We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

18

Comprehensive Omni-Directional Soccer Player Robots

Mehdi DaneshPanah[1], Amir Abdollahi[2], Hossein Ostadi[3] and Hooman Aghaebrahimi Samani[4]

[1] *Dept. of Electrical and Computer Eng., University of Connecticut, CT USA*

[2] *Dept. of Mechanical Eng., Yamaguchi University, Ube Japan*

[3] *Dept. of Mechanical Eng., Universitat Polytécnica de Catalunya, Barcelona Spain*

[4] *Dept. of Electrical and Computer Eng., National University of Singapore Singapore*

1. Introduction

RoboCup competitions were first proposed by Mackworth in 1993. The main goal of this scientific competition is to exploit, improve and integrate the methods and techniques from robotics, machine vision and artificial intelligence (AI) disciplines to create an autonomous team of soccer playing robots. At the time of preparing this chapter, RoboCup is organized in several different leagues including soccer simulator (2D and 3D), small size, middle size and legged robots, (Kitano, 1997a, Kitano et. al, 1997. Kitano, et. al, 1998). These leagues are designed to break down the problem into several venues so that the challenges can be addressed efficiently. Robots in middle-size league should operate autonomously only with local resources including local sensors, batteries and local vision. Each team can have a maximum number of four robots with a maximum footprint of 2000cm^2. They can communicate with each other through a central computer via a radio link. The rules in the competition are the same as the international soccer rules as far as they are practical for robots (Kitano, 1997b).

Recently, most conventional mobile robots have used a wheeled mechanism. Such mechanism consists of two independent driving wheels responsible for all needed robot motions (front-steering and rear-wheel driving mechanism). Motion restriction is a major problem in the use of such mechanism in mobile robots. There are also other suggested mechanisms such as universal wheel mechanisms, ball wheel, crawler and offset steered wheel mechanisms (Watanabe, 1998, West, 1992, Nakano, 1993).

Source: Robotic Soccer, Book edited by: Pedro Lima, ISBN 978-3-902613-21-9,
pp. 598, December 2007, Itech Education and Publishing, Vienna, Austria

Omni directional mobile robots have been popularly employed in several applications especially in soccer player robots considered in Robocup competitions. Such robots can reach to any position with no rotation through a straight line, so they can provide high mobility with no motion restriction. In these robots, providing high speed with an acceptable error is a very important factor in the competitive and dynamic environments (see Fig. 1). An omni directional robot can respond more quickly and it would be capable for more sophisticated behaviors such as ball passing or goal keeping.

Control and self- localization of omni directional mobile robots are important issues and different teams in the Robocup competitions have used different techniques to tackle it. Setting the PID controllers coefficients heuristically with no prior estimation based on just trials and errors is generally very time consuming. On the other hand, solving the set of coupled differential equations is very complicated and may not be practical for a real time control (Kalmar-Nagy, et. al, 2002). Some teams decoupled the mathematical model of the system while the others use fault tolerant control strategy for their systems (Jung, et. al, 2001). Real-time path generation based on the polynomial spline-interpolation with prediction of velocities of spline functions is also proposed and used (Paromatchik, et. al, 1994). A fuzzy model of the omni directional robot control was studied analytically in (Watanabe, 1998). In this chapter, we propose a calibration method for the robot controller in dynamic environments based on a simple motion to estimate initial values for the PID coefficients. For reliable and robust control, we propose a combined feedback from the odometry and vision mechanisms.

We show that the accuracy of the vision based self localization is not uniform everywhere in the field. Thus, we apply the sensitivity analysis method to evaluate the performance of the vision self-localization for feedback generation and we suggest techniques to improve the accuracy of the location feedback. By combining these strategies and utilizing the comprehensive omni directional robot (Samani, et. al, 2004), *Persia* Middle Size team won the 1st place in world Robocup technical challenge competitions in Portugal 2004 and 3rd place in Italy 2003.

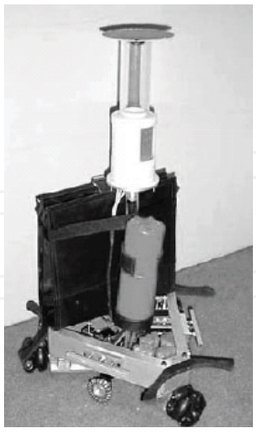


Fig. 1. A comprehensive omni directional robot having omni directional vision, motion and kicking systems

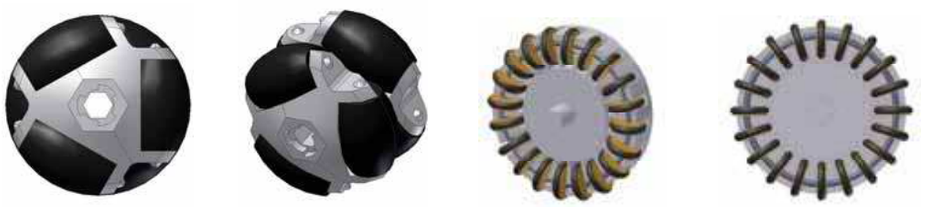
Another significant subject in the robots' soccer competition is artificial intelligence (AI), since soccer needs cooperative behavior and coordination between agents which need some form of intelligence. In this chapter, we propose a comprehensive AI architecture for this purpose in three well defined, distinct layers which provides the team with fully dynamic and flexible team work with little computational or architectural complexity cost.

This chapter is organized as follow. Sections 2, 3 and 4 describe the kinematics, dynamics and control of the robot respectively. Feedback generation and self localization using both omni-vision and odometry is presented in section 5. The omni directional kicking mechanism is described in section 6. The artificial intelligence algorithms in our robot are explained in details in section 7 and the experimental results is explained in section 8. We finally conclude this chapter in section 9.

2. Robot Kinematics

2.1 Omni Directional Wheels and Robot Chassis

Omni directional robots usually use special wheels. These wheels are known as omni directional poly roller wheel. The most common wheels consist of six spindle like rollers which can freely rotate about their longitudinal axis (Fig. 2a) (Asama, 1995, Watanabe, 1998).



(a) (b)

Fig. 2. (a) Omni directional poly-roller wheel, (b) Omni directional small-roller wheel

The surface shape and size of the poly rollers are designed such that all six rollers form a complete circle and generate a low vibration while rotating similar to a normal wheel. However, since the wheel has a low surface contact on the field compared with a normal wheel, the slippage is more severe. Due to the low vibration, this wheel is suitable for the actuating mechanism and is connected to DC motors while it is not proper for feedback generation considering its slippage. In order to avoid the slippage effects of this wheel, we design another type of omni directional wheel which consist of small cylindrical rollers mounted on the main body of the wheel in a feedback mechanism (Fig. 2b). As shown in Fig. 2b, this wheel covers a polygonal shape, so the wheel vibration is considerable. In fact, it should be mounted on the system with a flexible structure such as a flat spring (Fig. 3). Shaft encoders are mounted on these wheels.

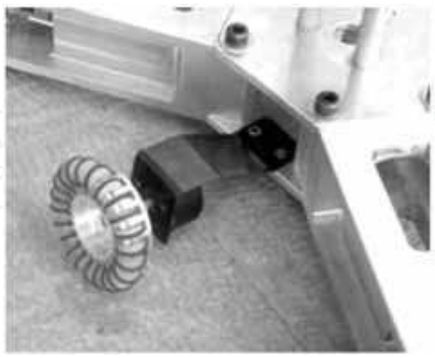


Fig. 3. Omni directional small-roller wheel connected to the body via a flat spring

A robot with three omni directional wheels can essentially follow any 2D trajectory. Our robot structure includes three big black omni-directional wheels for motion system (Fig. 5a), and three small free wheels on which shaft encoders are mounted as feedback mechanism (Fig. 4b) (Asama, et. al, 1995, Watanabe, 1998).

Fig. 4. (a) Three black omni directional poly-roller wheels act as actuators, (b) three free omni directional small-roller wheels used in a feedback mechanism

2.2 Kinematics Equations

Using omni directional wheels, the schematic view of robot kinematics can be shown as follows (Fig. 5) (Kalmar-Nagy, et. all, 2002).

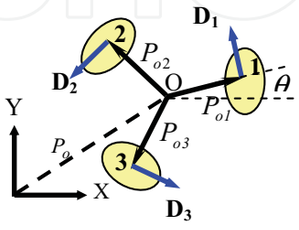


Fig. 5. Robot kinematics diagram

In the figure above, O is the robot center of mass, $\mathbf{P}_o$ is defined as the vector connecting O to the origin and D_i is the drive direction vector of each wheel. Using unitary rotation matrix, $\mathbf{R}(\theta)$ is defined as:

$$\mathbf{R}(\theta)=\begin{bmatrix}\cos(\theta) & -\sin(\theta)\\ \sin(\theta) & \cos(\theta)\end{bmatrix} \tag{1}$$

The positions vectors $\mathbf{P}_{o1},\ldots,\mathbf{P}_{o3}$ with respect to the local coordinates centered at the robot center of mass are given as:

$$\mathbf{P}_{o1}=L\begin{bmatrix}1\\0\end{bmatrix} \qquad \mathbf{P}_{o2}=\mathbf{R}(\frac{2\pi}{3})\times\mathbf{P}_{o1}=\frac{L}{2}\begin{bmatrix}-1\\\sqrt{3}\end{bmatrix} \qquad \mathbf{P}_{o3}=\mathbf{R}(\frac{4\pi}{3})\times\mathbf{P}_{o1}=-\frac{L}{2}\begin{bmatrix}1\\\sqrt{3}\end{bmatrix} \tag{2}$$

where L is the distance of wheels from the robot center of mass (O). The drive directions can be obtained by:

$$\mathbf{D}_i=\frac{1}{L}\mathbf{R}(\theta)\times\mathbf{P}_{oi} \tag{3}$$

$$\mathbf{D}_i=\begin{bmatrix}0\\1\end{bmatrix} \qquad \mathbf{D}_2=\begin{bmatrix}1\\0\end{bmatrix} \qquad \mathbf{D}_3=\frac{1}{2}\begin{bmatrix}\sqrt{3}\\-1\end{bmatrix} \tag{4}$$

Using the above notations, the wheel position and velocity vectors can be expressed with the use of rotation matrix $\mathbf{R}(\theta)$ as:

$$\mathbf{R}_i=\mathbf{P}_o+\mathbf{R}(\theta)\times\mathbf{P}_{oi} \tag{5}$$

$$\mathbf{V}_i=\dot{\mathbf{P}}_o+\dot{\mathbf{R}}(\theta)\times\mathbf{P}_{oi} \tag{6}$$

The vector $\mathbf{P}_o=[x \quad y]^T$ is the position of the center of mass with respect to Cartesian coordinates. The angular velocity of each wheel can be expressed as:

$$\phi_i=\frac{1}{r}\mathbf{V}_i^T\times\mathbf{R}(\theta)\times\mathbf{D}_i \tag{7}$$

where r is the radius of odometry wheels. Substituting for $\mathbf{V}_i$ from equation (6) yields:

$$\phi_i=\frac{1}{r}\left[\mathbf{P}_o^T\times\mathbf{R}(\theta)\times\mathbf{D}_i+\mathbf{P}_{oi}\times\dot{\mathbf{R}}(\theta)^T\times\mathbf{R}(\theta)\times\mathbf{D}_i\right] \tag{8}$$

Note that the second term in the right hand side is the tangential velocity of the wheel. This tangential velocity could be also written as:

$$L\dot{\theta}=\mathbf{P}_{oi}^T\times\dot{\mathbf{R}}(\theta)^T\times\mathbf{R}(\theta)\times\mathbf{D}_i \tag{9}$$

From the kinematics model of the robot, it is clear that the wheel velocity is a function of linear and angular velocities of robot center of mass, i.e.:

$$\begin{Bmatrix}\dot{\phi}_1\\\dot{\phi}_2\\\dot{\phi}_3\end{Bmatrix}=\frac{1}{r}\begin{bmatrix}-\sin\theta & \cos\theta & L\\ -\sin(\frac{\pi}{3}-\theta) & -\cos(\frac{\pi}{3}-\theta) & L\\ \sin(\frac{\pi}{3}+\theta) & -\cos(\frac{\pi}{3}+\theta) & L\end{bmatrix}\begin{Bmatrix}\dot{x}\\\dot{y}\\\dot{\theta}\end{Bmatrix} \tag{10}$$

Or in a equivalent vector form:

$$\dot{\boldsymbol{\varphi}} = \frac{1}{r}\mathbf{W} \times \dot{\mathbf{S}} \tag{11}$$

where L is the distance of wheels from the robot center of gravity (O) and r is the main wheel radius.

3. Robot Dynamics

Linear and angular momentum balance for the robot can be written as:

$$\sum_{i=1}^{3} f_i \mathbf{R}(\theta) \times \mathbf{D}_i = m\ddot{\mathbf{P}}_O \qquad L\sum_{i=1}^{3} f_i = J\ddot{\theta} \tag{12}$$

where $\ddot{\mathbf{P}}_o$ is the acceleration vector, f_i is the magnitude of the force produced by the ith motor, m is the mass of the robot and J is its moment of inertia about its center of gravity. Assuming no-slip condition, the force generated by a DC motor can be written as:

$$\mathbf{f} = \alpha\mathbf{U} + \beta\mathbf{V} \tag{13}$$

where, $\mathbf{V} = \{V_i(t), i = 1,2,3\}$ is the velocity of each wheel. The constants α and β are motor characteristic coefficients and can be determined either from experiments or from motor catalogue. Note that $\mathbf{U} = \{U_i(t), i = 1,2,3\}$ is the voltage applied by supplier to the DC motors. Substituting equation (13) into equation (12) yields:

$$\sum_{i=1}^{3} (\alpha U_i - \beta V_i)\mathbf{R}(\theta)\mathbf{D}_i = m\ddot{\mathbf{p}}_o \qquad L\sum_{i=1}^{3} (\alpha U_i - \beta V_i) = J\ddot{\theta} \tag{14}$$

This system of differential equations can be written in the matrix form as:

$$\begin{bmatrix} m\ddot{x} \\ m\ddot{y} \\ J\ddot{\theta} \end{bmatrix} = \alpha\mathbf{P}(\theta)\mathbf{U} - \frac{3\beta}{2}\begin{bmatrix} \dot{x} \\ \dot{y} \\ 2L^2\dot{\theta} \end{bmatrix} \tag{15}$$

$$\mathbf{P}(\theta) = \begin{pmatrix} -\sin\theta & -\sin(\frac{\pi}{3}-\theta) & \sin(\frac{\pi}{3}+\theta) \\ \cos\theta & -\cos(\frac{\pi}{3}-\theta) & -\cos(\frac{\pi}{3}+\theta) \\ L & L & L \end{pmatrix} \tag{16}$$

and

$$\mathbf{U} = \begin{bmatrix} U_1(t) & U_2(t) & U_3(t) \end{bmatrix}^T . \tag{17}$$

4. Robot Controller

PID controllers are used for controlling the robot position and orientation. In this chapter we propose a controller that is robust enough for controlling a soccer player robot (Jung, et. al,

2001). For obtaining the PID controller coefficients, one needs to first obtain the whole transfer functions of the system and then solve it accordingly. Since the equations, even if derived properly, are a set of coupled nonlinear differential equations, it is very difficult to solve them. Even though we derive and solve the equations, the results (PID coefficients) are not reliable since they depend on many other parameters such as ground surface friction factor, characteristics of batteries and so on. So the equations will be decoupled with these assumptions:

1. Omni directional mechanism is a mechanism which can reach to any position with no rotation ($\theta = 0$) through a straight line, this specification help the robot to reach the desired position in the least time compared with a 2 wheel mechanism. It is also true that every curve can be divided into some straight lines and at the end of each line the robot did not need to rotate to follow the next line.
2. Whenever it is necessary to rotate (for example when the robot kicker should be in a particular position) the robot rotates while it is moving in a straight line to reach the right position. This can be regarded as a pure rotation in addition to the first assumption.
3. The pure rotation in our robot is obtained by applying equal voltages to each motor.
4. In order to find PID coefficients for the robot position controller, moving through a straight line is very similar to move through an axis like X Direction (Y=0 in equation 15) The voltage obtained from position controller is then added to the voltage found by orientation controller.

Based on the above assumption, the robot position is not depend on θ, so for position control, we assume that θ=0. In the cases where rotation is required, the voltage obtained from orientation control for each motor equally added to the position controller output.

For PID tuning in position controller, a simple movement was considered, i.e., θ=0, Y=0 (or a constant value) in equation 15. Similarly, for orientation control, a pure rotation is considered, i.e., X=0 (or constant), Y =0 (or constant).

4.1 Position Controller Architecture

Figure 6 illustrates the overall block diagram of the system. As it is clear from Fig. 6, the omni directional robot control loop contains a PID and PD controller (with the transfer function $\boldsymbol{H}_{\mathrm{PID}}$), a plant transfer function ($\boldsymbol{H}_{\mathrm{P}}$ which is obtained from the system dynamics) and a self-localization transfer function (as a feedback function that only senses the robot's position).

A noise node, $\boldsymbol{N}$, is also included that has an additive effect on the system position input. The input of the system is considered to be a step function and the output is the robot position and orientation.

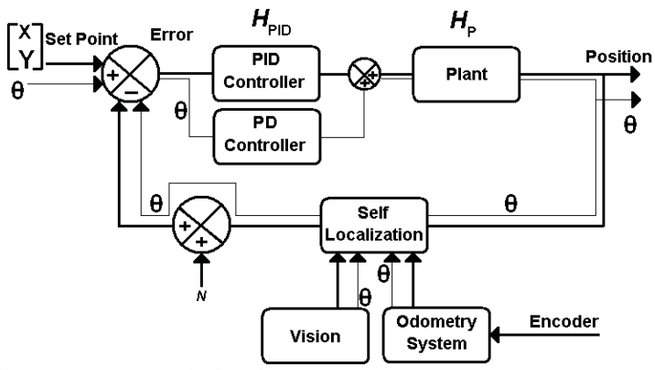


Fig. 6. Control diagram of the omni directional robot

4.1.1 H_{PID} Transfer Function

H_{PID} can be written in a general form as follows:

$$H_{PID}(s) = K_P + \frac{K_I}{s} + K_D s \tag{18}$$

Here k_P, k_I and k_D are proportional, integral and derivate gains respectively.

Experiments showed that the system overall performance is satisfactory and thus this type of controller is robust enough for controlling a soccer player robot (Kalmar-Nagy, et. al, 2002).

4.1.2 H_P Transfer Function

As it is mentioned in section 4, two simple motions were considered and solved, namely straight line motion of the robot in x direction and pure rotation about the z axis. The former means that one motor is turned off and the other two are turned on with the same angular velocity while the latter means that all three motors are turning with the same angular velocities.

We will study the orientation separately in section 4.2. The output voltage from the orientation controller (w) is then added to the voltage obtained from the position controller output (v_i). The assumption of summing up these voltages is valid while motors are operating in their linear regions. In order to apply the straight line motion, one can consider equation (15) with: $\theta = 0$ and $\dot{\varphi}_1 = \dot{y} = \dot{\theta} = \ddot{\theta} = 0$ and $\dot{\varphi}_2 = -\dot{\varphi}_3$. Equation (15) reduces then to:

$$m\ddot{x} + 3\frac{\beta \dot{x}}{2} = \sqrt{3}\alpha U_2 \tag{19}$$

Applying Laplace transfer function to equation (19) with the initial conditions: $X(0) = 0, \dot{X}(0) = 0$, one obtains:

$$H_P(s) = \frac{X(s)}{U_2(s)} = \frac{\sqrt{3}\alpha}{ms^2 + \frac{3\beta s}{2}} \tag{20}$$

It should be noted that for ideal case (in absence of noise) the complete transfer function for position control will be obtained as follows ($H_{Self\ Localization} = 1$):

$$H_{Total}(s) = \frac{H_{PID}H_P}{1 + H_{PID}H_P} = \frac{\sqrt{3}\alpha(k_D s^2 + k_P s + k_I)}{ms^3 + (\frac{3\beta}{2} + \sqrt{3}\alpha k_D)s^2 + \sqrt{3}\alpha k_p s + \sqrt{3}\alpha k_I} \tag{21}$$

Figure 7 shows the step and noise response curves with various k_p, k_I, k_D values.

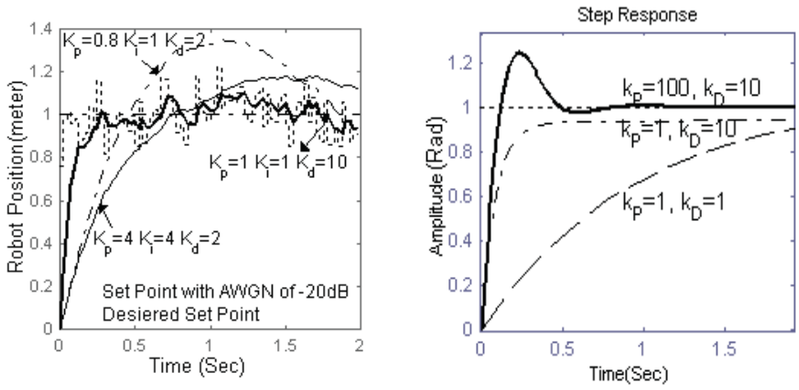


Fig. 7. (a) System step in addition noise, (b) Step response of the PD response for different values of k_p, k_I, k_D controller for orientation control

The following observations can be deducted. The dotted line in Fig. 7a shows a step function with an added noise of Gaussian distribution. In this curve the noise is applied to the system every 40 micro seconds due to the robot processing time. The mean value and standard deviation of the noise are 0. In Fig. 7a, by increasing k_P, k_I (dash-dotted line & solid line), the system response delay time will increase too. Also, there are some overshoots in these curves. However, by increasing the k_D value, this effect will reduce drastically. In order to find optimum values for the PID coefficients, different combinations of the parameters were selected and examined. Eventually, the proper PID coefficients were obtained for our system as $K_p = 1, K_I = 1, K_d = 10$. The response of the system for these values is depicted by thick solid line in Fig. 7a.

4.2 Orientation Control

It is also necessary to apply a controller for the robot rotation control. Assume that the robot only rotates about its vertical axis, i.e., Z-axis. We then derive: $\dot{\varphi}_1 = \dot{\varphi}_2 = \dot{\varphi}_3, U_1 = U_2 = U_3$.

Substituting these values into the third equation in relation (15) leads us to:

$$J\ddot{\theta} + 3\beta L^2\dot{\theta} = 3LU_1 \tag{22}$$

Applying Laplace transform to the above equation yields

$$\frac{\theta(s)}{U(s)} = \frac{3L}{Js^2 + 3\beta L^2 s} \tag{23}$$

and considering a PD controller for this case, we obtain the total transfer function for orientation control as:

$$H_{Total}(s) = \frac{3L(k_P + k_D s)}{Js^2 + (3\beta L^2 + 3Lk_D)s + 3Lk_P}. \tag{24}$$

Figure 7b shows the step response of the control system. Since the experience showed that residual error for orientation control is not of great importance in our scenario, a PD controller will result in desired system response. Therefore, there is no need to apply the Integrator controller for the orientation or robot.

The optimum parameters for this case are k_P=100, and k_D=10. The step response for these parameter values is shown by solid line in Fig. 7b. The slight overshoot is desirable since we did not consider the effects of friction that damps the response in our model.

4.3 Final Robot Controller

In order to implement the position controller, the position error vector is determined as follows:

$$\mathbf{e} = \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x' \\ y' \end{bmatrix} \tag{25}$$

while the vector [x y] and [x′ y′] are the initial and the desired position of robot in the field respectively. Thus, the position control output can be written as:

$$\mathbf{V} = K_P\mathbf{e} + K_I\int\frac{d\mathbf{e}}{dt} + K_D\frac{d\mathbf{e}}{dt} \tag{26}$$

where **V** expresses the output of the position controller for the driving units whose components on each driving wheel are extracted with:

$$\mathrm{V}_i = \mathbf{V}^T \cdot \mathbf{D}_i \tag{27}$$

In this equation, vector $\mathbf{D}_i$ is the drive direction of ith motor. The output of the position controller for each motor is v_i. Considering the PD controller for orientation control, assuming that the head angle of the robot is δ and the desired head angle is Δ, the error angle is then determined as:

$$e_\Delta = \Delta - \delta \tag{28}$$

The orientation controller output is thus given by:

$$w = K_P e_\Delta + K_D\frac{de_\Delta}{dt} \tag{29}$$

The voltage from the orientation controller output is then added to the voltage obtained from the position control output. The final applicable voltages on the motors are then computed as:

$$u_i = v_i + w \tag{30}$$

This voltage is applied to each motor to reach the desired point. Since the system sensitive parts such as electronic control board, computer, batteries, etc., may be damaged by rapid rotation of the robot, we need to apply an upper and a lower threshold for the orientation controller output. Practically we set the threshold to be ± 10v.

The PID and PD coefficients were obtained from the two previous cases, used as a first estimation. This is due to the robot working conditions such as friction, gear boxes clearances and tolerances, motors mechanical time constant and etc that are not considered in modelling. The proper coefficients were then tuned experimentally in each competition. The results showed that for real cases, the maximum changes in the calculated values were around 10%. Therefore, such simplification is a good approximation for control model. Therefore, such simplification is a good approximation for control model.

5. Feedback Generation and Self-Localization

The position control method described in the former sections, calls for some form of position feedback in order to work properly. The performance of this feedback lies in its reliability, accuracy and real time computability. There have been plenty of algorithms and methods proposed by different researchers in the literature (Borenstein 1997, Olson 2001, et. al, Talhuri, et. al, 1993). Among them self- localization by visual information and odometry approach are dominant due to their special characteristics which will be discussed in the following paragraphs.

In this work, a compound novel method was developed and optimized for RoboCup MSL (Middle Size League) in which both visual and odometry information are used to ameliorate a real time, accurate and reliable method. Although optimized for soccer player robots, the self-localization method proposed here has enough modularity and flexibility to be applicable in most robotics applications involving self-localization.

Each of these complementary methods (vision/odometry self-localization) operates autonomously and has its own advantage and drawbacks in providing position feedback for robot control. For example, odometry method is known to have memory-based operation, accumulative error, low jitter, simplicity of implementation, cheap hardware, etc.

On the other hand, vision-based self- localization algorithms often have memoryless implementations (although there exists memory-based ones), no error accumulation, high jitter, relatively high computation complexity and expensive hardware. That is why amalgamating these methods can bring us to a global novel algorithm with good performance in vast and diverse conditions.

In the first subsection, the vision self-localization is explored in details and its different aspects are inspected. The second subsection describes a sensitivity analysis on the proposed method that demonstrates its performance in different locations in the field. Then using the evidence of sensitivity, slight modifications are presented for further robustness of the

overall algorithm. Next, odometry self- localization is proposed in brief and at last, the fusing algorithm of both visual and odometry outputs is described together with an Additive White Gaussian Noise (AWGN) model for the jitter.

5.1 Vision-Based Self-Localization Using Omni Vision Sensor

5.1.1 Hyperboloidal Omni-vision Sensor

The hyperboloidal mirror is a specific solution among the family of polynomial mirror shapes. These mirrors do not provide a central perspective projection, except for the hyperbolic one, but still guarantee a linear mapping between the angle of elevation φ and the radial distance from the center of the image plane ρ (Fig. 8) (Gaechter, S., 2001).

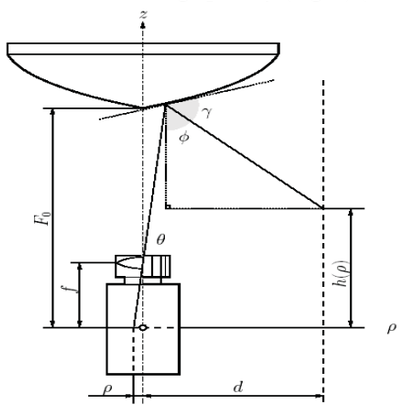
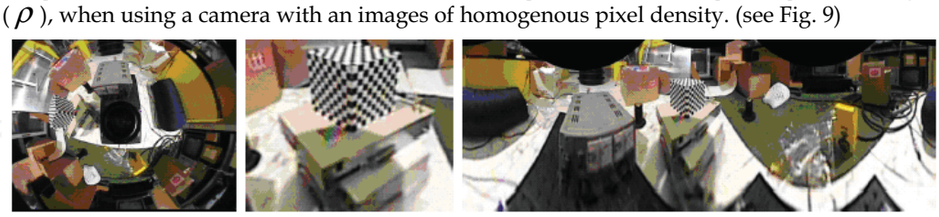


Fig. 8. Schematic diagram of the omni vision sensor

Another required specification is to guarantee a uniform resolution for the panoramic image. The resolution in the omni directional image increases with growing eccentricity, (ρ), when using a camera with an images of homogenous pixel density. (see Fig. 9)



(a) (b) (c)

Fig. 9. (a) Input omni directional image, (b) perspective and (c) panorama image (Yachida, M., 1998)

Searching through the literature, we found that the following hyperbolic curve is commonly used for omni directional vision mirror (Ishiguro H., 1998):

$$\frac{x^2}{233.3} - \frac{y^2}{1135.7} = -1 \tag{31}$$

However, this equation is suitable for the mirror with large size and wide view. For our soccer player robot, we need an image with a diameter of 3.5m on the field. To design a compact mirror with wide view, the above curve scaled down by a factor of 2.5 to yields:

$$\frac{x^2}{233.3} - \frac{y^2}{1135.7} = -6.25 \tag{32}$$

Next, a special three stages process was considered for the mirror manufacturing:

1) Curve fabrication , 2) Polishing, 3) Coating

In the first stage the curve was fabricated on steel 2045 with CNC machining. Then, the work-piece was polished by a special process and finally Ni-P electroless plating was employed.

5.1.2 Vision-Based Self-Localization

Vision module was designed with several goals in mind; preparing spatial information of ball, opponents, team mates and self-localization raw data, exploited by self-localization module, are the key prospects. Our robot platforms were equipped with omni directional cameras (Talluri, et. al, 1993), with which the projection of the whole field area was available to the camera with a hyper-parabolic mirror described in section 5.1.1. Since the omni directional mirror introduces a map with very high non-linearity between pixel separation in the scene and the real physical distance (of such pixels) in the field itself, it is not reliable enough to develop algorithms which use distances as their input data with such mirrors.

Instead angles are preserved completely in a linear manner if the center of mirror and camera are aligned perfectly. Therefore, the algorithms with angles as their input data are more reliable and can perform more efficiently.

Our approach in vision-based self-localization is based on arcs. In basic geometry, there is a fact that having an angle of observation ω to a fixed and spatially known object in a 2D plane, can provide us with possible loci of the observation point. Actually, the points are located on the circumference of two circles (C_1, C_2). This simple idea is illustrated graphically in Fig. 10.

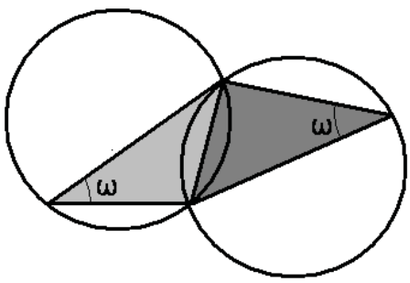


Fig. 10. Angle of observation ω and the two related arcs

The proposed algorithm here employs three different observation angles to constrain the unique position of observer (robot) in the field (assuming the ideal case with no visual noise). A good set of observation angles should have the following properties:

- Availability from different locations in the field.
- Extractable from visual data with low computational effort.
- The arcs resulting from these angles must be independent which means they should leave no location ambiguity at any point in the filed.
- The greater the angles magnitude, results in the lower sensitivity to visual noise (that will be discussed later).

Since goals are fixed landmarks and at least one of them has reasonable observation angle within the whole field area, their use for self-localization is popular in Robocup Middle Size League (Stroupe, et. al, 2002). An insightful examination through different combinations of possible observation angles for this purpose revealed that the following three angles are suitable regarding the above characteristics:

1. The observation angle from the robot itself to the nearest goal (α_{Goal}).
2. Angle between the center of the farthest goal and left side of the nearest one (β_{Goal}).
3. Angle between the center of the farthest goal and right side of the nearest one (γ_{Goal}).

These angles are depicted for an arbitrary location of the robot in the field in Fig. 11. Assume that the intersection points between Arc(j), and Arc(k) to be defined as:

$$P_i^{j,k} \qquad \begin{matrix} j,k \in \{1,1',2,2',3,3'\}; j \neq k \\ i \in \{1,2\} \end{matrix} \tag{33}$$

where the superscripts denote intersecting arcs and a subscript denotes the index of intersection. Note that the robot position is always on one point located on Arc (1).

First, a list of intersection points pairs are made using equation 33. In order to find the exact location of the robot, the Euclidian distances of different pairs of intersections are computed and the one which has zero norm is selected as the answer. In other words, there is only one point that is located on the intersection of three arcs and this point is the real position of the robot in ideal case, i.e., with no noise.

$$\underset{i,j,s,t}{Min}\left\{\left\| P_s^{1,i} - P_t^{1,j} \right\|\right\} \qquad \begin{matrix} i,j = 2,2',3,3' \qquad i \neq j \\ s,t = 1,2,3,... \end{matrix} \tag{34}$$

Considering imperfectness in visual information extraction, the intersection of Arc(1) with other two arcs may not coincide. In such a case, the set that yields the minimum distance introduces the possible position of the robot. The final position is simply computed by averaging over the neighboring intersection points that satisfy the above criteria (Fig. 11).

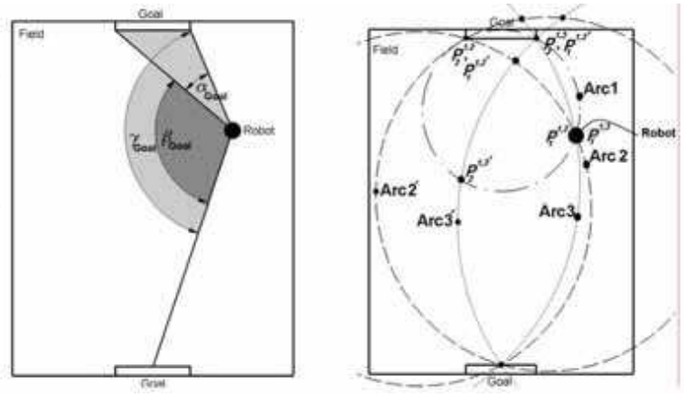


Fig. 11. (a) Angles observed by the robot, (b) The arcs and possible intersection positions

5.2 Sensitivity Analysis

The performance of vision-based self-localization method, developed in this work, relies on accurate visual information obtained from the vision module by means of image processing algorithms and techniques. Since goals are of two distinct colors in the play field (Yellow and Blue), the pixels representing them are distinguished by their position in RGB color space, and then their position and angle of observation are extracted with special region growing algorithms.

As mentioned before, although the angles are preserved linearly in the omni directional filed of view projected by the hyperbolic mirror, there is always the possibility that some error would exist in the detection procedure.

The sensitivity analysis of vision-based self-localization method reveals the regions in which the method is most sensitive to visual noise. The sensitivity of some performance characteristic y regarding parameter xi, is defined as the measure of its change Δy , resulting from a change Δx_i in the parameter xi. Suppose:

$$y = y(x_1, x_2, ..., x_n) \tag{35}$$

The variation of y is defined as:

$$dy = y\sum_{i=1}^{n}\left[\frac{x_i}{y}\frac{\partial y}{\partial x_i}\right]\frac{dx_i}{x_i} = y\sum_{i=1}^{n} S_{x_i}^{y}\frac{dx_i}{x_i} \tag{36}$$

where $S_{x_i}^{y}$ denotes the sensitivity of y with respect to parameter xi , and is computed as:

$$S_{x_i}^{y} = \frac{x_i}{y}\frac{\partial y}{\partial x_i} \tag{37}$$

Applying the above analysis on the proposed self- localization method in section 5.1 shows that in certain areas near the corner posts, the sensitivity increases and the accuracy of the method degrades drastically. Therefore, the proposed algorithm may be prone to severe errors in these regions (see Fig. 12). Since there are flags in the corner posts (that are of good

visibility and detectibly in that region by vision module), these landmarks are proper candidates for self-localization in these regions.

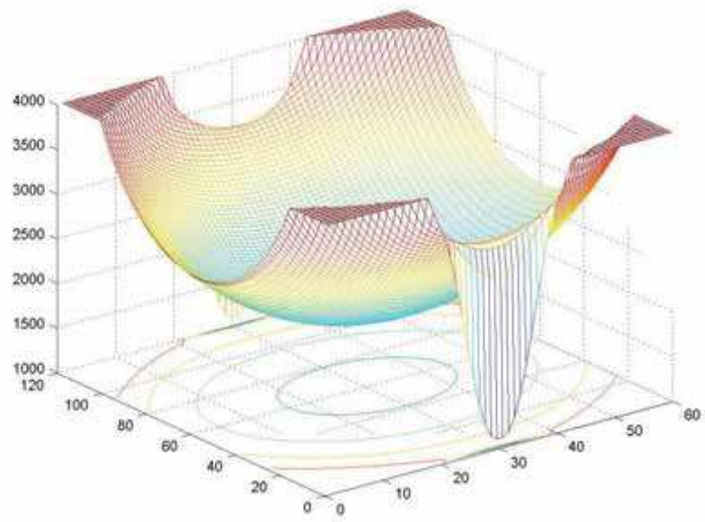


Fig. 12. Sensitivity of vision-based self-localization method at different location in the field

5.3. Localization Using Flags

For achieving better performance in the regions in which the sensitivity of the vision-based self-localization method is high, flags are used instead of goals to determine the position of robot. The procedure can be summarized as follow.

- By using visual data of goals and previous location of robot from its memory, the location of robot is roughly determined as Front-Left, Front-Right, Back-Left, Back-Right, where Front and Back show opponent and own side fields respectively.
- The nearest flag is then detected and the distance of robot to the flag base is approximated by a non-linear map constructed experimentally.
- Since the exact position of flag is known and the relative position of robot from flag is also available (R), then calculating the final robot position is a trivial task, i.e.:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} X_{FLAG} \\ Y_{FLAG} \end{bmatrix} + \begin{bmatrix} R\cos(\varphi) \\ R\sin(\varphi) \end{bmatrix} \tag{38}$$

Since the method of localization changes in these regions, and in order to avoid bouncing and confusion between these two methods, a hystersis strip (the grey area between two arcs near the flag in Fig. 13) is defined. Therefore, once the method changes to usage of flag (robot crosses the inner ring), it sticks to the new algorithm till the robots moves out of the outer ring in the hystersis strip.

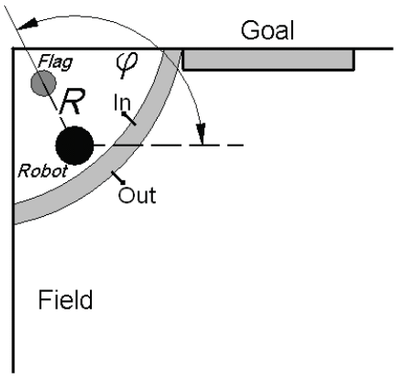


Fig. 13. The schematic view of the robot and flag near the corners (the grey strip is where the hystersis occurred)

5.4. Self-Localization Using Odometry

As it can be seen in Fig. 4b, three free rotating omni directional wheels are placed 60 degrees apart from the main driving wheels. These wheels are only passive, attached to three independent shaft encoders and have the role of odometry wheels. In reference (Samani, et. al, 2004) the direct method for position extraction using the data of shaft encoders is described. We only state the results here:

$$x = r\int \frac{1}{3\sin(\frac{\pi}{3})}\begin{bmatrix}(\cos(\frac{\pi}{3}+\theta)-\cos(\frac{\pi}{3}-\theta))\dot{\varphi}_1+\\(-\cos(\theta)-\cos(\frac{\pi}{3}+\theta))\dot{\varphi}_2+(\cos(\theta)+\cos(\frac{\pi}{3}-\theta))\dot{\varphi}_3\end{bmatrix}dt\ ,$$

$$y = r\int \frac{1}{3\sin(\frac{\pi}{3})}\begin{bmatrix}(\sin(\frac{\pi}{3}-\theta)-\sin(\frac{\pi}{3}+\theta))\dot{\varphi}_1+\\(-\sin(\theta)-\sin(\frac{\pi}{3}+\theta))\dot{\varphi}_2+(\sin(\theta)-\sin(\frac{\pi}{3}-\theta))\dot{\varphi}_3\end{bmatrix}dt\ , \quad (39)$$

$$\theta = r\int \frac{\dot{\varphi}_1+\dot{\varphi}_2+\dot{\varphi}_3}{3\mathrm{L}}\mathrm{dt}\ ,$$

where $[x \quad y \quad \theta]^T$ is vector containing the position and orientation of the robot. Further simplification of the third equation in (39) results in:

$$\theta = \frac{r}{3\mathrm{L}}\left[\int\dot{\varphi}_1\,\mathrm{dt}+\int\dot{\varphi}_2\,\mathrm{dt}+\int\dot{\varphi}_3\,\mathrm{dt}\right]=\frac{r}{3\mathrm{L}}(\varphi_1+\varphi_2+\varphi_3)\,. \quad (40)$$

5.5 Final Position Estimator

In order to obtain the final position estimation for the robot, both visual and odometry outputs must be fused in an appropriate fashion that would take advantage of each method to make flaws from the other one ineffective. For example, due to the inherent nature of vision-based self-localization, there is undesired jitter at its output, but, in return, odometry self-localization has smooth changes that can be used as a low-pass filter for vision-based self-localization results. Having this in mind, we came up with the following procedure for estimating the final position:

Step 1: Vision-based self-localization estimates the current position of the robot based on visual information from the current frame.

Step 2: Odometry utilizes the last computed position and determines the new position using the equation set (39).

Step 3: The position of robot is then computed as a weighted average of odometry and vision-based self-localization as:

$$\overline{P} = 0.9P_{Odometry} + 0.1P_{Vision} \tag{41}$$

Using these coefficients results in smoothing the variation (due to jitter in vision-based self localization) of final position estimation. The coefficients in equation (41) were obtained by carrying on tests on the robot position.

Step 4: The initial position for odometry in step 2 is then set to the computed robot position in the step 3 and the calculation continues for the next frame.

Since the outputs of both odometry and vision-based self- localization are prone to errors, and due to inherent random nature of these errors, a 2D Additive White Gaussian Noise (AWGN) is added to the output of a perfect self-localization block in the feedback path as shown in Fig. 6. The noise can be formulated as:

$$n_g(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} e^{-(\frac{x^2}{2\sigma_x^2}+\frac{y^2}{2\sigma_y^2})} \tag{42}$$

where σ_x and σ_y are noise deviation in X and Y directions, respectively. These values are then added to the position obtained from the self-localization module, (x0,y0), to obtain the probabilistic location of the robot, i.e. (x,y) as:

$$L(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} e^{-(\frac{(x-x_0)^2}{2\sigma_x^2}+\frac{(y-y_0)^2}{2\sigma_y^2})} . \tag{43}$$

6. Omni Directional Kicking System

For a soccer player robot, usually one direction is used for directing the ball to the goal or other destinations. Therefore, the ball handler and kicking mechanism is added in this direction which help the robot to get a suitable form for directing the ball.

Although the conventional mechanism can work for any soccer robot, it has some limitations which require additional movements for a particular behaviour. In other words, each robot has a specific head for kicking the ball and must adjust it to the proper direction during the game. These adjustments in the single head robot increase its rotation significantly and reduce its maneuverability. In order to minimize such rotations, we use two extra kicking mechanisms to form an omni directional kicking system. The position of these kicking systems is shown in Fig. 14a. As it can be seen from the figure, each kicker is assembled between two omni directional wheels and forms a system with three heads in 0°, 120°and 240° angles. Three types of soccer player robot in the form of 10 teams, which participated in Robocup, were examined in order to assess the rotation rate of each type.

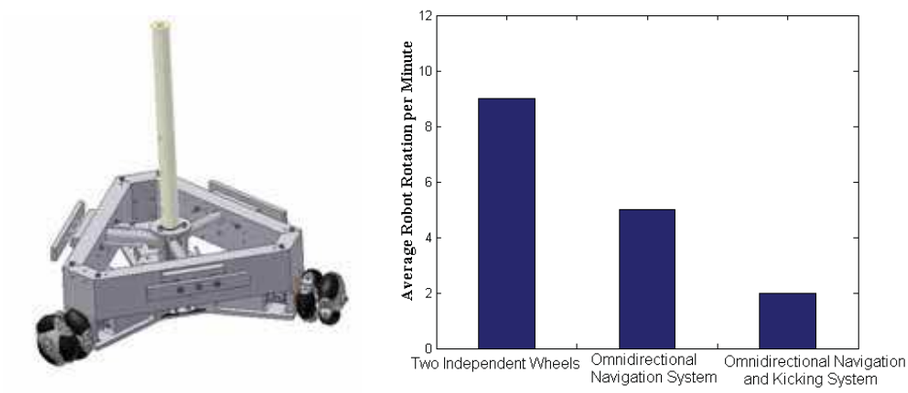


Fig. 14 (a) Omni directional kicking system of the robot, (b) Average rotation rate for three types of soccer player robots

In this assessment, the number of complete rotations for each robot in one minute was measured and the total average of them was then calculated. Fig. 14b shows these numbers for three types of robots, i.e.:

1- Two independent driving wheels mechanism

2- Omni directional navigation system (one head)

3- Omni directional navigation and kicking system (three heads)

There is a significant decrease in the rotation rate between the first type and the third type of these robots shown in Fig. 14b. The number of complete rotation per minute for omni directional single head and three heads robot are 5 and 2 respectively. It shows that using the omni-kick system is very useful and can reduce the rotation rate in robots with the same navigation system about 60%.

Essentially, fewer necessary rotations simplifies the algorithm needed for following a trajectory, cooperative behaviors and increases the speed and flexibility for directing the ball to the desired destination.

7. Artificial Intelligence

Artificial Intelligence has been of researchers' interest for many years since the need for autonomous systems emerged. Several approaches to this specific issue have been studied widely in different areas ranging from pure cognitive science outlook to pure engineering perspective; each having its own methodology. Although these approaches have different standpoints, it has shown that amalgamating the findings of each, results in very interesting achievements. In this section we will follow the engineering approach in general, for that its applications in practical engineering problems are more dominating.

From the early days of utilizing machines, systems able to make some simple decisions when needed, became necessary in some applications. Engineers overcame this type of criteria in their design with simple logics which were implemented mechanically or electronically.

But in the past few decades the need for complicated tasks by robots has called for sophisticated methods in artificial intelligence. For example, in situations where robots had to be utilized on another celestial body in the space, the remote control of such robots were not practical, so the robot needed to have its own intelligence in order to accomplish its mission successfully. As another example, assume a robot which has to perform like humans in a particular task (i.e. ping-pong player robot); as humans decide intelligently for such behaviors, such robots has to be able to decide intelligently as well.

From the discussion above, it's pretty much clear that in the field of concentration of this chapter (soccer player robots), artificial intelligence is of great importance in both agent level and team level behavior. In the following subsections the principle ideas have been developed and explored in detail. This section is organized as following, firstly the local and global world models will be discussed, and then the architecture of the proposed AI hierarchy and its modules will be explored in details. Next a discussion on communication protocols, used as a medium for transmitting data between agents, will be considered. And finally a brief discussion on trajectory computation will be carried out.

7.1 World Model Construction

Although each agent tries to extract the real world map from the fusion of visual and non-visual data as accurate as possible, but "noisy data" and "non-global optimized" algorithms reduce the reliability of processed data. First, let us clarify what is meant by "noise" and "optimized algorithms" with a few examples in a mobile robot. The flaws of color space modeling result in wrong color classification, which in turn makes the object detection algorithms prone to errors. As a result, a robot may not see an opponent because of its poor color table for the opponent's tag color, or it may see an orange T-shirt in the spectators' area as a ball! These wrong outputs are referred as "noise". By this classification, the CCD noise pattern or faulty shaft encoder samples due to motors noise are excluded. There is a trade off between speed and reliability in most algorithms. Middle size league in Robocup has a well-defined environment (e.g. distinct colors, defined sizes and etc), which can be very helpful in simplifying the design of a fast algorithm.

Since a predefined environment is assumed, any changes in this environment can more or less result in wrong movements. For example, for self-localization the width of goals are assumed to be fully viewable in close situations; when an object taller than a robot (like a human) cuts or occludes a part of a goal in the image, the output of the vision self-

localization module will not be reliable anymore. Detection of such a situation can be a very cumbersome task and making the algorithm very complicated and therefore slow.

From the discussion above, it is apparent that multi agent data fusion algorithms are necessary for constructing a better approximation of the real world. In addition to the software which resides on each robot, stand alone software for network communication, world model construction, cooperative behavior management and game monitoring need also to be developed. The world model module receives different data sets from every agent. Each data set contains different environmental information like self, ball and opponents' positions. Each data carries a 'confidence' factor; a larger confidence factor means a more reliable piece of information. The most recent data sets are then chosen for data fusion, in which the following rules and facts are applied:

- Closer object are of assumed to be of more accuracy.
- Objects further than a specific distance could be said to be totally inaccurate. (This distance is heuristically obtained)
- An object in the field cannot move faster than an extreme value.

With respect to the above facts, the module filters unwanted duplicates of objects, (e.g. many opponents close to each other seen by different agents), calculates the best approximation for ball and opponents' positions with first order Kalman filtering, gives every object a confidence factor, applies a low pass filter on data and finally constructs a complete world model. This new world model contains information about the objects which may not have been seen by each agent correctly and also enhances approximations of all environmental information. The constructed world model is then sent back to all agents so they will have a better view of the world around them!

7.2 Artificial Intelligence Architecture

The architecture proposed and used for the purpose of a soccer player team has a 3 layer hierarchical framework, namely AI Core, Role Engine and Behavior Engine (Murphy, R., 2000). These layers are completely independent with well defined interfaces to avoid complexities in further developments (i.e. adding new behaviors in order to accomplish a certain role more effective must not influence the AI Core layer). This particular 3 layer architecture enables us to decentralize the whole AI routines among a ground machine and robots in the field accordingly, which in practice means that AI Core, resides in and runs on a ground machine outside the field (along with the monitoring module), while Role and Behavior Engines run as local processes in individual robots' processing units. The following diagram shows the building blocks and their interaction in the proposed architecture.

The interaction between the modules on different machines is provided by a communication protocol which bundles commands and parameters generating command packets and interprets the incoming packets for other modules. In the following, each layer, its interface and parameters will be discussed in details. Finally the communication protocol designed for performing the interactions will be described briefly.

7.2.1 AI Core

As it can be seen from Fig. 15, AI Core is the topmost layer in our proposed architecture. This core has been implemented using case based reasoning method in which all the possible cases had been anticipated during the design process, these cases will be discussed shortly. Although no adaptation or learning takes place in this layer, clever design and useful parameter definition can give enough flexibility to this layer, while avoiding convergence problem in adaptive designs.

The objective of this layer can be simply stated as following:

- Collecting data from World Model Constructor (WMC) module.
- Collecting parameter values set by human supervisor before the game.
- Extraction of GameState from the above parameters and setting it to a member of the following set: GameState=*{HighDefence ,MedDefence, LowDefence, LowAttack, MedAttack, HighAttack}*
- Assigning each necessary role for each GameState to the robot which can execute the role better.
- Sending the role and its parameters along with world model information to selected robots. Now let's take a closer look at the algorithms performing the above steps. The GameState is uniquely derived from the following table.

Ball Region / Ball Ownership	Region 1	Region 2	Region 3	Region 4
Own Team	MedDefence	LowDefence	MedAttack	HighAttack
Opponent Team	HighDefence	LowDefence	LowAttack	MedAttack

Table 1. Derivation of GameState from world model information

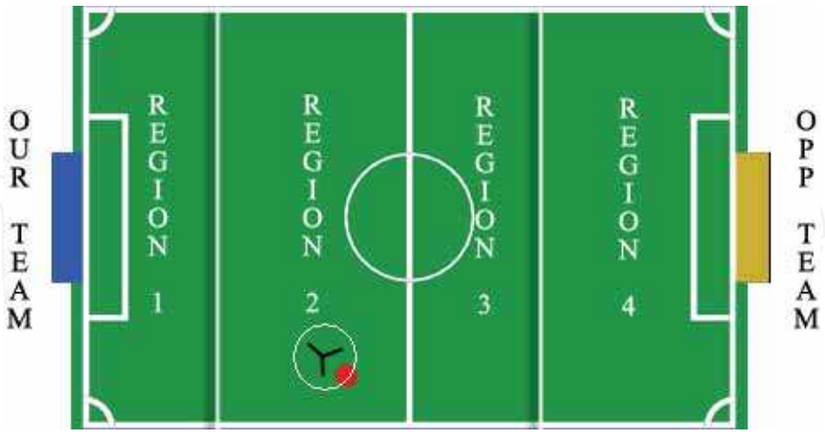


Fig. 15. AI core visual module and the defined regions

In order to avoid undesirable sudden changes of *GameState*, which can result in sudden changes in roles assigned to robots, *GameState* determination was implemented memory based. This means that the current *GameState* is selected as the most dominant *GameState* in a pool of *GameStates* from the last few seconds. In AI Core, roles are changed in a manner in which a continuity exists between current and previous assigned roles, therefore, robots never experiment sudden changes in roles (for example the role defense never changes to attack in the next cycle). Four roles are assigned for each *GameState* manually based on the evaluation of the opponent team strategy. For example, *MedAttack* might be associated with the following roles: *Goal Keeper*, *CenterDefence*, *Supporter* and *Attacker*. In a more conservative situation *Supporter* might be substituted with a *CenterFieldSupporter*.

7.2.2 Role Engine

After assignment of each necessary role to the robot which can perform it better by AI Core, the role, along with its optional parameters are sent to the agent through the communication layer. The task of the Role Engine is to:

- Initiate the new assigned role (i.e. by proper termination of the previous role)
- Initiate a thread to feed the Behaviour Layer with necessary behaviours in order to accomplish the assigned role.
- Determining the essential behaviours according to the parameters received from the AI Core, and feeding them sequentially to the Behaviour Layer.
- Watching the results of each behaviour returned from the Behaviour Layer (i.e. success or fail) and deciding the action to take according to results returned.

7.2.3 Behaviour Layer

Behaviours are the building blocks of the robot's performance which includes simple actions like rotating (*behRotate*), or catching the ball (*behCatchBall*) and etc. The Behaviour Layer is the lowest layer in our architecture.

This layer receives a sequence of behaviours along with some parameters from the upper layer (Role Engine) and executes the essential subroutines in order to accomplish a certain behaviour. These subroutines use world model information and trajectory data in order to perform necessary movements.

7.3 Cooperative Behaviour

Here we give an example of how all these layers and modules cooperate in a synchronous fashion to finally show an intelligent set of behaviours. Assume in a certain point of time during the match, AI Core evaluates the robots' positions, the ball location and the team possessing it from the global world model reported by WMC, and then concludes the state of the play to be *MedAttack* from table 1. This defines the strategy of the game which consequently requires some certain roles to be present in the field like Attacker, Supporter and Defender. The AI Core then assigns each role to the robot which is most qualified to perform that role. In order to avoid unwanted bouncing between roles of a single robot, there exists a First In Last Out (FILO) queue for each robot in the AI Core. This queue acts as

an intermediate between AI Core and each single robot, for that it fetches the roles assigned by the AI Core to each single robot and then from the previous values of its memory, determines if the new role has enough credibility to be assigned to the robot or if the current role is still the winner of the queue (having the majority of positions in the queue). This way the roles are somehow low pass filtered before assignment.

Now suppose the role Attacker is finally decided to be assigned to a robot by the FILO queue in the AI core; this role along with some parameters (i.e. shooting distance) is passed to the robot's Role Engine. The Role Engine evaluates the state of the robot by checking its global world model and determines if the robot possesses the ball or not. In case of no possession, it sends the Behaviour Layer a command to start *behCatchBall* behaviour. This routine gets the rudder of the robot, fetches the best path to the destinations from trajectory module and then issues appropriate commands for the control module to move the robot to its destination (i.e. behind ball in this example). If the behaviour was successfully finished, which means complete possession of the ball, it returns a success code to the role which has called *behCatchBall*. Having the possession of the ball, now, the role Attacker calls another behaviour *behDribbleBall* which is again a subroutine in the Behaviour Layer. The intent of this behaviour could be moving the ball toward the opponent's goal while avoiding obstacles (like opponent's robots). When this subroutine got the robot to the desired shooting distance (which was previously passed to the Role Engine by the AI Core), it returns the success code to Attacker role. This role then sends a request to the Behavior Layer to perform the *behShootToGoal* behaviour, and this process repeats as many times as needed. The key point here is that the Role Engine evaluates the state of the robot and selects which behaviour is needed; the rest is left to the Behaviour Layer to watch over proper execution of the behaviour.

8. Experimental Results

In order to evaluate the performance of the position controller suggested in section 4.1 and self-localization error, four experiments were designed.

First, PID position control was applied. The robot tracked on a straight line of 1m length near the center of the field with no rotation. Second, the PD orientation control was employed with just rotation about the Z-axis of the robot. Third, the robot was programmed to follow a sinusoidal curve ("A" in Fig. 16) with the wave-length of 5m and amplitude of 3.5m near the center of the field. Finally, the robot pursued two sinusoidal curves similar to curve A, but far from the center of the field ("B" and "C" in Fig. 16).

In the first experiment, the PID constants were set as those calculated in section 1.C. The maximum deviation from the straight-line tracking and the final position error were measured to be 8 cm and 4 cm respectively (right line in Fig. 16).

In the second experiment, again the PD controller parameters are set to the calculated values for orientation control (section 4.2). The maximum error from the set point angle was 0.03π radians.

These two experiments show that the final error for both tracking and pure rotating are in an acceptable level and the PID and PD controller parameters are selected properly.

In the third experiment, the robot had to track the sinusoidal curve ("A" in Fig. 16) while rotating about its Z-axis.

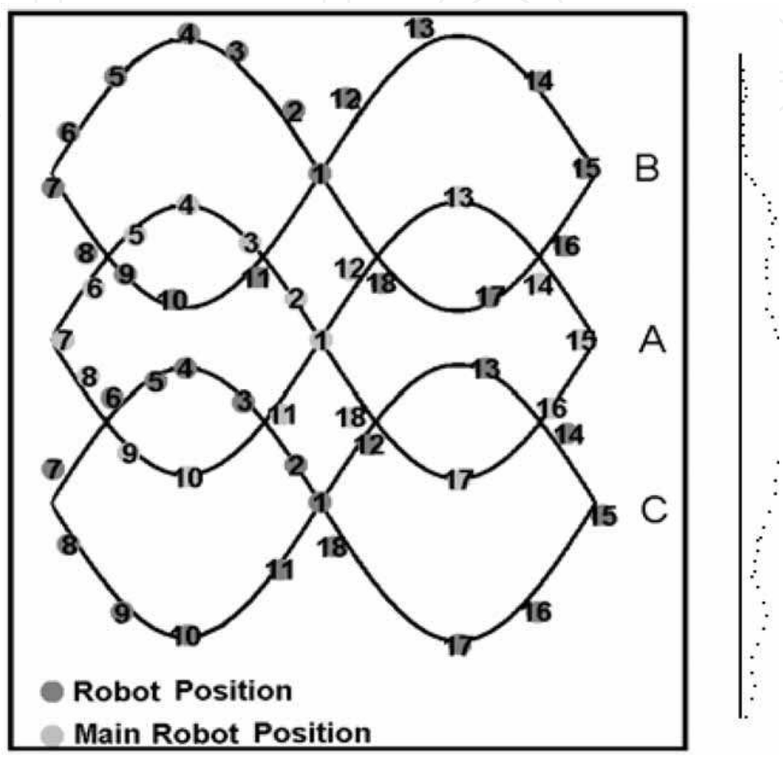


Fig. 16. A, B and C depict the robot trajectories. The numbers shows each robot position on each curve. The right picture illustrates the straight line followed by the robot

The measured errors were between 10 to 12 cm and occurred at points 4, 10, 13 and 17 in curve "A" in Fig. 16. The maximum deviation was measured to be around 12 cm that occurred in point 4.

In the last experiment, the curves were located near the edges of the field ("B, C" in Fig. 16) The maximum deviation between the real and desirable path was measured to be around 23 cm that is less than 7% for this case study.

Figure 16 shows the position of the three robots on the field at different times while Fig. 17. is a picture of the robot tracking on the curves "A", "B", and "C" in the competition field.

Fig. 17. Notations "A", "B", "C" are the robots that followed the corresponding "A", "B", and "C" curves in Fig. 21

9. Conclusion

1. In this study, we propose PID and PD controllers for position and orientation controls respectively. Then, the controller parameters were estimated using a simplified model by taking into account the effect of noise. By using these parameters in the real robot, it was shown that the strategy was appropriate for an omni directional robot.
2. Self-localization method utilized in this study used a combination of the odometry system (using a shaft encoder) and vision-based localization. Using the geometrical properties of circles, we managed to calculate the exact position of the robot in the field. Next, a sensitivity analysis was carried out to determine the inaccurate points in the field. For those regions, we used the flags as our landmarks in the corners to overcome such difficulty. An algorithm, employing the above techniques, was developed and tested on the real field.

   The test results showed that the asymmetric errors for omni directional mobile robots were reduced drastically on those areas. The improvement of performance was more than 80% in position and orientation in comparison with the time when only the original localization was used.
3. For the first time, omni directional navigation system, omni-vision system and omni-kick mechanism have been combined to create a comprehensive omni directional robot. This causes great reduction in robot rotation during soccer play.
4. The idea of separating odometry sensors from the driving wheels was successfully implemented. Three separate omni directional wheels coupled with shaft encoders placed 60 apart of the main driving wheels. The result was reducing errors such as slippage in the time of acceleration.

10. References

Asama, H.; Sato, M.; Bogoni, L.; Kaetsu, H.; Matsumoto, A. & Endo, I. (1995). Development of an omni directional mobile robot with 3 DOF decoupling drive mechanism, *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 1925-1930, Japan, May 1995, IEEE, Nagoya

Borenstein, J.; Everett, H. R.; Feng, L. & Wehe, D. (1997). Mobile robot positioning: Sensors and techniques, *J. Robot. Syst.*, Vol. 14, (April 1997) (231–249), 0741-2223

Carlisle, B. (1983). An Omni-Directional Mobile Robot, In: *Development in Robotics*, (79-87), IFS Publication Ltd., 0444865942, England

Chenavier, F.; Crowley, J. (1992). Position estimation for a mobile robot using vision and odometry, *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 2588-2593, France, May 1992, IEEE, Nice

Gaechter, S. (2001). Mirror Design for an Omni directional Camera with a Uniform Cylindrical Projection when Using the SVAVISCA Sensor, In: *The Technical Description of Deliverable D2*, (45-52), Czech Technical University, Czech Republic

Ishiguro H. (1998). Development of Low-cost Compact omni directional sensors and their applications, *Proceedings of International Conference on Information Systems, Analysis and Synthesis*, pp. 433-439, USA, July 1998, Florida

Jung, M.; Kim, J. (2001). Fault tolerant control strategy for OmniKity-III, *Proceedings of IEEE International Conference on Robotics Automation*, pp. 3370-3375, Korea, May 2001, IEEE, Seoul

Kalmar-Nagy, T.; Ganguly, P. & D,Andea, R. (2002). Real-time trajectory generation for omni directional vehicles, *Proceedings of the American control conference*, pp. 285-291, USA, May 2002, IEEE, AK

Kitano H. (1997a). RoboCup: The robot world cup initiative, *Proceedings First International Conference on Autonomous Agents*, pp. 340-347, USA, Feb. 1997, Springer-Verlag, CA

Kitano, H. (1997b*). RoboCup 97: Robot Soccer World Cup I*, Springer-Verlag, 3540644733, New York

Kitano, H.; Asada, M.; Kuniyoshi, Y.; Noda, I.; Osawa, E. & Matsubara, H. (1997). RoboCup: A challenge problem for AI, *AI Magazine*, Vol. 18, No. 1, (April 1997) (73-85), 0738-4602

Kitano, H. (1998). Research program of RoboCup, *Applied Artificial Intelligence*, Vol. 2, No. 2-3, (March 1998) (117-125), 0883-9514

Kortenkamp, D.; Bonasso, P. & Murphy, R. (1998). *Artificial Intelligence and Mobile Robotics,* AAAI Press / MIT Press, 0262611376, USA

Mackworth, A.K. (1993). On seeing robots, In: *Computer Visions: Systems, Theory and Applications*, Basu, A.; Li, X., (1-13), World Scientific, 9810213921, Singapore

Menegatti, E.; Pagello, E. & Wright, M. (2002). Using Omni directional Vision within the Spatial Semantic Hierarchy, *Proceedings of IEEE International Conference on Robotics & Automation*, pp. 908-914, USA, May 2002, IEEE, Washington DC

Muir, P.F.; Neuman, C.P. (1987). Kinematic modeling for feedback control of an omni directional wheeled mobile robots, *Proceedings of IEEE International Conference on Robotics & Automation*, pp. 1772-1786, USA, March 1987, IEEE, NC

Murphy, R. R. (2000). *An Introduction to AI Robotics,* MIT Press, 0262133830, USA

Killough, S.M. (1994). A new family of omni directional and holonomic wheeled platforms for mobile robots, *IEEE Transaction on Robotics and Automation,* Vol. 10, No. 4, (August 1994) (480-493), 0882-4967

Nakano, E.; & Koyachi, N. (1993). An Advanced Mechanism of the Omnidirectional Vehicle (ODV) and Its Application to the Working Wheel Chair for the Disabled, *Proceedings of 83th International Conference on Advanced Robotics*, pp. 277-284, USA, May 1993, IEEE, Atlanta

Olson, C. F. (2000). Probabilistic Self-localization for Mobile Robots, *IEEE Transactions on Robotics and Automation,* Vol. 16, No. 1, (Feb. 2000) (55-66), 0882-4967

Paromatchik, I.; Rembold, U. (1994). A practical Approach to motion Generation and Control of Omni directional Mobile Robot*, Proceedings of IEEE International Conference on Robotics & Automation,* pp. 2790-2795, USA, May 1994, IEEE, CA

Samani, H.; Abdollahi,A.; Ostadi, H.; Ziaie Rad, S. (2004). Design and development of a comprehensive omni directional soccer player robot, *International Journal of Advanced Robotic Systems*, Vol. 1, No. 3, (Sept. 2004) (191-200), 1729-8806

Stroupe, A.; Sikorski, K. & Balch, T. (2002). Constraint-based landmark localization, *Proceedings of RoboCup 2002: Robot Soccer World Cup IV,* pp. 447-453, Japan, 2002, Springer-Verlag, Fukuoka

Talluri, R. & Aggarwal, J. K. (1993). Position estimation techniques for anautonomous mobile robot—A review, In: *Handbook of Pattern Recognition and Computer Vision*, Chen, C. H.; Pau, L. F. & Wang, P. S. P., (769–801), World Scientific, 9812561053, Singapore

Watanabe, K. (1998). Control of an omni directional mobile robot, *Second International Conference on Knowledge Base Intelligent Electronic Systems*, pp. 51-60, Australia, April 1998, Adelaide

West, M. & Asada, H. (1992). Design a Holonomic Omni Directional Vehicle, *Proceedings of IEEE International Conference on Robotics & Automation,* pp. 97-103, France, May 1992, IEEE, Nice

Weigel, T.; Gutmann, J.-S.; Dietl, M.; Kleiner, A.; Nebel, B. (2002). CS Freiburg: coordinating robots for successful soccer playing**,** *IEEE Transactions on Robotics and Automation*, Vol. 18, (Oct. 2002) (685-699), 0882-4967

Weiss, G. (2000). *Multiagent systems: A Modern Approach to Distributed Artificial Intelligence,* MIT Press, 0262232030, USA

Yachida, M. (1998). Omni directional Sensing and Combined Multiple Sensing, *Proceedings of IEEE and ATR Workshop on Computer Vision for Virtual Reality Based Human Communications*, pp. 20-27, India, Jan. 1998, IEEE, Bombay

Yagi, Y. (1999). Omni directional Sensing and its Applications, *IEICE TRANS ,INF. & SYST,* Vol. E82-D, No.3, (March 1999) (568-579), 0916-8532

Robotic Soccer

Edited by Pedro Lima

ISBN 978-3-902613-21-9

Hard cover, 598 pages

Publisher I-Tech Education and Publishing

Published online 01, December, 2007

Published in print edition December, 2007

Many papers in the book concern advanced research on (multi-)robot subsystems, naturally motivated by the challenges posed by robot soccer, but certainly applicable to other domains: reasoning, multi-criteria decision-making, behavior and team coordination, cooperative perception, localization, mobility systems (namely omni-directional wheeled motion, as well as quadruped and biped locomotion, all strongly developed within RoboCup), and even a couple of papers on a topic apparently solved before Soccer Robotics - color segmentation - but for which several new algorithms were introduced since the mid-nineties by researchers on the field, to solve dynamic illumination and fast color segmentation problems, among others. This book is certainly a small sample of the research activity on Soccer Robotics going on around the globe as you read it, but it surely covers a good deal of what has been done in the field recently, and as such it works as a valuable source for researchers interested in the involved subjects, whether they are currently "soccer roboticists" or not.

InTech Europe

University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China

Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen