	14	15	[X]	[X]	[X]	[X]	[X]	5	4	5	6	7	8
15	14	13	14	[X]	[X]	[X]	[X]	[X]	4	3	4	5	6	7
14	13	12	[X]	[X]	[X]	6	5	4	3	2	3	4	[X]	[X]
13	12	11	[X]	[X]	[X]	5	4	3	2	1	2	3	[X]	[X]
12	11	10	[X]	[X]	[X]	4	3	2	1	0	1	2	[X]	[X]
11	10	9	8	7	6	5	4	3	2	1	2	3	4	5
12	11	10	9	8	7	6	5	4	3	2	3	4	5	6
13	12	11	10	9	8	7	6	5	4	3	4	5	6	7
14	13	12	11	10	9	8	7	6	5	4	5	6	7	8
15	14	13	12	11	10	9	8	7	6	5	6	7	8	9

Fig. 6. A potential field for motion-control based on Manhattan distances. Each cell in the grid shows the shortest distance to a destination (marked with Zero) while avoiding obstacles, which are marked with '[X]'.

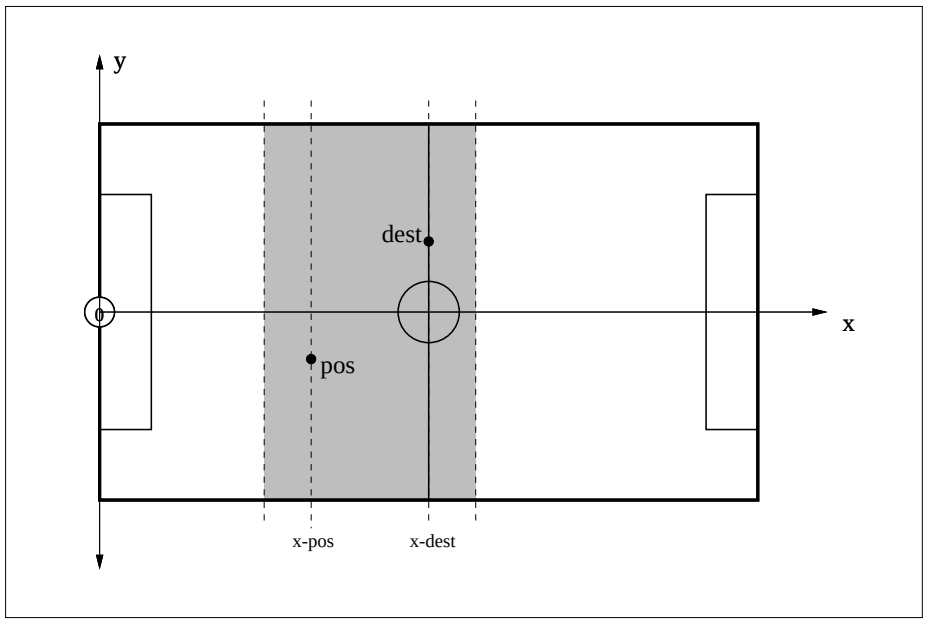

Fig. 7. The potential field (grey area) is not computed for the whole soccer-field. Instead, it is limited in the x-direction to save computation time.

Path planning is with most common approaches rather computationally expensive. Therefore, we developed a fast potential field algorithm based on Manhattan-distances. Please note that this algorithm is presented here only to demonstrate the computing capabilities of the RoboCube. A detailed description and discussion of the algorithm is given in [Bir99].

Given a destination and a set of arbitrary obstacles, the algorithm computes for each cell of a grid the shortest distance to the destination while avoiding the obstacles (figure 6). Thus, the cells can be used as gradients to guide the robot. The algorithm is very fast, namely linear in the number of cells. The algorithm is inspired by [Bir96], where shortest Manhattan distances between identical pixels in two pictures are used to estimate the similarity of images.

The basic principle of the algorithm is region-growing based on a FIFO queue. At the start, the grid-value of the destination is set to Zero and it is added to the queue. While the queue is not empty, a position is dequeued and its four neighbors are handled, i.e., if their grid-value is not known yet, it is updated to the current distance plus One, and they are added to the queue.

For the experiments done so far, the resolution of the motion-grid is set to 1cm. As illustrated in figure 7, the potential-field is not computed for the whole soccer-field to save computation time. Given a robot position *pos* and a

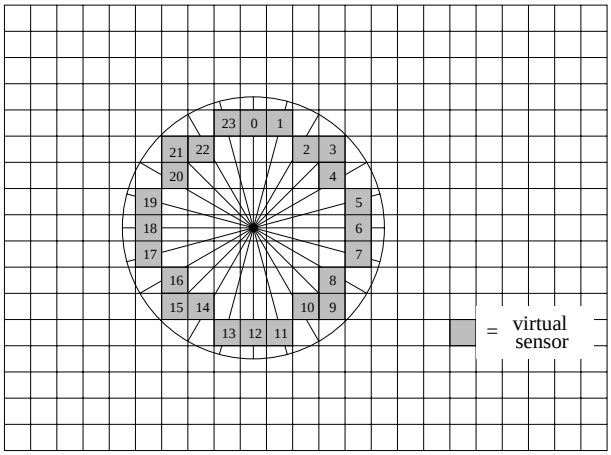

Fig. 8. Twenty-four so-called virtual sensors read the potential values around the robot position on the motion grid. The sensor values can be used to compute a gradient for the shortest path to the destination, which can be easily used in a reactive motion-control.

destination *dest*, the field is restricted in the x-direction to the difference of *pos* and *dest* plus two safety-margins which allow to move around obstacles to reach the destination.

The motion-grid is used as follows for our soccer-robots. The global vision detects all players, including opponents and the ball, and broadcasts this information to the robots. Each robot computes a destination depending on its strategies, which are also running on-board. Then, each robot computes its motion-grid. In doing so, all other robots are placed on the grid as obstacles.

Robots have so-called virtual sensors to sample a motion-grid as illustrated in figure 8. The sensor values are used to calculate a gradient for a shortest path to the destination, which is ideal for a reactive motion control of the robot. In doing so, dead-reckoning keeps track of the robot's position on the motion-grid.

Of course, the reactive control-loop can only be used for a limited amount of time for two main reasons. First, obstacles move, so the motion-grid has to be updated. Second, dead-reckoning suffers from cumulative errors. Therefore, this loop is aborted as soon as new vision information reaches the robot, which happens several times per second, and a new reactive controller based on a new motion-grid is started.

Figure 9 shows performs-results of the path-planning algorithm running on a RoboCube as part of the control-program of the robot-players. The different tasks of the control-program proceed in cycles. The execution time refers to a single execution of each task on its own (including the overhead from the operating system). The frequency refers to the frequency with which each tasks is executed as part of the player-control, i.e., together with all other tasks.

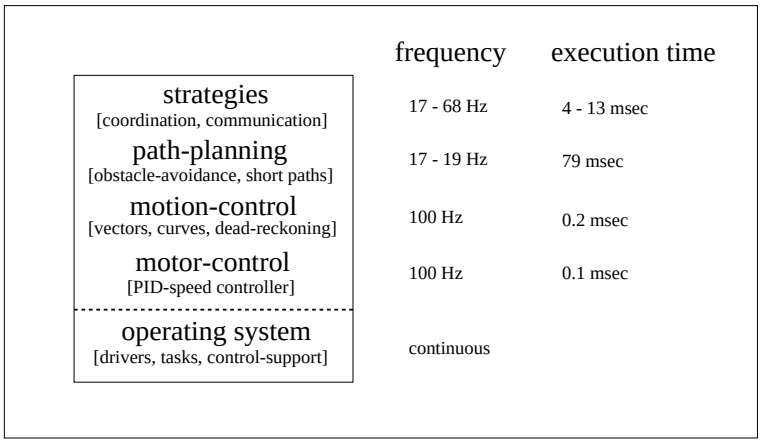

Fig. 9. The path-planning is part of a four-level software architecture which controls the robots players. It runs, together with the CubeOS operating system, completely on board of the RoboCube.

The control-program consists of four levels which run together with the CubeOS completely on-board of the RoboCube. The two lowest levels of motor- and motion-control run at a fixed frequency of 100 Hz. Single iterations of them are extremely fast as the TPU of the MC68332 can take over substantial parts of the processing. The strategy and path-planning level run in an "as fast as possible"-mode, i.e., they proceed in event-driven cycles with varying frequencies.

The execution of the pure strategy-code, i.e., the action-selection itself, takes up only a few milliseconds. Its frequency is mainly determined by whether the robot is surrounded by obstacles or not, i.e., whether path-planning is necessary or not. The computation of the motion-grid takes most of the 79 msec needed for path-planning. As two grids are used, one still determines the motion of the robot while the next one is computed, the cycle-frequency is at least 17 Hz. So, in a worst case scenario where the player is constantly surrounded by obstacles, the action-selection cycle can still run at 17 Hz.

4.3 Heterogeneity and Team Coordination with On-Board Control

Heterogeneity is an important feature for soccer with human players as much as with robot players. It is the main basis for adaptability of a team, either to different opponent teams within a tournament, or to the general progress of a particular game, or to very momentary situations. Heterogeneity within soccer can range from high-level roles of players in a team like forward or defender, down to different body features covering a wide-range of physical trade-offs like e.g. speed versus torque.

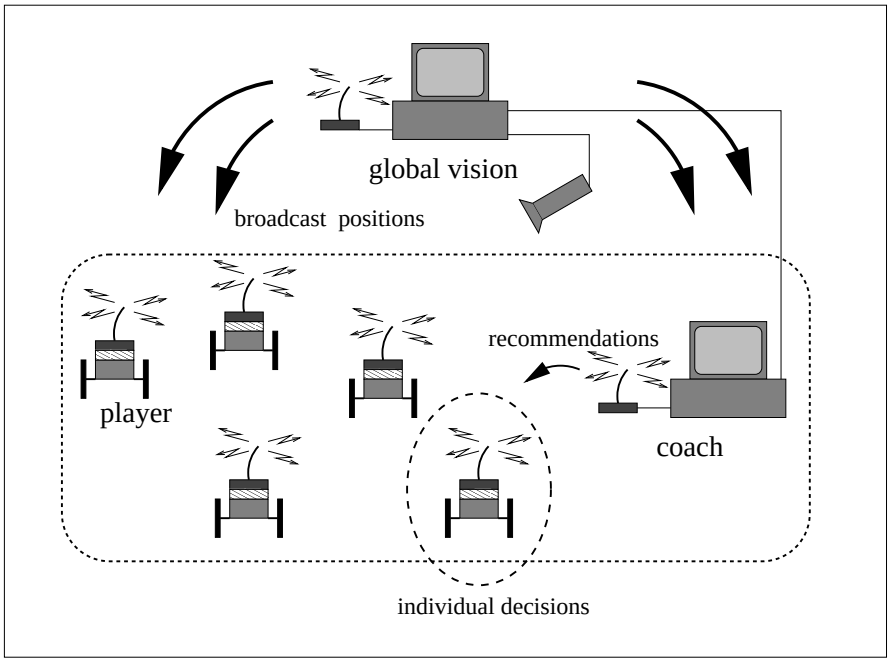

Fig. 10. A "pares inter pares" coach, residing on the same level as the players in the coordination hierarchy. This coach can be used for facilitating the coordination of heterogeneous team approaches while keeping as much as possible on-board control. The basic idea is that most of the time, the robot players decide completely on their own what to do based on their on-board control-program. Only occasionally the coach interferes as he has additional information about the capabilities of the players.

Straightforward approaches to team coordination with the expressive power of finite state automata are doomed to fail under such wide ranges of heterogeneity due to the combinatorial explosion of states. Therefore, we investigate coordination schemes based on operational semantics, which allow an extremely compact and modular way of specifying team behaviors. One step in this direction is the *Protocol Operational Semantics (POS)*, an interaction protocol based on abstract data-types and pattern matching capabilities. So far, it has only been tested in simulations, but the results are very promising. A detailed description can be found in [OBK99].

Here, we focus on the question how this approach can be integrated with a substantial exploitation of on-board control. The problem is that the expressive power of operational semantics is bought at the price of computational power. POS for example is implemented in Pizza [OW97], a super-set of Java.

A solution for this problem is a kind of additional player in form of a "pares inter pares" coach. This coach resides not above the other players in a coordination hierarchy, but resides on the same level (figure 10). The position information from the global vision is broadcasted to all players and the coach. Most of the time, all robot players individually decide what to do, based on their on-board computations. Only on rare occasions, the coach interferes as he has more background information available than the players.

To illustrate this idea, let us assume there are two heterogeneous robots of type A and A', with rather limited differences and which can be substituted against each other in the team. Both can simply run the same on-board control program, deciding most of the time the actions of the player. Only in situations when the difference plays a role, the coach interferes and provides additional information, recommending alternative actions to the player.

5 Conclusion

We claim that for serious AI and robotics research, it is necessary to work with "real" systems, i.e., heterogeneous devices with on-board control. As our contribution towards a suited infrastructure for this type of research, we develop the CubeSystem, a kind of advanced construction-kit for mobile robots and other autonomous systems. The CubeSystem consists of a special embedded hardware, the RoboCube, a set of sensors and actuators, and software support in form of a special operating system, the CubeOS, and highlevel languages.

The "string-puppet" approach of simple radio-controlled toy-cars also has its validation. It can for example serve as a rather easy and inexpensive way to enter RoboCup, it can be useful for educational purposes; shortly, it can be good for a start and to get acquainted with the basic issues of RoboCup.

But in the long run, we hope that participants in the small robots league of RoboCup cooperate to improve the options of on-board features. Only through a joint effort, it will be possible to overcome the pitfalls and to mutually benefit from the positive potential of the limited size requirements in this league.

Acknowledgments

The VUB AI-Lab team thanks Sanders Birnie BV as supplier and Maxon Motors as manufacturer for sponsoring our motor-units. Andreas Birk is a research fellow of the Flemish Institute for Applied Science (IWT); research on RoboCup is partially financed within this framework (OZM980252).

References

[Bir96] Andreas Birk. Learning geometric concepts with an evolutionary algorithm. In *Proc. of The Fifth Annual Conference on Evolutionary Programming.* The MIT Press, Cambridge, 1996.

[Bir99] Andreas Birk. A fast pathplanning algorithm for mobile robots. Technical report, Vrije Universiteit Brussel, AI-Laboratory, 1999.

[BKW98] Andreas Birk, Holger Kenn, and Thomas Walle. Robocube: an "universal" "special-purpose" hardware for the robocup small robots league. In *4th International Symposium on Distributed Autonomous Robotic Systems*. Springer, 1998.

[BKW00] Andreas Birk, Holger Kenn, and Thomas Walle. On-board control in the robocup small robots league. *Advanced Robotics Journal*, 2000.

[Fis] The fischertechnik^{TM} website. http://www.fischertechnik.de/.

[FK97] Masahiro Fujita and Koji Kageyama. An open architecture for robot entertainment. In *Proceedings of Autonomous Agents 97*. ACM Press, 1997.

[KAK+97] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, and Eiichi Osawa. Robocup: The robot world cup initiative. In *Proc. of The First International Conference on Autonomous Agents (Agents-97)*. The ACM Press, 1997.

[KTS+97] Hiroaki Kitano, Milind Tambe, Peter Stone, Manuela Veloso, Silvia Coradeschi, Eiichi Osawa, Hitoshi Matsubara, Itsuki Noda, and Minoru Asada. The robocup synthetic agent challenge 97. In *Proceedings of IJCAI-97*, 1997.

[Min] The lego mindstorms^{TM} website. http://www.legomindstorms.com/.

[Mir] The micro-robot world cup soccer tournament (mirosot). http://www.mirosit.org.

[OBK99] Pierre-Yves Oudeyer, Andreas Birk, and Jean-Luc Koning. Interaction protocols with operational semantics and the coordination of heterogeneous soccer-robots. Technical report, Vrije Universiteit Brussel, AI-Laboratory, 1999.

[OW97] Martin Odersky and Philip Wadler. Pizza into Java: Translating theory into practice. In *Proc. 24th ACM Symposium on Principles of Programming Languages*, 1997.

[SM] Inc. Sun Microsystems. Xdr: External data representation standard. Request for Comments.