The Dutch F2000 RoboCup Team

Dutch-Team

Pieter Jonker, Emiel Corten, Natasha Polykarpova, Frans Groen

PJ, NP: Pattern Recognition Group, TNW, Delft University of Technology
EC, FG: Intelligent Autonomous Systems, WINS, University of Amsterdam

1 Introduction

Creation of real world agent systems involves the solution of numerous problems from different areas, such as distributed sensor data fusion, development of the adaptive team behaviour, building of the architecture and computational environment. Many of these problems are being dealt in the AIR (Autonomous Interacting Robotics) project, performed by groups from the universities of Amsterdam (UvA, VU), Utrecht (UU) and Delft (TUD) in The Netherlands [15] [16]. To provide a standard problem for the examination and integration of technologies developed in the framework of the multi agent paradigm, the soccer game was chosen. The soccer application includes the solution of major typical problems, which arise in the course of multi agent system creation. Starting from 1997, international competitions in simulation, and small and middle size robot leagues have been held [1]. This paper is devoted to the hard and software architecture of soccer playing autonomous robots for the 'Dutch Team' of the co-operating universities. The software architecture is set-up in such a way that it not only defines the distributed computing and communication between the different robots, but also supports sole software simulations of multi-agent systems and its visualisation of the world-model shared by the robots: Most of the development of strategies and learning behaviour for multi-agent systems can be done in client/server simulations, where the best results are then implemented, tested and optimised for the real-world agent systems.

2 Hardware Architecture

The robots that will be used are the Nomad Super Scout II robots [2], see Figure 1. It is a mobile robot system with vision, 16 ultrasonic sensors, a tactile bumper ring, odometry sensors, a control processor for low-level tasks and an onboard industrial PC for high-level tasks. The effective range of the ultrasonic sensors is from 15 cm to 6.5 m. The hardware architecture we use is drawn in Figure 2. The high-level processor is a Pentium II 233MHz, 64MB, 4GB, on an Advantech Multi-media Single Board

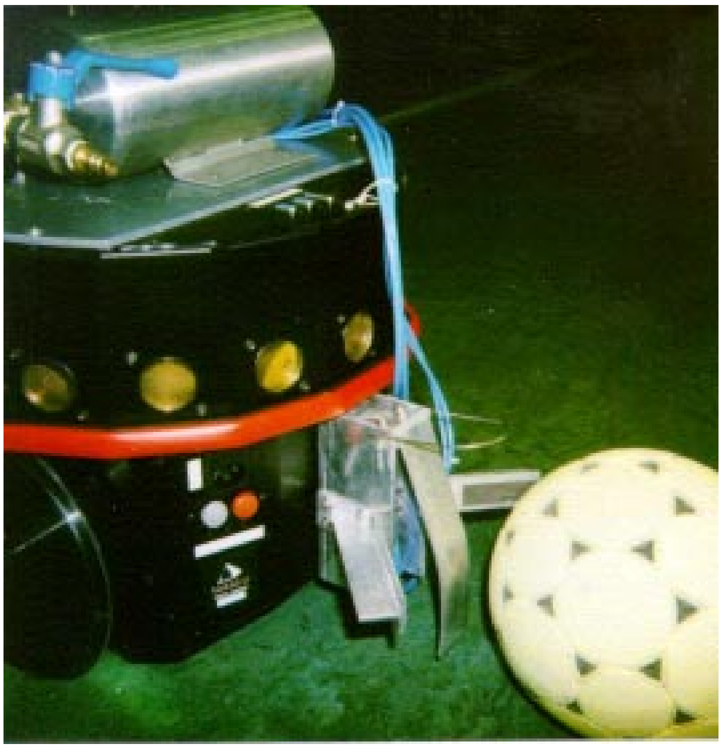

Figure 1: The Nomad autonomous robot with (prototype) kicking device

Computer PCM5862 [3]. It communicates with the low-level processor, a MC68332, through a serial port. Additionally, a TMS320C14 DSP is responsible for high-bandwidth motor control at 2 KHz control rates. The robot has a differential drive system, with two independent drive motors. For communication with other robots, the Ethernet port is connected to a wireless system BreezeCom SA-10 PRO [4], with a data rate of max. 3 Mbps, giving the robots an action radius of up to 1 Km. It uses the 2.4GHz ISM band and is compliant with IEEE 802.11.

Further Specifications:

Diameter: 41 cm., Height: 35 cm., Ground Clearance: 1.5 cm, Weight: 25 kg. (incl. batteries), Payload: 5 kg.

Battery Power: 432 watt-hour (removable)

Motion: 2 wheel differential drive @ geometric center, Speed: 1.0 m/sec, Acceleration: 2.0 m/s^2

Odometry: Encoder Resolution: Translation: 756 counts/cm, Rotation: 230 counts/degree

An AX10410 AD/DA/DIO board [5] is used to control a ball kicking mechanism. This mechanism is a pneumatic device, of which the pressure and hence the kicking and catching force can be controlled by a FESTO proportional pressure regulator MPPE3-1/8-6/010B [6] through the DA converter. The pressure in the air container can be read out with the AD converter.

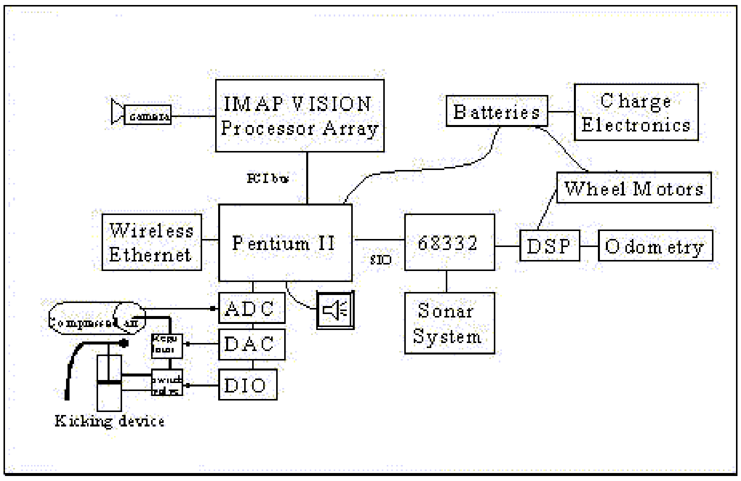

Figure 2: The Hardware Architecture

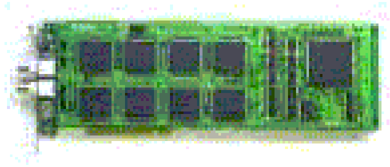

Figure 3: The IMAP-Vision System PCI board

The container is good for about 50 full force kicks. The robot can be programmed using a RedHat 5.2 [7] Linux-based software development environment.

The standard robots are equipped with a PCI framegrabber card with a colour camera. In this case, the high level processor will need to do the image processing tasks. As tracking the ball, team-mates and competitors was one of the most difficult tasks in past RoboCup matches, we will equip at least one of the robots with an IMAP-VISION System from NECs Incubation Center [8] instead of the standard WinTV card. This system is a Linear Processor Array with 256, 8 bits data processors in SIMD mode, colour framegrabbing hardware and a RISC control processor on one PCI board. It is a parallel architecture for real-time image processing, where a single column of an image is mapped onto one data processor. See Figure 3.

The IMAP-VISION system is a Linear Processor Array with 256, 8 bits data processors in SIMD mode, colour framegrabbing hardware and a RISC control processor on a PCI board. It is a parallel architecture specially made for real-time image processing, where a single column of an image is mapped onto one data processor. The system is programmed with a version of C extended for data parallel processing. See Figure 4. The language 1DC is designed as an enhanced C language to support virtual LPAs. The enhancement is straightforward: extended declaration of entities which associated to the PE array (SEP or distributed variables), extended constructors for selecting active processor groups, and extended operators (like mif) for manipulating data on the PE array.

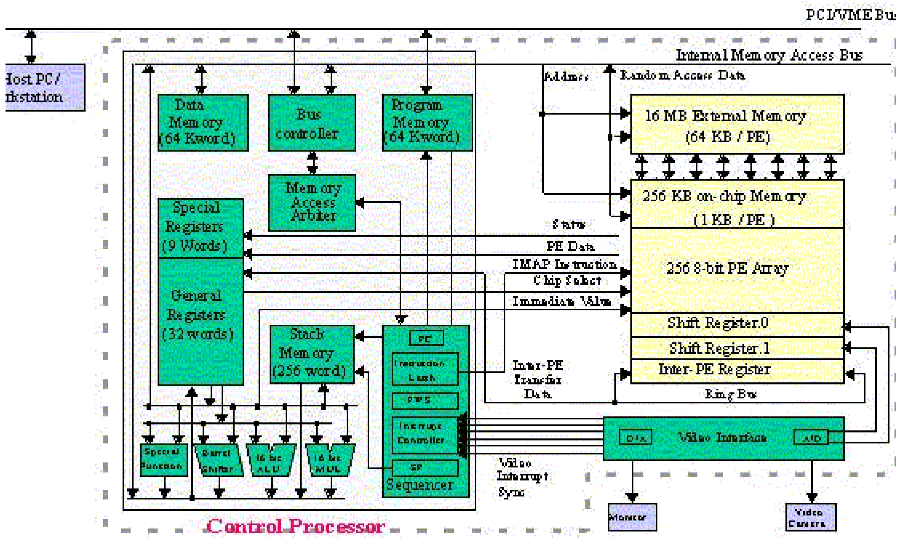

Figure 4: Hardware architecture of the IMAP Vision Board

The cameras used are Chugai YC02B cameras [9] with a Santec TC2814M-1/3, fixed zoom manual iris lenses, with diameter 1/3", focus 2.8 mm, aperture 1.2, opening angle 94 (!), CS mount [10].

3 Software Architecture

The proposed software architecture of the soccer robots is shown in Figure 5. It is based on the concept of information hiding modules [11], an old but still valuable concept that started the object oriented wave, well known now in languages as C++. Within this concept, to informally describe software, modules on a higher layer may use modules on a lower layer for their functioning. This uses relation is indicated with a line. The initiative for action always comes from the upper module, however upper modules may deposit a signal request at lower modules so they can be triggered by important events. Each module has its own autonomous functionality, which can be accessed through its interface function (in the figure to be imagined as residing at the top of a module). Hence the module can considered to be a virtual machine with a command set. Note that in general no assumptions have been made on how a module uses another. This may be done by a function call, a process fork, a thread submission or inlining of the interface function code. Figure 5 shows three identical robots connected to each other via a communication module. The architecture has three layers. The basic layer consists of virtual devices, the second layer of basic skills of the robot and the top layer consists of the mission strategy module. The software modules of the lowest layer hide the details and augment the capabilities of the physical devices. The software of the middle layer contains the intelligent basic skills of the system. Implementations based on learning systems, though not essential, may be important here. Within the top level, decisions can be made to grossly change the character of the game. The two upper layers can be used within a simulator as well as within the real-world soccer robots. Given the interfaces of the modules from the virtual device layer, separate groups can now focus on the development of

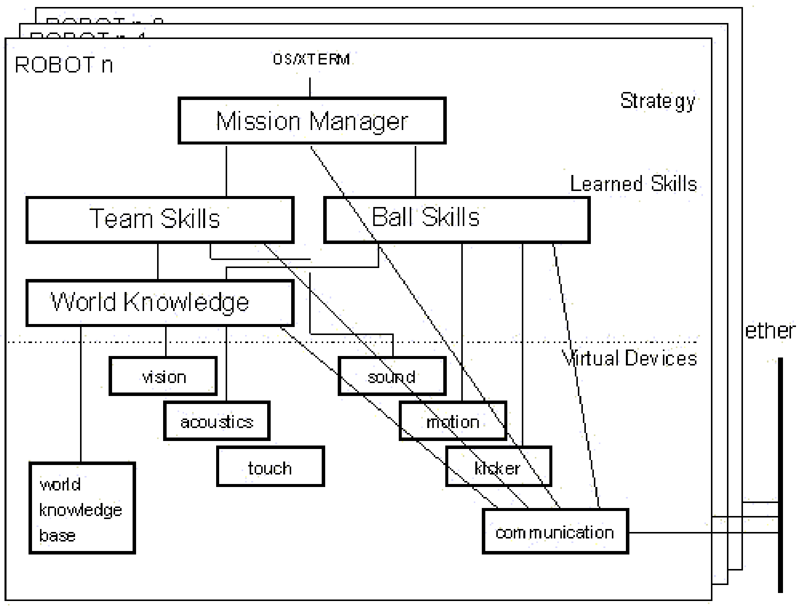

Figure 5: Software architecture of the autonomous soccer playing robots

the intelligence within the upper modules, using simulation.

The two upper layers can be used within a simulator as well as within the real-world soccer robots [12]. From the SkilledPlayer simulator [13], [14], we derived the module descriptions. In the implementation of the modules an overall error mechanism is used. Triggers are implemented using message passing techniques. A lower module deposits a message in the high priority queue of a module higher in the hierarchy. Normal public interface functions are implemented as strings to be deposited in the low priority task queue of the module. This mechanism is only used for the three modules in the "learned" skills layer. The public functions of the modules of the Virtual Device Layer as well as the Mission Manager Module in the top layer are implemented as direct funcion calls or "in lines" or macro's for a better speed performance. The upper modules have the following function:

- The World Knowledge Module autonomously maintains a consistent view on the world. Confidence on position, orientation, and headings is maintained by fusing information from all sensors and through world information obtained from the other robots.
- The Ball Skills Module controls autonomous skilled ball manipulation.
- The Team Skills Module manages the position of the player in the team. It can use predefined agreements in the team and information received by communication between players during the game.
- The Mission Manager Module autonomously fulfils the mission and role as communicated by the coach, while listening to the referee. Provisionally a decision tree approach will be used to generate the

control. In a second stage a system based on priorities and confidences. In a final stage learning will be used to optimise the robot's skills, team behaviour and mission fulfilment.

4 Acknowledgements

This work was partially sponsored y the Dutch Foundation for Pure Research NWO.

5 Bibliography

[1] [1][http://www.robocup.org]

[2] [2][http://www.robots.com/nsuperscout.htm]

[3] [3][http://www.advantech-usa.com/prselect/index.htm]

[4] [4][http://www.breezecom.com]

[5] [5][http://www.axiomtek.com/Products]

[6] [6][http://www.festo.com]

[7] [7][http://www.redhat.com]

[8] Y. Fujita et. al. "A 10 GIPS SIMD Processor for PC based Real-Time Vision Applications, Architecture, Algorithm Implementation and Language Support", Proc. of the IEEE Workshop on Computer Architectures for Machine Perception (CAMP 1997), pp22-32, Cambridge, MA, USA, 1997

[9] [8][http://www.chugai.com/cctv/cam]

[10] [9][http://www.teleconnect.nl/cctv-pages/lenzen.htm]

[11] D.L. Parnas, "A technique for Software Module Specification with Examples" Communications of the ACM. Volume 15, no 5, 1992

[12] [10][http://ci.etl.go.jp/ noda/soccer/server/]

[13] E. Corten, E. Rondema, "Team description of the Windmill Wanderers", in "Proceedings of the second RoboCup workshop, RoboCup-98, Paris", Juli 1998, pages 347-352

[14] J. Lubbers, R.R. Spaans, E.P.M. Corten, F.C.A. Groen, "AIACS: A Robotic Soccer Team Using the Priority/Confidence Model", in "Proceed-

[1]Ref: http://www.robocup.org/
[2]Ref: http://www.robots.com/nsuperscout.htm
[3]Ref: http://www.advantech-usa.com/prselect/index.htm
[4]Ref: http://www.breezecom.com/
[5]Ref: http://www.axiomtek.com/Products
[6]Ref: http://www.festo.com/
[7]Ref: http://www.redhat.com/
[8]Ref: http://www.chugai.com/cctv/cam
[9]Ref: http://www.teleconnect.nl/cctv-pages/lenzen.htm
[10]Ref: http://ci.etl.go.jp/ noda/soccer/server/

ings Xth Netherlands/Belgium conference on AI", 19-19 November 1998, p. 127-135

[15] [11][http://www.wins.uva.nl/ mielko/Soccer/Dutch-committee.html]

[16] [12][http://www.nat.vu.nl/ robocup]

[11]Ref: http://www.wins.uva.nl/ mielko/Soccer/Dutch-committee.html
[12]Ref: http://www.nat.vu.nl/ robocup