A light software architecture for a Humanoid Soccer Robot

Alberto Maggi[†], Tommaso Guseo[*], Federico Wegher[†], Enrico Pagello[*†], Emanuele Menegatti[*†]
[*] Intelligent Autonomous Systems Laboratory (IAS-Lab)
Department of Information Engineering
University of Padua, Italy

***Abstract*— In this paper, we present a software architecture that can be implemented on a humanoid platform with low–computational power to make it to autonomous. The platform we used is the Robovie-V of V-Stone. the application we chose is the RoboCup soccer competitions. We will present the simple behaviour selection architecture implemented with a finite state machine (FSM) in our robot. We will present the highly optimized algorithms used for image processing and we will give some hints on how it is possibile to extend the flexibility of a low computational power humanoid with a customized operating system. These solutions are quite general and can be applied to any humanoid platform with low-computational power.**

I. Introduction

Our robots are based on a modified Robovie-M platform of VStone. These robots are fully autonomous: CPU, power supply, sensor and obviously actuators are on-board. Two of the main activities of an autonomous robot are: processing the information gathered from the environment and select the most appropriate behavior to act in the environment. In the history of the RoboCup Humanoid League, there are many successful examples of autonomous humanoid robots performing complex tasks in a challenging environment, as the soccer game. Most of these robots needs relatively large computational resources to process the information gathered from the environment and to appropriately plan and execute their actions. For instance, the robots of Darmstadt Dribblers [12] mount an Intel PXA 255 at 512 MHz, the robots of NimbRo [11] mount a FSC Pocket Loox 720, which features a 520MHz XScale processor PXA-272 with 128MB RAM. In the end, the robots of the three-times World Champion TeamOsaka [13] use two CPUs: a GeodeLX 800 MHz with 256 MB of RAM and a Renesas SH2-7054 with 384 + 64 KB of RAM. The main CPU, i.e. the Geode LX 800, is used for image acquisition, image processing and robot behaviour control, while the second CPU, i.e. the SH2-7054, is used to generate the commands to be sent to the motors, to send the commands to the single joint motors and to close the loop on them. We present an autonomous platform that uses only this second one CPU (i.e. the Renesas SH2-7054) to completely control the robot from the image acquisition to the motor control. With our platform, image processing has been highly optimized in order to allow fast features extraction and information gathering. The vision module collects information that will be the input for the reasoning module that involves the development of behavior control. Complexity of soccer

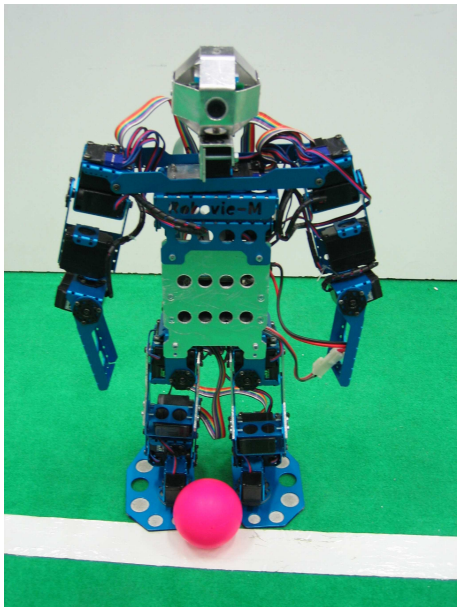


Fig. 1. Our version of Robovie-M.

make necessary playing with the development of complex behaviors, for example situations of coordination or different role assignment during the match. There are many types of behavior control, everyone with advantages and disadvantages: reactive control is the simplest way to make the robot playing, but do not permit more elaborated strategies as explained for example in [6]. On the other side behavior-based control, are more complex but more difficult to implement, and enables in general the possibility high-level behavior control, useful for showing very good performances [7]. Our approach includes the development of Finite State Machine (FSM) that permits a good work.

II. The Hardware Platfom

The Robovie-M platform of VStone is an embedded low cost system and has a low computatational power. It has been modified, as shown in Fig. 1, in order to be compliant with the rules of Humanoid Kid Size League, see [14].

Our Robovie-M has size of $480 \times 235 \times 70 \mathrm{mm}$ and a weight of 2.2kg. It is a fully autonomous humanoid robot that use as only sensor a camera.

Proceedings of the
Workshop on Humanoid Soccer Robots
of the 2006 IEEE-RAS International Conference on Humanoid Robots
Genoa, Italy, December 4, 2006
pp. 25-31
ISBN: 88-900426-2-1

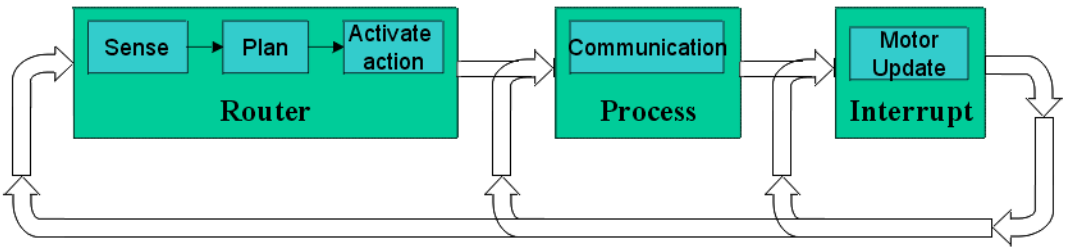


Fig. 2. The software architecture implemented in our software RobotCore.

Our robots has 22 degrees of freedom distributed as follow: six for each lower limb, four for each upper limb and two for the trunk; the actuators that move all DOF are servomotors. These actuators are directly controlled by the CPU with PWM signals.

Instead of the standard Robovie-M main board we mounted the VS–7054 board produced by VStone. The VS–7054 mounts a SH2–7054 MCU of Renesas running at 40Mhz with an internal FLASH memory of 384KByte and a RAM of 16KB. As external memory resources has got a RAM of $256\text{K} \times 16\text{bit}$ and a EEPROM of 64KB. This board can control a digital camera and the CPU provides just enough computational power to process image data, to control the behaviors of the robot and to output the signals for the 22 motors. However, the Renesas SH2–7054 MCU has been developed (and it is usually used) in automotive projects, so its computational power is quite limited and it does not provide a programming environment very friendly for the final user.

The digital camera is produced by OmniVision, the OV7620. This camera is mounted front looking and can be used in two resolution modes: low resolution (QVGA, i.e. 320x240) and high resolution (VGA, i.e. 640x480).

There is one additional internal sensor: an ADXL202E a two axes accelerometer produced by ANALOG DEVICES. This can sense, if the robot is standing or if it is laying down.

III. Software Architecture

The firmware of VS–7054 called *RobotCore* was developed by our team following the paradigm "*Sense, Plan and Act*" as defined in [8]: the robot senses the environment, plans the next actions and then acts, as shown in Fig. 3. This very simple architecture was mainly chosen because when the robot is moving the displacement of the camera is not controlled. So, in order to simplify the image processing software, the image is grabbed only when the robot is standing still.

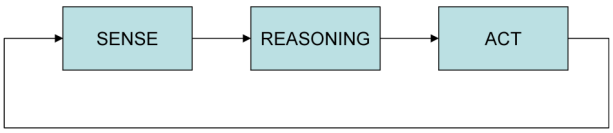


Fig. 3. The programming paradigm used in the software RobotCore.

The software architecture can be represented by three loops running at different speeds, as shown in Fig. 2. The inner loop controls the PWM motor output signals. These signals are obtained in asynchronous mode using interrupts. The medium loop include high speed functions as serial port and motor interpolation. The outer loop is the slowest one. In this loop is placed the behavior decision mechanism of robots. This module acquires and elaborates information on the surrounding environment, decides an action (accordingly to team task) and starts the action.

A. Vision System and Image Processing

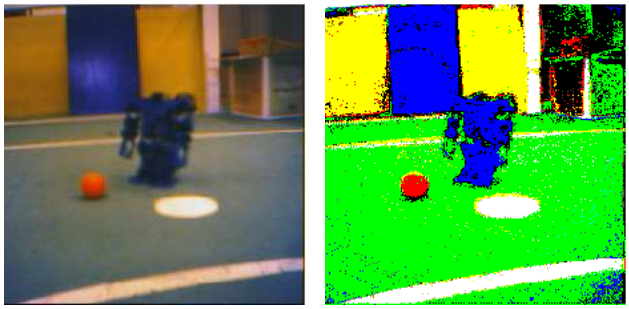
Even if up to now our robots are fitted only with perspective cameras, the vision system has been designed to allow the use of either perspective cameras or omnidirectional cameras.

1) Algorithm Flow.: The image processing algorithm proceeds as follows (Fig. 4):

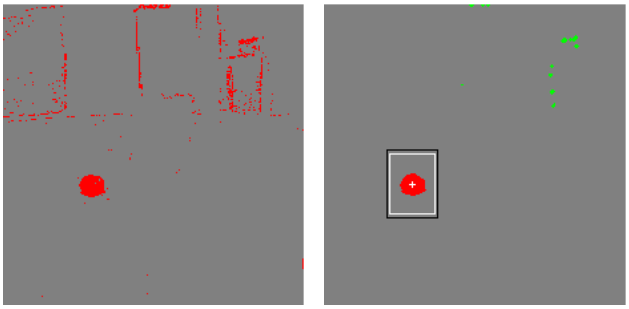
1) acquisition of the image, either in QVGA mode to acquire a complete frame covering the whole field of view of the robot's camera, or in VGA mode to acquire at higher resolution only the regions of interest (ROI) in which we want to focalize the attention;
2) demosaicing of Bayer pattern: for QVGA mode use *Periodic Reconstruction Interpolation*, for VGA mode using *Linear Interpolation with Laplacian Second–order Correction Terms*, as we described in [1];
3) color segmentation, with a look up table manually build offline and saved in EEPROM;
4) blob detection with labeling of connected components;
5) compute the centroids and the variance in two orthogonal direction of the pixel distributions of the ball, the goals, and the other robots.

B. High-Performance Image Processing

Usually, detecting several colors means creating several binary image maps as shown in Fig. 4 for the color red. In our case, as mentioned in section II, the computational power and the working memory are very limited, so we developed a way to extract the desired information from the image by keeping only three copies of the image in memory.In the following, we present our algorithm implementation.



(a) Acquire. (b) Segmentation.



(c) Blob detection. (d) Ball extraction.

Fig. 4. Example of image processing and features extraction of the ball.

1) Eroding: After the steps of acquisition, demosaicing, interpolation, and color quantization, we have a quantized image (i.e., every pixel gets a numerical label associated to one of the 8 colors of RoboCup). The *Erosion* procedure eliminate noise from the image by scanning for every pixel its its neighbors. If any of the neighbor pixels has a color not recognized as one of the 8 RoboCup colors (i.e., if any is labeled `color unknown`) the examined pixel is set to `unknown color`.

2) Dilating: To achieve a fast implementation, we do dilation and labeling in the same step, decreasing computational cost.

In procedure III.1 *existNotNullNeighbor(image1,i,j)* check if there is a not null and method *getNotNullNeighbor(blob,i,j)* gets the same color neighbor of processed pixel. When procedure ends, we get the labeled and quantized image. Last step consist of connecting labeled components.

3) Labeling of connected components: The image map that we have at this moment, contains group of pixel having the different label. The approach we adopted, extends the label of a colored pixel to all their neighbors with the same color. This way the output is an image sub standing this rule: for every pixel $i = 1...n \times n$ with color x has label j, every pixel with color equal to x has the same label j. The procedure is composed of two label propagation wipes: top-down and bottom-up.

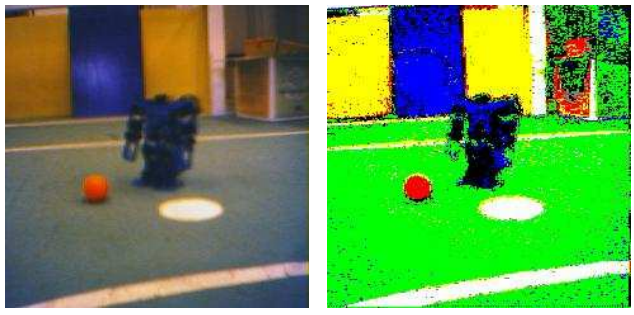
In this procedure we call a method to find the minimum label to extend, among pixel neighbors. This last operation in not so immediate and we had to develop some color consistency techniques.

Algorithm III.1 Dilation and labeling

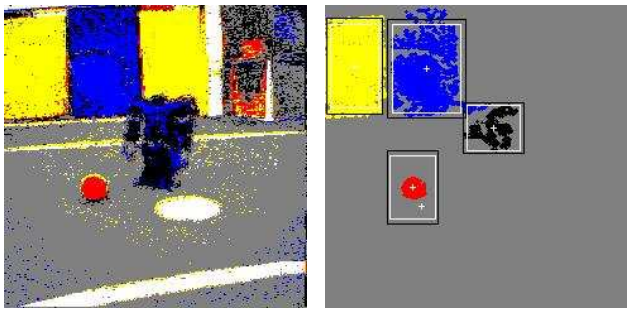
```
1: input: image1 ←quantized, eroded image
2: ouput: image2 ←quantized, eroded, dilated, labeled image
3: ouput: image3 ←quantized, eroded, dilated image
4: for i := 0 to width do
5:    for j := 0 to width do
6:       if (image1(i,j) <> UNKNOWN) then
7:          image2(i,j) := newLabel()
8:          image3(i,j) := image1(i,j)
9:       end if
10:      if (image1(i,j) := 0 ∧
         existNotNullNeighbor(image1,i,j)) then
11:         image2(i,j) := newLabel()
12:         pp = getNotNullNeighbor(blob,i,j)
13:         if (pp <> 0) then
14:            image3(i,j) := pp
15:            else image3(i,j) := 0
16:         end if
17:      end if
18:   end for
19: end for
```

C. Features estraction

Extracting information about blob is very fast with our implementation: we scan the image, and insert every pixel processed in the appropriate blob structure, that we had previously created. Then we read blob information directly accessing to structure parameters we are interested for. Some interesting information are color, area, center of blob, statistical variance.



(a) Original. (b) Quantized.



(c) Main colors. (d) Recognition.

Fig. 5. Example of image processing and features extraction of several colors.

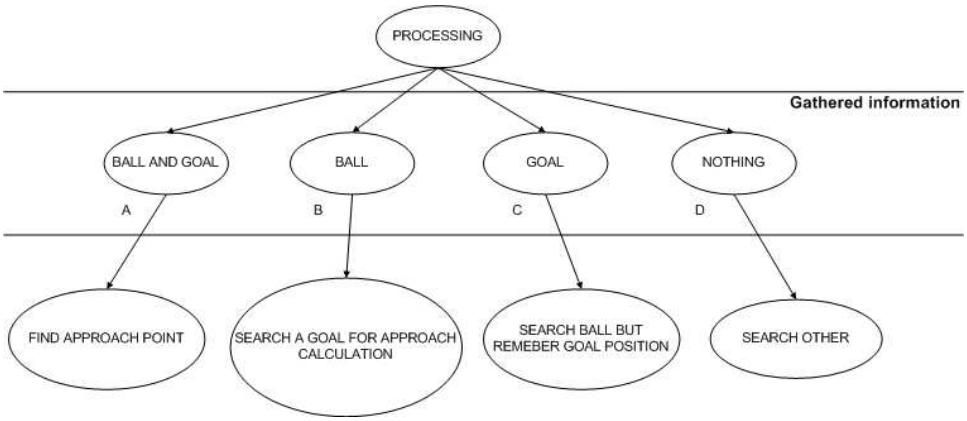


Fig. 6. Arbiter in context environment.

D. Complexity and time analysis

Given a square image composed by $n \times n$ pixels:

- Erosion: involves $n \times n$ pixel and a four-pixel kernel, the algorithm access every pixels five times in the worst case;
- Dilating and Labeling: every pixel is accessed five times, as before. But now we access to a five pixel kernel in another copy of the image to recover neighbor color of not-null pixels.
- Connect components: every pixel is accessed three times, but we have two phases, top-down and bottom-up.
- Blob detection: we simply collect every pixel and insert in the appropriate blob structure.

The resulting complexity is summarized in table I:

Operation	Complexity
Erosion	$O(5n^2)$
Dilating and Labeling	$O(5n^2+5m)$
Connecting components	$O(6n^2)$
Blob detection	$O(n^2)$

TABLE I

COMPLEXITY OF OUR ALGORITHM. N ARE GENERIC PIXELS, M THE "COLOR UNKNOWN" PIXELS.

For time analysis we aided with $HEW3$, the Software Development Kit used to write the firmware RobotCore. During the execution $HEW3$ gave the CPU cycles, so multiplying these with frequency clock (40Mhz), lead us to get the algorithm execution time. Obviously, the execution time depends on the features to extract, but we can say that in a general case the total processing of our algorithm involves about 6500000 cycles of clock , taking to a process time of $0,025\ s$. A great result considering our platform.

IV. BEHAVIOR SELECTION ARCHITECTURE

The behavior selection module has to determine the correct action to be taken accordingly to the measures of the sensors and to the team-play policy. Each robot has a set of basic movements (including walk, rotate, kick, lye down and stand up).

The deliberative behavior of the robot is based on this assumption: if the information extracted from the image processing are enough the robot can take an action, if not, it has to gather more information exploring the environment. For example if the behavior selection module does not have enough information to take its decision, then the robot move its head right or left to get more data from scene and then re-evaluate all acquired informations.

The behavior selection module logically stands and acts over a context selection. This method was proposed in [9]. Four different situations are provided depending on the quantity of information collected (i.e., on the features identified in the image): full information (ball and goals have been recognized), partial information (only a goal or only the ball have been detected) and lack of information (nothing has been recognized); the context selected gives a different behavior.

There are additional sets of complex movements that can be triggered only by a particular context such as peculiar game situations (e.g., throw in, penalty kick, etc.) or a particular role assumed by the robot: goalie, defender or attacker.

The context switching method enables to manage the complexity of dynamical selection of the behaviors. The effectiveness of this approach has been shown in [10]. Our aim was the development of a FSM represented in Fig. 7 being able to use the limited computational power of our platform.

V. EXTENDING THE SOFTWARE ARCHITECTURE

To improve both performance and robustness of the firmware developed by our team, the software architecture presented before has been extended with the introduction of an operating system layer. We provided the robot with a higher abstraction layer based on the functionalities of FreeRTOS's core, an opensource realtime microkernel widely used in embedded applications. Since no porting exists for the Renesas SH2 architecture uptonow, our team developed the HAL in order to use FreeRTOS on Robovie-M.

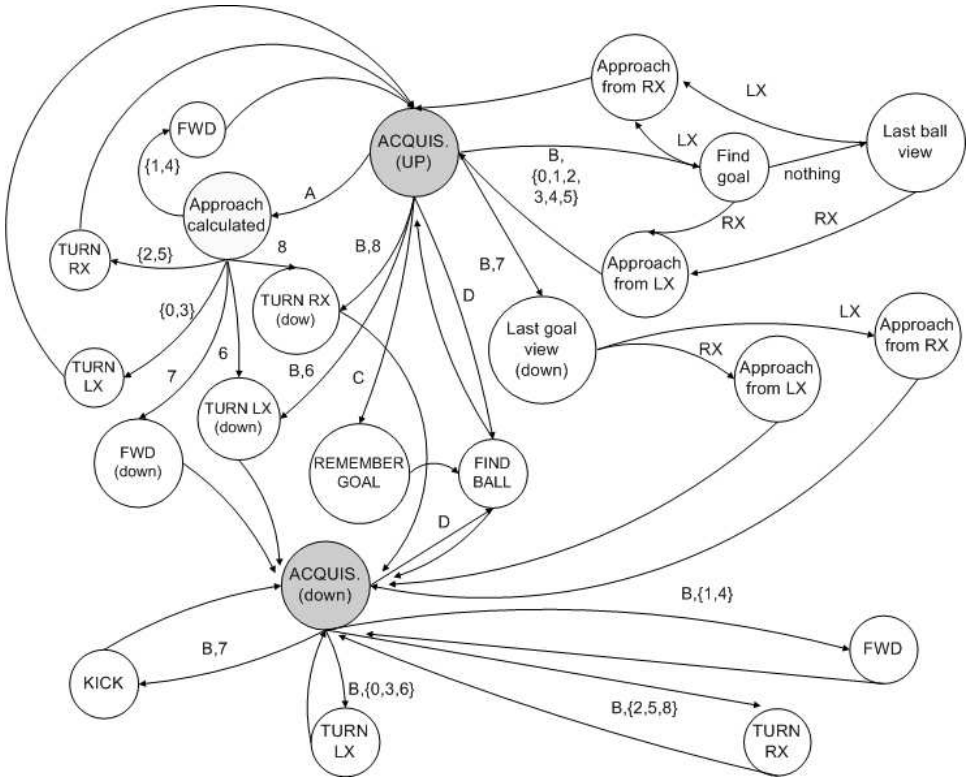


Fig. 7. The finite state machine implemented in RobotCore controlling the behaviour of the soccer robot.

The programming model offered by FreeRTOS comes from the theory of realtime systems: a set of tasks which run indipendently of each other, synchronized and communicating through the kernel's IPC services. The priority scheme adopted by the scheduling algorithm lets the programmer enhance flexibility in his/her code by using many different priority queues; also the debug phase can be riduced thanks to the tracing toolkit.

This software architecture is typical of embedded systems, where memory resources are limited and high performance is required by design specifications. The application level can interact with the hardware platform either passing through the OS layer or accessing the hardware by itself. This allows the programmer to balance between flexibility and efficiency in his/her code.

VI. Conclusion

In this paper, we introduced the software architecture implemented on our humanoid robots named Leonardo and Galileo. They are based on a Robovie-V platform by Vstone with 22 degrees of freedom and equipped with a low computational power CPU. The memory and computation power constraint fostered us to develop efficient image processing algorithms and a very fast behavior selection architecture. The software we presented in this paper was tested at RoboCup 2006 in Bremen, Germany. In Fig. 8, three different phases of the RoboCup competition.

VII. Acknowledge

We wish to thanks IT+Robotics S.r.l. for its sponsorship and its support. IT+Robotics is a spin–off of the University of Padua, Italy.

References

[] Cheng P. and LaValle S. M.: Reducing metric sensitivity in randomized trajectory design. In Procedings IEEE/RSJ Int'l Conference on Intelligent Robots and Systems, 43–48 (2001).

[1] Guseo, T.: *Architettura Software per Robot Umanoide Autonomo*, Tesi di laurea, University of Padova, Padova (2006) (In Italian).

[2] Guseo, T., Menegatti, E.: *Demosaicing Low Resolution QVGA Bayer Pattern for Focus of Attention in Humanoid Robots*, submitted to an international conference.

[3] Kuffner J.J., LaValle S.M.: RRT-Connect: An Efficient Approach to Single-Query Path Planning, In Procedings IEEE Int'l Conf. on Robotics and Automation, 995–1001, (2000).

[4] LaValle S.M.: Rapidly-Exploring Random Trees: A New Tool for Path Planning, TR 98–11, Computer Science Dept., Iowa State University, October (1998).

[5] Menegatti, E., Pagello, E., Wright, M.: Using Omnidirectional Vision within the Spatial Semantic Hierarchy. *ICRA 2002* 908–914, (2002).

[6] Sven Behnke and Raul Rojas. A hierarchy of reactive behaviors handles complexity. *In Balancing Reactivity and Social Deliberation in Multi Agent Systems*, volume 2103 of LNAI, pages 125-136. Springer, 2001.

[7] Sven Behnke. Playing soccer with humanoid robots, KI - *Zeitschrift Kuenstliche Intelligenz*, vol. 3, pp. 51-56, 2006.

[8] Murphy, R.R.: *Introduction to AI Robotics*, A Bradford Book, The MIT Press, Cambridge, Massachussetts, London, England (2000).

[9] Pagello, E., Montesello, F., Garelli, F., Candon, F., Chioetto, P., Griggio, S.,: Getting Global Performance through Local Information in *PaSo-Team'98 RoboCup 1998* 384–389 (1998).

[10] E.Pagello, A. D'Angelo, E. Menegatti Cooperation Issues and Distributed Sensing for Multi-Robot Systems IEEE Proceedings of IEEE Vol. 94 Iss. 7, July 2006, pp. 1370- 1383

(a) Dribbling challenge. (b) Goalkeeper. (c) Ball Search.

Fig. 8. Three different situations at RoboCup 2006 in Bremen.

[11] S. Behnke, J. Mller, M. Schreiber *Toni: A Soccer Playing Humanoid Robot* In I. Noda, A. Jacoff, A. Bredenfeld, and Y. Takahashi, editors, RoboCup-2005: Robot Soccer World Cup IX, pp. 59-70, Lecture Notes in Artificial Intelligence, LNAI 4020, Springer, 2006.

[12] M. Friedmann, J. Kiener, R. Kratz, T. Ludwig, S. Petters, M. Stelzer, O. von Stryk, D. Thomas Darmstadt Dribblers 2005: Humanoid Robot (Team Description Paper)

[13] URL: http://www.vstone.co.jp/top/products/robot/v2/

[14] RoboCup, compiled by Sven Behnke: *RoboCupSoccer Humanoid League Rules and Setup for the 2006 competition in Bremen, Germany*, URL: http://www.robocup.org/ (2006).