View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Universidade do Minho: RepositoriUM

Vision, Kinematics and Game strategy in Multi-Robot Systems like MSL RoboCup

Fernando Ribeiro, Ivo Moutinho, Pedro Silva, Carlos Fraga, Nino Pereira

Grupo de Automação e Robótica, Departamento de Electrónica Industrial,
Universidade do Minho, Campus de Azurém, 4800 Guimarães, Portugal
fernando.ribeiro@dei.uminho.pt, ivo@aeiou.pt,
pedro_silva@portugalmail.com, carlosfraga@portugalmail.pt, ninopereira@clix.pt

Abstract. In Multi-Robot systems like the RoboCup football challenge, there are a small number of key issues which are of extreme relevance for the successfulness of the final application. In MSL RoboCup these main issues are three: a) The vision system, which has to be as reliable and fast as possible in order to perceive the necessary entities to carry out the game actions; b) Correct kinematics of the robot, that makes the robots move towards the desired goal in the fastest, shortest and optimized away; c) Game strategy, which needs collaboration and communication between all the agents in the field. Other issues are also important but these three consist of the fundamental ones towards the next step in this challenge which is ball pass between the robots in a controlled way. A team of robots will only be able to pass the ball to another robot only when these three issues are sorted out. This paper describes how these three issues were tackled by the MINHO team and shows their next directions.

1. Introduction

Although many teams prefer to buy a standard off the shelf robotic platform and implement some changes in hardware/software, Minho team which participates on RoboCup since 1999, builds its own platforms from scratch. Being part of an Industrial Electronics department they build the mechanics, hardware and software, bearing in mind a very low budget. This continuous participation in RoboCup led to some new developments in many fields.
The next step of this team is to pass the ball between robots. In order to achieve this, three key issues need to be sorted out: the vision system, correct kinematics of the robot and game strategy which implies collaboration and communication between the agents in the field. These issues are described in the following sections of this paper.

2. MINHO Team Robots Description

A short and very brief description of the MINHO team robots is made in this section. Since the robots were completely developed within our labs, they were planned and drawn in a CAD system in order to check its assemblability (see **Fig. 1**) and only then they were built. Some mistakes could be prevented by drawing them first.

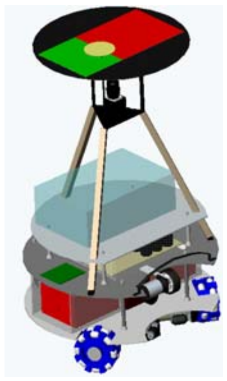
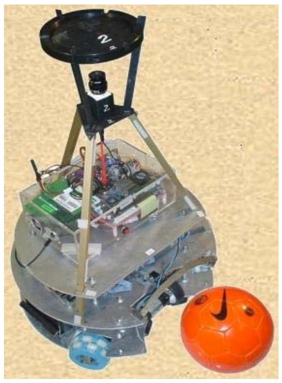

Fig. 1. CAD drawing and final robot

The locomotion of these robots is achieved by three omni wheels, coupled to D.C. geared motors with brushes, 5Nm and 33 Watts from Crouzet. Encoders are coupled to the motor veins in order to count the number of turns.
The robots have a ball handler made with a DC motor coupled to a rubber wheel which when in contact with the ball it starts up and pulls the ball towards the robot.
In order to kick the ball a magnetic kicker was developed and is used. It proved to be very strong and it consists of a magnetic coil with a movable iron core which is pushed towards the coil centre when electric current passes through the coil.
The Computer used is a low consumption motherboard VIA EPIA M + C3 running at 933 MHz, with 256 Mbytes of P266 memory and a 2.5" IDE Flash Drive with 256 Mbytes. This motherboard has a VGA board embedded, supports USB 2.0, it has 1 parallel port (from where the outputs are sent to the motor controllers), 1 serial port and a PCI slot. A two PCI raiser is used in order to connect the two boards (wireless Network board and the frame grabber). The operating system used is Linux Mandrake and the whole software is written is C language.
The vision head is placed on the highest position to increase the field of view. It consists of a colour camera facing upwards onto a spherical mirror facing downwards. The image is distorted but the software reshapes it to a planar image.
The hardware to control the motors, kicker and ball handler was completely developed in our lab.

3. Three Wheels Drive

There exists a great variety of ways to move across a solid surface by mobile robots. The most important are wheels, tracks and legs [1]. Wheels are the most used since they offer simpler mechanics and construction easiness.
Most teams use two wheels drives with some caster wheels and control them with a differential drive technique. Many teams including Minho used the two wheels drive [2][3]. The CMU team [4] used a two-wheeled drive unit with a passive trailer. Other teams prefer to use steering on some wheels like the Sharif team [5]. Philips team [6]

uses a four wheels drive and four wheels steering. The Matto team [7] used four pairs of omni wheels, each pair being driven by a unique DC-motor. The Artisti Veneti team also used an holonomic platform as described in [8].
This three Omni directional wheels solution adopted since last year by this team significantly reduces the robot's reaction time due to the rich manoeuvrability, it simplifies the game strategy and the motor control algorithm is simple. This type of wheel has small rollers to allow the wheels to move freely on any direction. They move along the primary diameter, just as any other wheel. Though, the smaller rollers along the outside of this diameter allow free rotation along an orthogonal direction to the powered rotation. The mechanical construction is shown in **Fig. 2**.

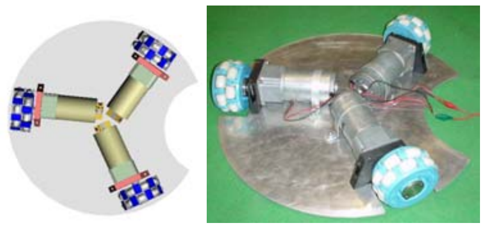

Fig. 2. Three-Wheel drive mechanical construction (design and physical)

On the left image, the grey circle represents the robotic platform, and the three motors coupled to the omni wheels are mounted with 120 degree between them, aligned like in an equilateral triangle so that their axis intersect at the robot centre. In the centre it can be seen the specially built encoders coupled to each motor axis.

4. Kinematics of Minho Robotic Platform

The inverse kinematics model is simple. It was considered that the representative coordinates of the robot were located in its centre. Each wheel is placed in such orientation that its axis of rotation points towards the centre of the robot and there is an angle of 120º between the wheels. The velocity vector generated by each wheel is represented on **Fig. 3**-b by an arrow and their direction relative to the Yr coordinate (or robot front direction) are 150º, 30º and 270º respectively.

4.1 Linear Movement

Total platform displacement is achieved by summing the three vectors contributions:

$$\vec{F_T} = \vec{F_A} + \vec{F_B} + \vec{F_C} \tag{1}$$

A software simulator was built and is depicted in **Fig. 3**. The user inputs three variables (linear speed, linear direction and angular speed) and the program outputs each motor contribution. For now, only linear speed is described.

First of all, some definitions need to be considered. **Fig. 3**-a) represents the diagram of the mobile robot platform with the three wheels. The robot front side is in the Yr direction and represents 0 degrees (positive side to its left). The three wheels coupled to the motors are mounted at angle position +60, -60 and +180 degrees respectively. It is important to remember that the wheel driving direction is perpendicular to the motor axis (therefore 90 degrees more). The line of movement for each wheel (when driven by the motor and ignoring sliding forces) is represented in **Fig. 3**-b) by the segments A, B and C. The arrow indicates positive direction contribution.
The total platform displacement is the sum of three vector components (one per motor) and is represented as a vector in the platform body centre. In **Fig. 3**-c) it is depicted a vector representing the desired movement; the angle α represents the direction and the vector length represents the velocity. In order to find out the three independent motor contributions, this vector is projected on A, B and C axis representing the line of movement of each wheel. **Fig. 3**-d) shows the projections that represent the three vector components of the contributions. The vectors can have a positive or negative direction which represents the direction in which the motor has to move (forward or backwards respectively).

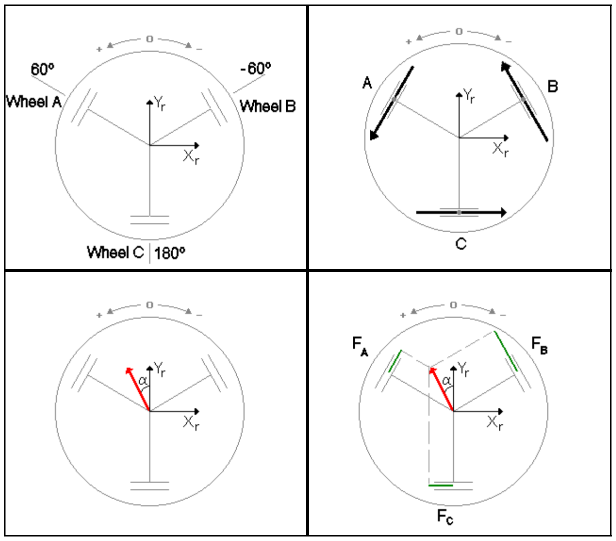

Fig. 3. Graphical representation of motor contribution; a) platform motors/wheels distribution; b) wheels driving axis; c) desired movement; d) motor contributions

Since the robot forward direction is represented by Y_r, each motor contribution consists of the cosine of the angle α (*DesiredDirection*) projected on each wheel drive direction, multiplied by the velocity, given by:

$$F_n = velocity \cdot \cos(WheelDriveDirection_n - DesiredDirection) \qquad (2)$$

Considering that the three wheels driving directions of this robot are 150, 30 and 270 degrees respectively, the contribution for each motor for linear velocity is given by:

$$F_A = velocity \cdot \cos(150 - DesiredDirection) \quad (3)$$

$$F_B = velocity \cdot \cos(30 - DesiredDirection) \quad (4)$$

$$F_C = velocity \cdot \cos(270 - DesiredDirection) \quad (5)$$

Where: *F* - is the motor vector contribution
A, B, C – represents the motors
Velocity - is the linear velocity the robot should move
DesiredDirection - is the angle α of the desired movement

4.2 Angular Movement

Considering now angular movements, and assuming accurate wheels alignment, pure rotation over its centre can be achieved by driving all wheels in the same direction at the same speed. The angular velocity is the linear peripheral speed of the wheels divided by the radius of the robot. **Fig. 4** still applies for angular velocity. Once again, the positive values make the robot rotate to its left and negative values to its right.

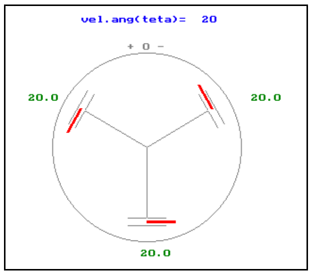

Fig. 4. Graphical representation of motor contributions for a positive angular velocity

4.3 Mixed Linear and Angular Movement

Should the robot need to rotate its body while moving towards the ball, linear and angular velocity can be combined by calculating the sum of both contributions. In **Fig. 5** a linear and angular movement is described. Added to a typical linear velocity (as described in **Fig. 3**) an angular velocity contribution is computed by adding the two vectors. On the left side the two contributions are separated and on the right side only the final value is represented.
It is important to point out that this movement is always relative to the robot centre. By adding both linear and angular contributions, speeds over the motor maximum can happen, but this saturation is avoided by limiting the maximum sum between linear and angular velocity, in which case, priority is given to angular over linear velocity.

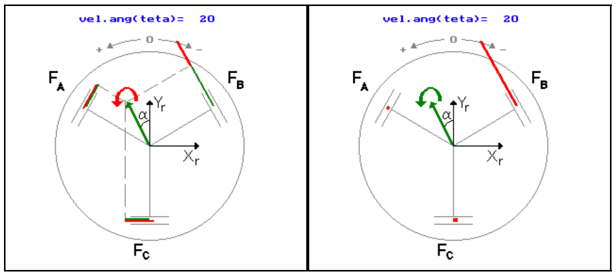

Fig. 5. Combined linear and angular velocity

Tests on the motors were carried out with no load. A maximum speed achieved was 170 rpm. With 100mm diameter wheels the maximum linear speed is about 0,9 m/s.

5. Motor Control

PID control is by far the widest type of automatic control used, probably because it is very simple and easily implemented. Our approach used the ideal PID algorithm [9]:

$$y(t) = Kp\left(x(t) + \frac{1}{Ti}\int_0^t x(\tau)d\tau + Td\frac{dx(t)}{dt} \right) \qquad \textbf{(6)}$$

The transfer function of our system was unknown, so a trial and error approach was carried out to determine the best type of control and the optimal parameters. Several experiments were carried out to optimise rotational movement. First, only proportional gain was implemented, and then added the integral gain, and finally the differential gain. The robot started from 70º position and rotated to a reference value of 180º.

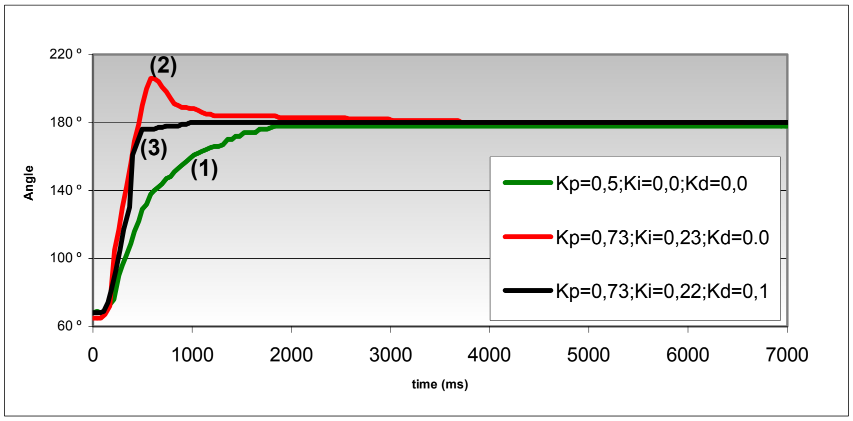

Fig. 6. Rotational movement with different Kp, Ki and Kd parameters

As shown in **Fig. 6**, integral gain (2) decreases the rise time and eliminate the s-s error, but increases the overshoot and the setting time. To overcome this, a derivative component (3) was added, and the desired effect was attained: no overshoot, no s-s error, fast rise time and settling time.
A very fast and accurate response was obtained with PID control. The time spent to rotate from 70º to 180º is 900ms.

6. Vision System

In literature different mirror geometries have been proposed. Conic, Parabolic, Spherical and Hyperbolic are only some examples. Some descriptions of omni directional vision sensors can be read in [10] and [11], and some RoboCup teams usage can be read in [12] and many others. Marchese and Sorrenti even suggest a multi-part mirror on [13]. Many teams on RoboCup use omni directional vision systems and Minho team is no exception. This year a new mixed shape hyperbolic/conic mirror is used. It makes the image slightly smaller but with more area captured. The conic part of the mirror avoids having the farther objects to look so small. For strategy purposes, this increases the amount of perception, simplifying the programming task and giving much more reliable and practical behaviours to the robots.

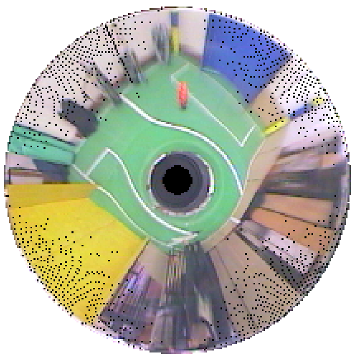

Fig. 7 Circular image seen by MINHO robots

Fig. 7 shows the image acquired by Minho vision system. The mirror still has some defects since the technique used to make it is not very accurate but the final effect can be seen (new mirrors will be built). The image is rounded and centred on the robot itself, making part of the image upside down. The calculi to get some perception out of the image need to use some trigonometry, which is normally relatively slow.
In order to simplify the strategy algorithm it was decided to transform the image onto a planar one as if the robot sees from inside a cut cylinder and all around it. The final image after transformation is shown in **Fig. 8**, where image is easily perceptible. The X coordinate corresponds to angle direction and Y coordinate corresponds to distance

(although not directly proportional, but simply calculated knowing the mirror shape). Egocentric spatial representation is achieved.

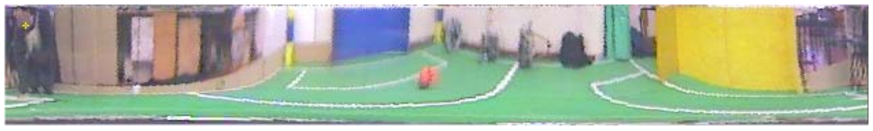
Fig. 8 Resulting image after shape transformation

6.1 Mathematical Transformation

The mathematical transformation is simple and consists of converting polar coordinates into Cartesian coordinates as **Fig. 9** shows.

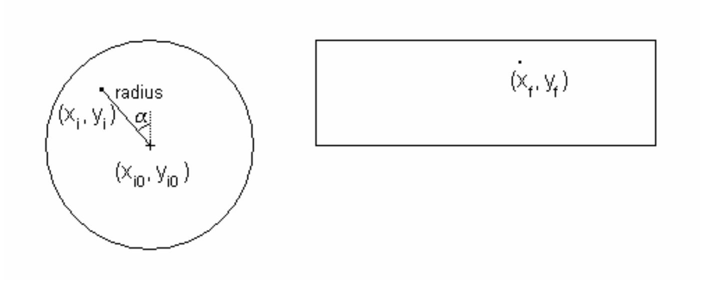

Fig. 9 Mathematical transformation from round to rectangular image

The mathematical formulae are described by:

$$\begin{aligned} x_f &= x_{i0} - radius \cdot \sin(\alpha + 180) \\ y_f &= y_{i0} - radius \cdot \cos(\alpha + 180) \end{aligned} \tag{7}$$

where	X_i, Y_i	- initial coordinates on the round image
	X_{i0}, Y_{i0}	- coordinates of the image centre
	α	- angle of the pixel
	radius	- distance of the pixel to the image centre
	X_f, Y_f	- final Cartesian coordinates

The reason for the 180 sum up to α, is to make the front of the robot appear in the centre of the image. The reason to mirror the image viewed (with the minus sign) is to remove the mirror effect of the image so that the left side of the image corresponds to the left side of the field as seen from above the robot.
But these calculi use some trigonometry and it can take some time to do it, since they need to be used for each pixel. If maximum number of frames is to be achieved and maintained, the software code needs to be much optimized.

6.2 Software Optimization

This image transformation and game entities perception has to be carried out in real time and the number of frames is very critical for a quality game. Therefore, optimization was tackled very carefully and to the maximum extent. In order to transform the image with these formulae, some tests were carried out to check the time it would take.

The rounded image area really used consists of a radius of 105 pixels and ignores a 9 pixels radius in the centre of the image where the robot sees itself (see **Fig. 7**). The perimeter is to be spread over 630 pixels (length of the final rectangle). Therefore, the final resulting shape is a rectangle of 630 pixels width by 96 pixels height, making a total number of 60480 pixels. With the microprocessor VIA mini-itx running at 933MHz, it takes around about 350ms to calculate the coordinates of the 60480 pixels, making it impracticable for a real time vision system, just to show the image on the computer screen with the right shape. This time is approximate since the operating system in use (Linux Mandrake) is not real time.

Therefore, when programming the image grabbing software, a structure was created containing the memory address where the frame grabber places the pixel and the memory address on the screen where the pixel should be after the shape transformation. The structure is very simple and given by:

```
typedef struct
{
  int *video_addr, *screen_addr;
} IMG_PLANA;
IMG_PLANA img[IMG_MAX_PIX];
```

The IMG_MAX_PIX variable consists of the total number of pixels of the rectangle (630x96) which is in this case 60480 pixels. Then, a routine to fill in this array is called only once at the beginning of the computer program and applying the mathematical formulas described above. After the array is fill in, each time an image is grabbed, the array is scanned to copy the values grabbed to the right screen coordinate.

For the value of IMG_MAX_PIX equal as 60480 this routine takes 6,25 ms to be performed. Once a frame is to be grabbed, an *ioctl* instruction is called and the microprocessor is immediately released so that other instructions can be carried out. This means that while one frame is being grabbed another frame can be processed. The time to grab one frame is 20ms and therefore the 6,25ms time to copy one frame to the screen is much less than the 20ms. This means that the time needed to change the image from a rounded shape to a rectangular shape can be neglected, since it is carried out during the following grab. In fact, two frames (even and odd) are grabbed (one at a time). The number of frames achieved (without any image processing routines) is 50 frames per second after transformation.

The memory space used by this array is 60480 pixels times 2 memory pointers times 4 bytes (32 bits each pixel) which makes it 483840 bytes. It is a value perfectly acceptable nowadays since the computer has 256Mb Memory.

As previously said, the X coordinate of the new image corresponds to the object angle related to the robot front, and the Y coordinate corresponds to object distance. **Fig. 10** shows an example of an image grabbed where both goals and ball are visible. It is easily calculated the angle between the robot and the ball (or any other entity), as well as the distance between the robot and the ball (or any other entity).

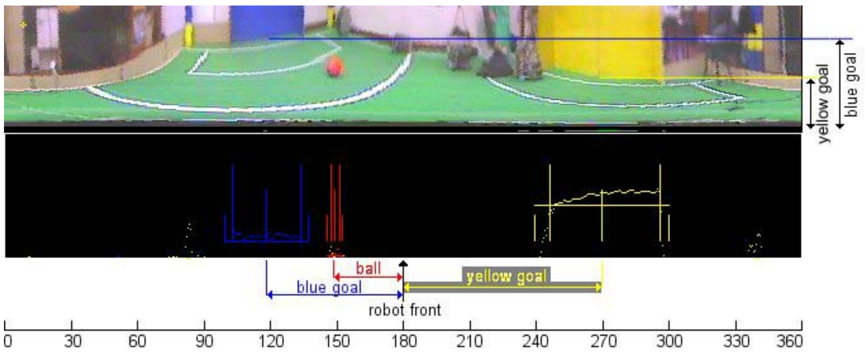

Fig. 10 Entities coordinates extraction

7. Game Strategy

Time has come for the robot players start passing the ball, make nicely planned plays and not just play individually. The facility the robots have to communicate between them through a wireless network board makes it easy to broadcast a structure of information to their team mates, which includes their location. Knowing their location they can start passing the ball.

This team defined four different Player Positions in the field (or types of behaviour): *Goal Keeper*, *Defender*, *Forward* and *Second Forward*. Although the first three types are easily understandable the last one is new. The *Second Forward* does not behave as a forward otherwise two players would go towards the ball. It consists of a player which positions itself in a clear line between the opponent goal and the Forward player. If the Forward has no clear line to kick to the goal, it will pass the ball to this Second Forward which will do that. This new type of player will not run after the ball but it will position itself in an easy location to score. **Fig. 11** shows a scenario where opponent players A and B tend to move towards the ball and the red lines on the field represent the clear space for ball pass between two player of the same team. It is important to say that the kick strength is controllable. It kicks 31 different levels of strength, varying from 1 (ball does not move) to 31 (ball travels about 40 meters).

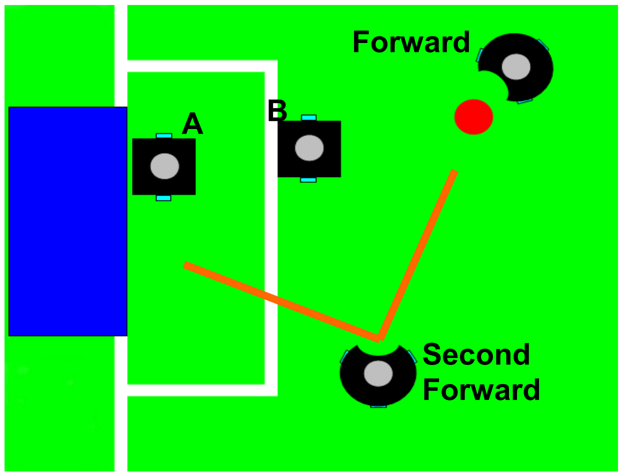

Fig. 11. Second Forward player position and lines clear of obstacles

Sometimes robots have to be taken out of the field either for faults/cards or for break down. This team uses the number of players on the field to influence their behaviour on the field, like in a real human football game. For example, if only one robot player is on the field, it becomes a goalkeeper. With two players on the field, one will be Goal Keeper and the other forward. With three players, the third will become Defender and with four, the fourth will become a Second Forward. The order of behaviour is then: 1) Goal Keeper, 2) Forward, 3) Defender and 4) Second Forward.

8. Conclusions

In RoboCup football games, the time a robot takes to reach the ball is of extreme importance. The faster it gets the ball the more chances it has to score. With the 3 wheel drive configuration described in this paper a robot can move in a straight line. The control software is simple and efficient. According to the direction angle and desired speed only three values are calculated. The PWM to control the motors is generated by a PIC, leaving the computer processor free for other more complex tasks like the image processing and the game strategy. Most time the motors do not drive at their maximum speed leaving a tolerance for when linear and angular speeds are required at the same time.
This configuration allows linear and angular speeds at the same time and this is of extreme importance for this team since each robot carries a fixed position kicker. If the kicker is not in the robot moving direction an angular speed needs to be used together with the linear speed while the robot moves towards the ball, in order to point the kicker to the right direction.
The Minho Team vision system is described. The image seen by the hyperbolic/conic mirror is transformed onto a planar shape and only then game entities are percepted and sent to the strategy software. The grabbing system mathematical transformation is

described as well as the optimization needed to make it in real time. Optimization allows the transformed image being displayed on the screen in less time than the actual grabbing, allowing a total of 50 frames per second before running any image processing routines. Having the image in a planar form makes it easy to implement game strategy since only Cartesian coordinates are used. The X coordinate corresponds to the object angle related to the robot front, and the Y coordinate corresponds to object distance.

References

1. Joseph L. Jones and Anita M. Flynn, “Mobile Robots – Inspiration to Implementation”, A K Peters, Wellesley, Massuchusetts, ISBN 1-56881-011-3, page 139.
2. C. Machado, I. Costa, S. Sampaio, e F. Ribeiro, “Robotic Football Team from MINHO University”, in RoboCup-99: Robot Soccer World Cup III", M. Veloso, E. Pagello, H. Kitano (eds), Springer-Verlag, Berlim, Alemanha, 2000.
3. Fernando Ribeiro, Carlos Machado, Sérgio Sampaio, Bruno Martins, "MINHO robot football team for 2001", in “RoboCup 2001: Robot Soccer World Cup V”, Andreas Birk, Silvia Coradeschi, Satoshi Tadokoro (eds), Springer, LNAI 2377, Berlim, 2002, page 657-660.
4. Steve Stancliff, Ravi Balasubramanian, Tucker Balch, Rosemary Emery, Kevin Sikorski and Ashley Stroup, "CMU Hammerheads 2001 Team Description", in “RoboCup 2001: Robot Soccer World Cup V”, Andreas Birk, Silvia Coradeschi, Satoshi Tadokoro (eds), Springer, LNAI 2377, Berlim, 2002, page 631-634.
5. M. Jamzad, H. Chitsaz, A. Foroughnassirai, R. Ghorbani, M. Kazemi, V.S.Mirrokni, B.S.Sadjad, "Basic Requirements for a Teamwork in Middle Size RoboCup", in “RoboCup 2001: Robot Soccer World Cup V”, Andreas Birk, Silvia Coradeschi, Satoshi Tadokoro (eds), Springer, LNAI 2377, Berlim, 2002, page 621-626.
6. http://www.cft.philips.com/html/subExamplesAppl/pdf/robocup.pdf
7. Kosei Demura, Kenji Miwa, Hiroki Igarashi and Daitoshi Ishihara, “The Concept of Matto”, in RoboCup-99: Robot Soccer World Cup III", M. Veloso, E. Pagello, H. Kitano (eds), Springer-Verlag, Berlim, Alemanha, 2000, pages 723-726.
8. E. Pagello, M. Bert, M. Barbon, E. Menegatti, C. Moroni, C. Pellizzari, D. Spagnoli and S. Zaffallon, "Artisti Veneti: An Heterogeneous Robot Team for the 2001 Middle-Size League", in “RoboCup 2001: Robot Soccer World Cup V”, Andreas Birk, Silvia Coradeschi, Satoshi Tadokoro (eds), Springer, LNAI 2377, Berlim, 2002, page 616-619.
9. Aström, K.J. and Hägglund, T., PID Control. Theory, Design and Tuning. 2nd ed., (Instrument Society of America, Research Triangle Park, NC, 1995).
10. Y. Yagi, S. Kawato, S. Tsuji, "Real-time omni-directional image sensor (COPIS) for visionguided navigation", IEEE Trans. on Robotics and Automation, Vol. 10, N. 1, pp. 11-22, 1994
11. P. Lima, A. Bonarini, C. Machado, F. M. Marchese, C. Marques, F. Ribeiro, D. G. Sorrenti, “Omni-Directional Catadioptric Vision for Soccer Robots”, Special Issue of the Robotics and Autonomous Systems Journal, Elsevier, Amesterdam, Vol 36, nº 2, 31st Augout 2001.
12. A. Bonarini, P. Aliverti, M. Lucioni, "An omni-directional sensor for fast tracking for mobile robots", Proc. of the 1999 IEEE Instrumentation and Measurement Technology Conference (IMTC99), IEEE Computer Press, Piscataway, NJ, pp. 151-157
13. Fabio M. Marchese, Domenico G. Sorrenti, “Omni-directional vision with a multi-part mirror”, Workshop on EuRoboCup’2000, Amsterdam, Netherlands, 28th May – 2nd June, proceedings on CD-Rom, Springer.