Embedded Real-Time Ball Detection Unit for the YABIRO Biped Robot

Markus Bader[1], Miguel Albero[2], Robert Sablatnig[1]
José E. Simó[2], Ginés Benet[2], Gregor Novak[3] and Francisco Blanes[2]

[1]Institute of Computer Aided Automation Pattern Recognition and Image Processing Group
Vienna University of Technology, Austria
e0026038@student.tuwien.ac.at, sab@prip.tuwien.ac.at

[2]Departamento de Informática de Sistemas y Computadores,
Universidad Politécnica de Valencia, Spain
mialgil@doctor.upv.es, (jsimo,gbenet,fblanes)@disca.upv.es

[3]Institute of Computer Technology,
Vienna University of Technology, Austria
novak@ict.tuwien.ac.at

Abstract — *Estimation of objects in a 3D space is a fundamental problem in computer vision and robotics. This paper describes an algorithm and its implementation for a vision module as a sensor of a biped robot (YABIRO). The vision sensor is able to estimate the position of spheres on an embedded platform. Objects are defined with their size and color in a model. The vision sensor detects the positions or at least the directions to the objects and stores them in a history. The algorithm includes a new voting system for detected objects, based on how trustable the detection was, and a new edge filter to terminate edges on the circle border for the circle detection. The systems frame rate depends on the area of interest and lies between 5 Hz and 20 Hz. With a mechanical size of 36x32 mm it is smaller than a matchbox and can also be used in different environments.*

1 Introduction

Before a robot can start to plan a path through an environment it has to know the environment. There are many ways how a robot can recognize or measure its environment. Active sensors like infrared, sonar or laser range finder are able to provide accurate data, but they are also slow and/or disturbing other systems while they are active. A camera system allows a robot to move in its environment without disturbing other robots. Furthermore the robot YABIRO [1] should be able to play in the *Humanoid League* at the *RobotCupSoccer 2006* where active sensing is forbidden [2]. Hence the major task for the vision sensor is to detect and measure the soccer ball. This task can be solved with a mono camera system like in [3], [4], [5] and [6]. Circle detection algorithms are well discovered, like in [7], [8],

[9], [10] and [11] . The following paper describes an object detection algorithm focused on spheric objects implemented on an embedded hardware with a new voting system to measure how trustable the detection of an object was. As well as a new color based edge filter to lower the computation power on the circle measurement will be described. An Analog Devices Blackfin BF533 DSP similar as in [3] and [12] is used to integrate the vision system in the size of 36x32 mm. The environment makes it also necessary to detect objects with a frame rate between 5 Hz and 20 Hz to give the robot enough time to react. The next section will describe briefly the related work. Sections 3 presents the YABIRO robot. Section 4 documents the embedded camera hardware and the used algorithm. The results are then presented in Section 5 followed by the conclusion.

2 Related Work

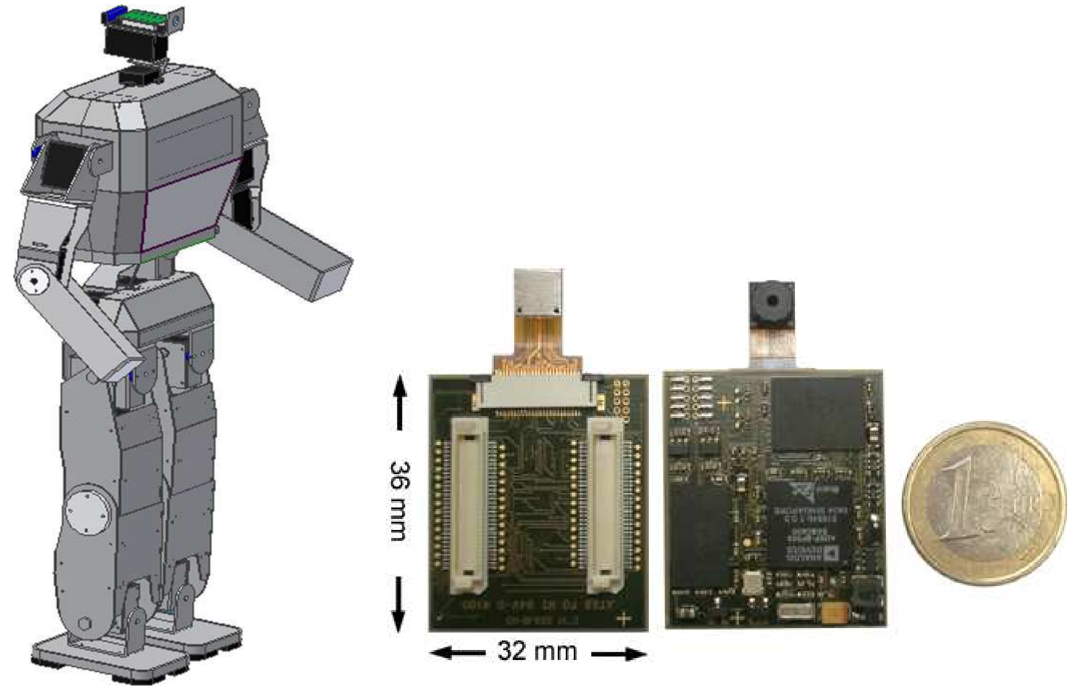
The related work can be divided into two groups. One Group represents the object detection algorithm implemented on a PC and the other the implementations based on other hardware. Implementations on PCs like [6], [4] and [5] are also focused on circle detection and are able to reach frame rates up to 20Hz by using sophisticated tracking systems and accurate shape detection algorithm. On the other hand there are systems like [13] which use a color blob base algorithm to find objects. This system works with a frame rate of 16.7Hz. [3] and [12] are presenting also implementation without a PC but they are using color and edge information to detect objects. Their implementation is limited to one spheric object in one single predefined color but it performs the detection with a frame rate of 60Hz. [3] and [12] where also focused building a low energy consuming device. The bottleneck on all reported hardware near systems is the limited cache.

3 YABIRO Robot

The YABIRO Robot (Figure 1.a) is a low cost biped robot designed to provide a robotic platform to design and test new control architectures. The difference to other humanoid robots like [14], [15], or [16] is that YABIRO uses intelligent servo nodes equipped with a CAN controller to communicate with the main control node by using a TDMA-CAN bus. Also the vision sensor is connected via this CAN bus to the distributed system. The main control node uses a Transmeta Crusoe control board as embedded-PC and a PIC18F458 microcontroller with a PC104-CAN controller as communication board to manage the real time CAN protocol[17]. The communication board has two main tasks: first, to adapt all the input messages from the embedded-PC to the TDMA-CAN bus and second, to guarantee and control the timing correctness for all the time slots on the bus.

4 Vision Sensor

The Vision Sensor is designed for object detection. It can detect different types of objects but we will focus on spheres. An object model is used to define objects with their real dimensions and colors. Based on this data the program fills a second model for shapes. The entries in the shape model are holding the thresholds and settings for the detection in the image. The shape form defines then the used search algorithm. The current algorithm is able to find and measure circles and to find rectangles. The idea is to segment the image into blobs by the colors of the defined objects. Then it performs the shape detections only



a) YABIRO robot b) Vision sensor

Figure 1: The image a) shows a general CAD view of the Robot with the vision sensor mounted on the top. Image b) shows the vision sensor with camera but without the CAN controller.

in the area in and around the blobs. The accuracy of a detected object is defined by the positive passed detection steps and tests. Detected objects are stored into a history which allows also to compensate incomplete measured objects. This paper describes in the following sections the 3D *ball detection* with the related 2D *circle detection*. Figure 2 shows the program data flow.

4.1 Hardware

The vision sensors hardware (Figure 1.b) is based a on Analog Devices Blackfin BF533 mounted on a Core Module with a 32MB SDRAM clocked rate 133 MHz, 2MB Flash and 64 kbyte SRAM/Cache. The used Omnivision OV7660 CMOS camera is directly assembled on the module and connected via PPI. [18] describes the hardware in more detail. A MCP2510 controller which supports CAN is connected via SPI to the Core Module to connect the vision module to the network on the robot.

The Blackfin processor does not provide a floating point unit. For this reason faster $atan2$ and $sqrt$ functions were implemented. This functions are only using integers and lookup tables. The new function *unsinged char arctan2i(short y, short x)* returns the angle in the domain of 0 to 255 where 0 represents 0 degrees and 255 an angle of $360 - 360/255$ degrees.

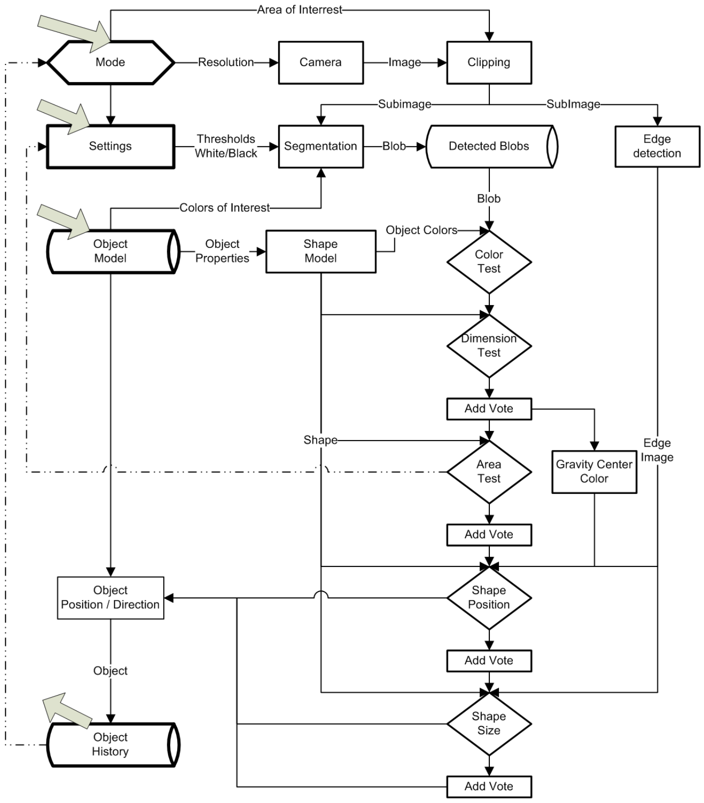


Figure 2: Data flow of the object detection. The large arrows are marking the in- and output.

4.2 Image Segmentation and Edge Detection

The camera provides the system with a UYVY color image I_{yuv}. This image will be reduced to 28 colors plus white and black and represented by 32 bit per pixel (4). This representation simplifies the comparison between colors and makes the image insensible to brightness changes. The edge detection is done in the same function as the color reduction to avoid two equal accesses to the input image. Most of the computing power of the controller is used by the edge detection. All edges are stored in an indexed list so that they are fast accessible. Equation (1) - (4) describes the color transformation of the input image. An opening and closing operation like in [19] is used to remove noise on all colors at once. We will name this image I_{c32}. An additional image I_{s32} will be also computed by using again an opening function on I_{c32}. This image will be then used later in the shape detection in filter 1.

$$b = \mathrm{Y} - \sqrt{((V-128)^2 + (U-128)^2)} \tag{1}$$

$$c = \begin{cases} 1, & \text{if } b < \epsilon_{black} \\ 2, & \text{if } b > \epsilon_{white} \\ \frac{arctan2i(u,v)}{10} + 3, & \text{else} \end{cases} \tag{2}$$

$$p = 0\text{x}00000001\text{h} << c \tag{3}$$

$$I_{i32} = |_{x=1}^{x=width-1}|_{y=1}^{y=height-1} p(I_{yuv}(x, y)) \tag{4}$$

Each bit in the 32 bit representation represents a color of the YUV spectrum, the zero bit represents no color information. Blobs are generated by using the 32 bit color information of the object in the model and the 32 bit color image I_{c32}. This blobs are then candidates for detected objects. A blob will be an object if the pixel count in the blob is over a defined threshold. The objects pass then the following four tests/detections: *dimension test, area test, shape position and shape size*. Each test votes the object.

4.3 Dimension and Area Test

Before a blob is used for further tests it has to match in color and dimension with a shape. The dimension test T_d for circles is shown in (5)

$$T_d = \begin{cases} \text{failed,} & \text{if } \frac{blob.height}{blob.width} > 2 \\ \text{failed,} & \text{if } \frac{blob.width}{blob.height} > 2 \\ \text{passed,} & \text{else} \end{cases} \tag{5}$$

The area test uses the relation between the number of the pixels in the blob and the area of the shape. Equation (6) shows the relation between the circle area and the pixel in the blob. The half width of the blob is used for the radius. Tests showed that the width has a better accuracy to estimate the radius than the height because the brightness level changes more with the height while the color in the horizontal is constant over the whole width. A P_a of 1 means that the shape matches probably with the color structure in the blob. This value can be used to redefine the camera black and white thresholds used in (2). Equation (6) and (7) describes the area test T_a .

$$P_a = \frac{\left(\frac{blob.width}{2}\right)^2 \times \pi}{blob.pixelcount} - 1 \tag{6}$$

$$T_a = \begin{cases} \text{failed,} & \text{if } |R_a| > \epsilon_{area} \\ \text{passed,} & \text{else} \end{cases} \tag{7}$$

If T_a and T_d are positive the current measurement of the shape gets voted by increasing the accuracy value.

4.4 Shape Position and Size

The dimension test T_d must be positive, while the area test can fail to perform the shape detection. The used algorithm is similar as the Hough Transform for Circles described in [11]. But line L_i are are only drawn[1] for edge in the blob with predefined border toward the gravity center of the blob color C_b. The circle center will be then represented as a peak where all the lines are intersecting. Equation (8) shows the line equation. We assume that C_b will be near the real circle center C_c. Hence it is only necessary to draw the lines in a small patch around C_b. The following two filters are used to skip edges which to not belong to the circle border.

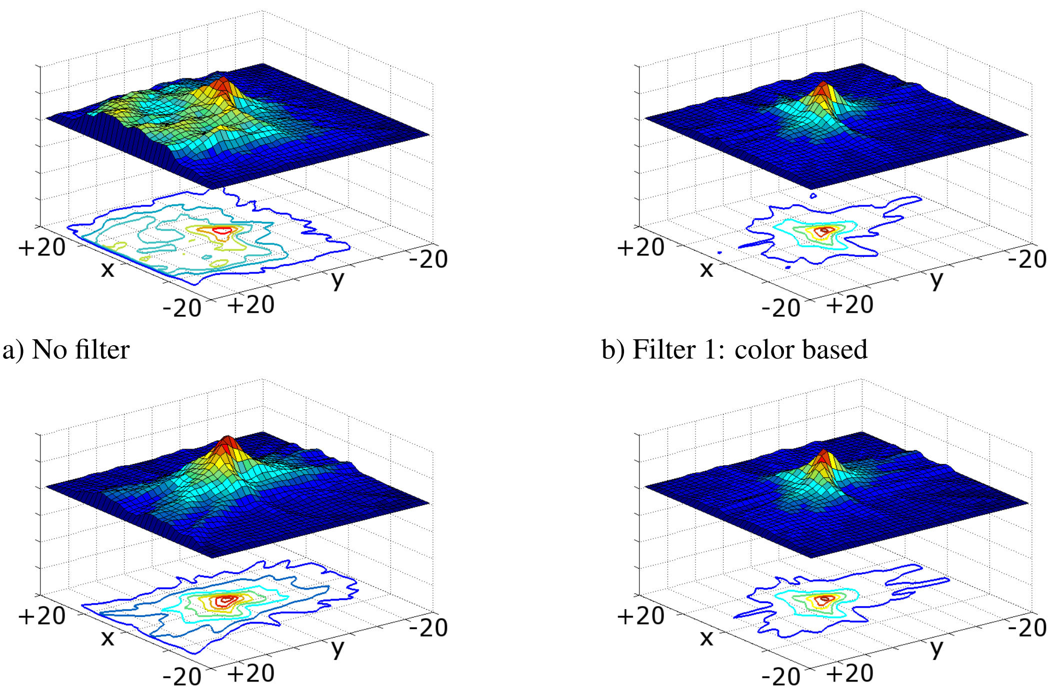
$$L_i = x_i + tan(\alpha_i) \times y_i \qquad \alpha_i \text{ ... gradient direction} \tag{8}$$

[1]The draw function adds the line and will not delete existing lines

An edge will be skipped if

- Filter 1:
  Edge is overlaid by a color entry
  $I_{s32}(edge.x, edge.y) \oplus S_{c32} > 0$
  I_{s32} is described in Section (4.2)
  S_{c32} represents the color of the shape in 32 bit format
  $\oplus$ .. bitwise AND
- Filter 2:
  Edge gradient does not look to the gravity center of the color
  $|tan(\alpha_i) - \arctan(x_i - x_b/y_i - y_b)| < \epsilon_{angel}$
  α_i ... gradient direction

Filter 1 will remove edges generated by fine structures on the ball which appear on spheres like golf balls. Figure 4 shows such spheres with edges inside the circle. The graph in Figure 3 shows the drawn patch around the C_b with different filters applied.



a) No filter

b) Filter 1: color based

c) Filter 2: edge direction based

d) Filter 1 and 2

Figure 3: Peaks formed by the lines L_i corresponding to the sphere number 2 in Figure 4 with different filters applied.

The shape center test T_c is used to vote the detection. Equation (9) shows when the T_c fails.

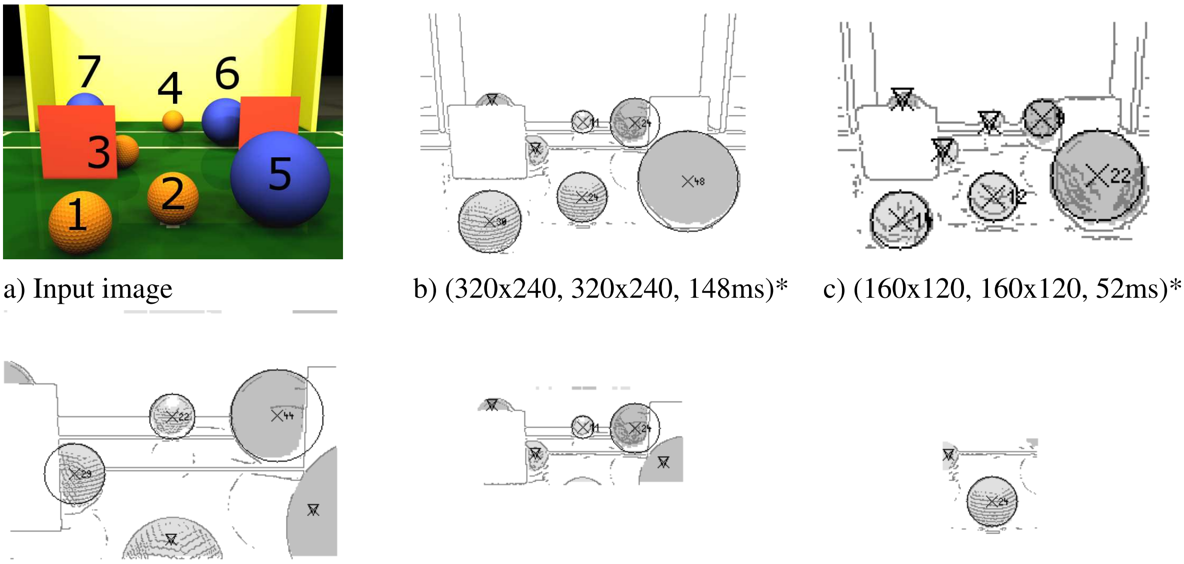
$$T_c = \begin{cases} \text{failed}, & \text{if } \Sigma\ Edges < \epsilon_{MinEdges} \\ \text{failed}, & \text{if } \frac{PeakSize}{\Sigma\ Edges} < \epsilon_{CenterPeakSize} \\ \text{passed}, & \text{else} \end{cases} \tag{9}$$

The radius can be detected as a peak in the histogram R. R is indexed by i where i represents the distances of used edges to the estimated center C_c. Similar to the center test T_r will be used to vote the shape measurement. Equation (10) shows when the T_r fails.

$$T_r = \begin{cases} \text{failed}, & \text{if } \Sigma\ PeakSize < \epsilon_{MinPeak} \\ \text{failed}, & \text{if } \frac{PeakSize}{\Sigma\ Edges} < \epsilon_{RadiusPeakSize} \\ \text{passed}, & \text{else} \end{cases} \tag{10}$$

4.5 Mode and Settings

The hardware can be used for different tasks which need different threshold settings. Because of that it is important to define modes for different tasks. Such a mode can an be used to search for a ball with a high resolution where it is necessary to set the threshold for the blob size to a low value. Another mode can be used to track a ball very fast so it is necessary to set the threshold for the blob size to a higher value. If the direction to the ball has a higher priority than the radius measurement the radius detection can be skipped. The mode defines also the resolution and the area of interest.



a) Input image　b) (320x240, 320x240, 148ms)*　c) (160x120, 160x120, 52ms)*

d) (640x480, 320x240, 133ms)*　e) (320x240, 200x100, 49ms)*　f) (320x240, 100x100, 23ms)*

Figure 4: Image a) shows spheres of two different types on a virtual generated *Robot-CupSoccer TeenSize* playground. The images b) to e) are representing the results with computation time by different resolutions and settings.
* (resolution, area of interest, computation time)

5 Results

The frame rate depends on the size of the area of interest and not on the image resolution. The algorithm supports 640x480 pixels, 320x240 pixels and 160x120 pixels but with a limited area of interest. Tests showed that the average lies around 5 Hz for a 320x240 pixels area of interest and 20 Hz for 160x120 pixels. If the image size gets smaller, the frame rate relays higher on the number of detected objects than on the area of interest (Table 5). The maximum area of interest is limited and defined to 320x240 pixels because of the cache memory. The computed distance to a ball depends on the detected radius. Tests showed that a radius under 20 pixel is not a good base to evaluate the distance but it can still be used to estimate the direction to the object. A direct comparison with other algorithms is not possible because they are performed on PCs or for a specific hardware. Table 5 shows the computation time needed and also the timing of the function used by the detection to compute the result for the images in Figure 4. The triangles in the images are pointing to blobs where the color matched with a shape but the accurate value was not high enough to represent them as a object.

	Figure 4.b	Figure 4.c	Figure 4.d	Figure 4.e	Figure 4.f
Resolution	320x240	160x120	640x240	320x240	320x240
Area of Interest	320x240	160x120	320x240	200x100	100x100
Capturing Image	2ms	3ms	9ms	2ms	2ms
Cleaning Data	2ms	0ms	6ms	2ms	2ms
Edge det.	68ms	18ms	67ms	19ms	10ms
Open/Closing	12ms	3ms	12ms	3ms	1ms
Blobs det.	27ms	7ms	28ms	7ms	4ms
Shape det.	39ms	24ms	20ms	18ms	6ms
Object det.	$<$1ms	$<$1ms	$<$1ms	$<$1ms	$<$1ms
Cycle time	148ms	52ms	133ms	49ms	23ms
Frame rate	6.8Hz	19.2Hz	7.5Hz	20.4Hz	43.5Hz

Table 1: Function timing of the vision sensor by using the virtual environment from Figure as input 4 with different resolutions and areas of interest.

6 Conclusion

The vision module can be used as a sensor to detect objects. The object history makes it possible to track objects or to solve measurement errors. The new proposed color base filter 1 detects edges which are belonging to a circle. This filter works similar efficient as the angle filter 2 as we can see in Figure 3. The reached frame rate makes the vision unit usable in the RobotCupSoccer environment. Furthermore the voting system allows to estimate objects even if the are not clear detected. The accuracy can be improved by using a stereo camera system. A higher frame rate can also be reached by using a dual core module like the BF561 because the bottleneck is found in the edge detection as we can see in Table 5. The dual core system allows to process the edge detection while the other core is performing the shape detection of the last frame. On that way the frame rate can be nearly twice as fast. Using the concept of this algorithm allows to implement additional shape detections so that the sensor can not only detect spheres.

References

[1] M. Albero, F.Blanes, G. Benet, J. Simo, and P. Perez. Yabiro, a new approach to small biped robots. In *Intelligent Autonomous Vehicles IAV-04*, pages 200–206, Lisbon (Portugal), 2004. Elsevier.

[2] Seven Behnke. *RobotCupSoccer Humanoid League Rules and Setup for the 2006 competition in Bremen, Germany*, January 2006.

[3] Stefan Mahlknecht, Roland Oberhammer, and Gregor Novak. A real-time image recognition system for tiny autonomous mobile robots. In *IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 324–330, 2004.

[4] S. Pagnottelli D. Scaramuzza and P. Valigi, editors. *Ball Detection and Predictive Ball Following Based on a Stereoscopic Vision System*. IEEE International Conference on Robotics and Automation, April 2005.

[5] T. D'Orazio, N. Ancona, G. Cicirelli, and M. Nitti. A ball detection algorithm for real soccer image sequences. In *ICPR '02: Proceedings of the 16 th International Conference on Pattern Recognition (ICPR'02) Volume 1*, page 10210, Washington, DC, USA, 2002. IEEE Computer Society.

[6] Xiao-Feng Tong, Han-Qing Lu, and Qing-Shan Liu. An effective and fast soccer ball detection and tracking method. In *ICPR '04: Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 4*, pages 795–798, Washington, DC, USA, 2004. IEEE Computer Society.

[7] Richard O. Duda and Peter E. Hart. Use of the Hough transformation to detect lines and curves in pictures. *j-CACM*, 15(1):11–15, jan 1972.

[8] C. Kimme, D. Ballard, and J. Sklansky. Finding circles by an array of accumulators. *Commun. ACM*, 18(2):120–122, 1975.

[9] Wing Hong W. Yu. Memory-efficient circle and ellipse detection algorithm in digital images. In *Systems, Man and Cybernetics, 1995. 'Intelligent Systems for the 21st Century'., IEEE International Conference on*, 1995.

[10] Ali Ajadri Rad, Karim Faez, and Navid Qaraqozlou. Fast circle detection using gradient pair vectors. In *VIIth Digital Image Computing: Techniques and Applications*, 2003.

[11] Pär Kierkegaard. A method for detection of circular arcs based on the hough transform. *Mach. Vision Appl.*, 5(4):249–263, 1992.

[12] Gregor Novak, Abdul Bais, and Stefan Mahlknecht. Simple stereo vision system for real-time object recognition for an autonomous mobile robot. In *IEEE International Conference on Computational Cybernetics (ICCC2004)*, pages 213 – 216, 2004.

[13] Anthony Rowe, Chuck Rosenberg, and Illah Nourbakhsh. A low cost embedded color vision system. In *Proceedings of IROS 2002*, 2002.

[14] Fuminori Yamasaki, Tatsuya Matsui, Takahiro Miyashita, and Hiroaki Kitano. Pino the humanoid: A basic architecture. In *RoboCup 2000: Robot Soccer World Cup IV*, Lecture Notes in Artificial Intelligence, Vol. 2019, pages 269–278. Springer, 2000.

[15] Takuya Saito. Wasedanian robot, presented at the robocup 2004 humanoid league. lisbon (portugal), june 27 - july 5, 2004.

[16] H.H. Ngia, T.Stessh, C. Chee, and H. Geok. Rope robot, presented at the robocup 2004 humanoid league, lisbon, portugal june 27 - july 5, 2004.

[17] P. Perez, G. Benet, F. Blanes, and J.E. Simo. Communications jitter influence on control loops using protocols for distributed real-time systems on can bus. In *5th IFAC International Symposium on Intelligent Components and Instruments for Control Applications. SICICA03*, Aveiro (Portugal), 2003. Elsevier.

[18] Bluetechnix Mechatronische Systeme GmbH, Waidhausenstr. 3/19, A-1140 Vienna, AUSTRIA/EUROPE. *Hardware User Manual CM-BF533 V2.0*, 12 2005.

[19] Gregory A. Baxes. *Digital image processing: principles and applications*. John Wiley & Sons, Inc., New York, NY, USA, 1994.