An Efficacious Method to Assemble a Modern Multimodal Robotic Team: Dilemmas, Challenges, Possibilities and Solutions

Amir Abdollahi
Dept. of Applied Mathematics III
Universitat Polytécnica de Catalunya, Spain

Hooman Aghaebrahimi Samani
Dept. of Electrical and Computer Eng.
National University of Singapore, Singapore

1 Introduction

A modern multiagent robotic platform consists of a cooperative team of humans which develop a collaborative team of robots. The multimodal nature of both the system and the team causes a complex problem which needs to be solved for optimum performance. Both the management and the technical aspect of a modern robotic team are explored in this Chapter in the platform of the RoboCup Competition.
RoboCup is an example of such an environment where researchers from different disciplines join to develop a robotic team for completion as an evaluation challenge (Robocup, 2011). 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 disciplines to create an autonomous team of soccer playing robots (Kitano, 1997a; Kitano, 1997b; Kitano et al., 1997). Such experiment includes several challenges, from inviting an expert of specific field to the team to choosing bolts and nuts for each part of the robots. Usually each challenge has several possible solutions and choosing the best one is often challenging. We have participated in several worldwide RoboCup competitions (Abdollahi, Samani et al., 2002, 2003 & 2004) and share our experience as an extensive instruction for setting up a modern robotic team including management and technical issues.

As per management aspect, an efficient method for setting up a team consisting of researchers and engineers is presented. Robotics is multidisciplinary research field and team members from different disciplines should work closely with other team members efficiently. Any miscommunication and misunderstanding between team members will cause wasting time and energy while deadline is always fixed. Such issue is even more problematic when team members are from different disciplines with different backgrounds. In Section 2 of this Chapter, methods and approaches for efficient collaboration and team working are discussed while the maximum technical expertise of each member is employed.

As per technical aspect, the scientific approach for design and undertaking the most efficient solution for each technical challenge is presented. Omni-directional mechanisms, optimized odometry system, cooperative behavior and world modeling are elaborated with comparing all the possible solutions and reasoning for picking the appropriate one. Despite recent advances, effective control and self-localization of omni-directional mobile robots remain as important and challenging issues. A simplified model of the system is derived for fast tuning

of the control system parameters. In particular, strategies for fast tuning of PID/PD coefficients for position and orientation control are devised. A vision-based self-localization and the conventional odometry systems are fused for robust self-localization. The methods have been tested in the RoboCup competition field using three middle size omni-directional robots. The experimental results are shown to demonstrate the effectiveness of the proposed system. All of these technical issues are discussed in Section 3. The Chapter is concluded in Section 4.

2 Team Management

According to the Contingency theories of team management, the optimal method to organize a corporation is dependent upon the internal and external situation. Hollenbeck et al. extended the structural contingency theory and considered issues of external fit simultaneously with its examination of internal fit at the team level by applying that to several teams working on interdependent team tasks (Hollenbeck et al., 2002). They indicated that divisional structures demand high levels of cognitive ability on the part of team members. Woodward et al. argued that technologies directly determine differences in organizational attributes that are typically associated with or best fit the use of different technologies as span of control, centralization of authority, and the formalization of rules (Woodward, 1978; Woodward et al., 1980).

We believe that Contingency theories can be applied in the robotic team internally and externally. External fit deals with matching team structure and environment while internal fit deals with matching structures and people. Success in a robotics team depends upon a number of variables, including the leadership style, qualities of the followers and aspects of the situation.

Members of a robotic team are basically from three disciplines; Mechanics, Electronics and Computer science. These three disciplines form two main groups of software and hardware. Such a hybrid structure requires optimum performance between team members.

As first example, choosing a suitable processor requires opinion from computational members for the best software performance, electrical members for energy consumption and relevant electronics and mechanical members for design considerations. However the role of computational members is more critical in this aspect.

As second example, a control board needs to be designed by electrical members according to the requirements of the control by computational members which is also needs to be considered by mechanical members for robot design regarding system dynamics and robot implementation.

As third example, mechanical team designs a setup for the vision system which needs to be appropriate according to computational members for software issues and electrical members for hardware issues.

Above three examples illustrate three samples of basic collaborative tasks in the process of development of a robot in a team with different levels of importance in each one.

2.1 Management Hierarchy

Base on our experience in RoboCup competitions, we suggest a management hierarchy for a robotics team. On top of this hierarchy a team supervisor is required. Team supervisor usually can be one of the academic staff of a university or research center. Team supervisor requires general management skills without essentially deep knowledge about robotics.

In the second layer of management, two managers are in direct contact with the team supervisor: technical and administrative managers. The suitable technical manager can be a person with overall knowledge about robotics from both software and hardware points of view. However the technical manager doesn't need extensive knowledge about three mechanical, electrical and computer disciplines. Administrative manager handles overall administrative task of the team. He/she doesn’t require any specific robotics expertise and in small teams can be any of the team members which are willing to handle paper works, financial matters, public communication and other relevant issues.

Third layer of management are three group leaders in mechanical, electrical and computer disciplines. Usually these three members can be chosen as most experienced person in each field with extensive knowledge about each of three disciplines. These three group leaders should have deep knowledge and familiarity with technical aspects of their group. The abovementioned proposed management hierarchy is illustrated in Figure 1.

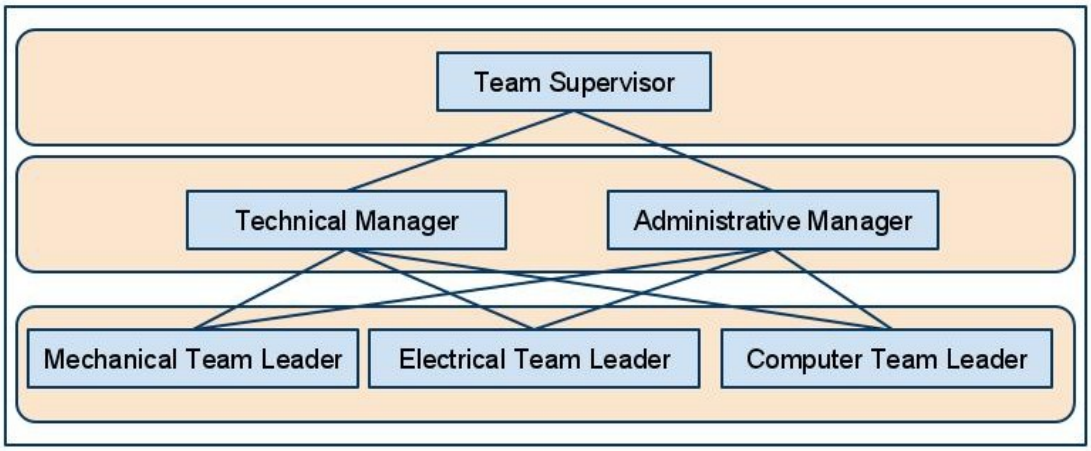


Figure 1: The proposed management hierarchy of a modern robotic team.

One of the most common mistakes in robotic teams is holding public meetings which requires the participation of the all the members. Ideally team supervisor only needs to be in contact with technical and administrative managers which can even be separately. Most the meetings need to be only between the technical manager and three group leaders. Naturally group members don't need many formal meetings as they are working closely together. Administrative issues should be reflected to the administrative manager by technical managers who him/her self received those issues from group leaders. Such a work flow optimizes the administrative work load of the team which generally is much more than expected before forming a robotic team.

However we believe that Contingency theory should be considered while applying the above proposed management hierarchy. The optimal management structure is dependent upon internal and external situation of the robotic team. The structure of a robotic team and particularly RoboCup is naturally dynamic. Often one of the members may handles few roles in the team, deadlines change priorities in the team, administrative issues affect technical issues indirectly, the structure needs to be changed in different time and most of the members are students who cannot dedicate full time on the project considering their other obligations. These are only few common examples of dynamic challenges in the structure of the team. Hence we suggest that proposed management hierarchy to be considered as the main structure of the team which changes according to the internal and external situation dynamically for the best efficiency.

3 Technical Concepts

Among many suggested motion mechanisms such as universal wheel, ball wheel, crawler and offset steered wheel, and omni-directional wheel (Watanabe, 1998; Kitano et al., 1998; West et al., 1992; Nakano & Koyachi, 1993), omni-directional wheels can provide high mobility with no motion restriction. In practice, providing high speed with an acceptable error is very important factor for success in a competitive and dynamic environment such as RoboCup competitions. Figure 2 presents an omni-directional robot which can reach to any posi-

tion with no rotation through a straight line. For this purpose, *fast* yet *robust* and reliable self-localization and control approaches must be adopted. Additionally, in the context of novice operation (such as in the student's competition contest), or time-pressured situations, the system must be *simple* to develop and tune.

Despite many works related to self-localization of robots (Borenstein et al., 1997; Talluri & Aggarwal, 1993; Olson, 2000; Stroupe, 2002; Skrzypczynski, 2005; Simmons & Koenig, 1995; Fox et al., 1998; Thrun et al., 1998), the problem is still open. Common methods of dead-reckoning (Borenstein et al., 1997) are prone to errors that are accumulated over time. Therefore, it is necessary to combine other methods such as triangulation landmarks or map matching, in order to probabilistically update robot's localization. The problem is usually formulated with a likelihood function over all possible positions of the robot and a measure is used to find a probabilistic match between local and global maps (Simmons & Koenig, 1995; Fox et al., 1998; Thrun et al., 1998). However, these approaches are usually complicated and time-consuming. Reliability and robustness of many of these approaches are also questionable for robotic soccer competitions (Olson, 2000; Martinelli, 2007). We have proposed a simple, efficient, and reliable hybrid self-localization method using a fused system of odometry and vision feedbacks (Samani, Abdollahi et al., 2004). Each of these feedbacks has its own advantages and limitations. Odometry provides ease and low cost of implementation and computation, but is limited by the slippage effect and accumulation of odometry errors. Vision-based self-localization ensures flow of rich information unaffected by the slippage effect, yet limited by the camera occlusion and camera calibration errors (of extrinsic and intrinsic parameters). Also, image processing techniques might be time-consuming. The hybrid odometry system is proposed to compensate for disadvantages of both methods (Samani, Abdollahi et al., 2004). In particular, localization errors, e.g., the slippage effects of driving wheels, will not dominate the self-localization results. Additional contribution of our work includes the sensitivity analysis of the performance of a vision self-localization and feedback system. The objective is to obtain sensitivity of the localization method to visual noise. The results show that using one method for all points in the field was not perfect. Hence utilizing other landmarks in the field was proposed.

From control perspective, advanced control techniques have been proposed for omni-directional robots, with many being computationally inefficient, or impractical, or difficult to tune, and/or implement (Watanabe, 1998; Kalmar-Nagy, 2002; Jung, 2001; Paromatchik, 1994). Among many control techniques, Proportional-Integral-Derivative (PID) control remains outstanding due to its simplicity, robustness, effectiveness, a wide range of applicability, and near-optimal performance (Cominos et al., 2002). Therefore, PID strategy was adopted for the position control of the robots. We propose a simple strategy for fast yet effective tuning of a PID control. The orientation control is achieved using PD control law. It is a time consuming process to set the PID controllers coefficients manually with no prior estimation and based on just trials and errors. On the other hand, solving a set of coupled differential equations is very complicated and may not be practical for a real time control. Some teams decoupled the mathematical model of the system while the others used fault tolerant control strategy for their systems (Jung, 2001). Real-time path generation based on the polynomial spline-interpolation with prediction of velocities of spline functions was also proposed and used (Paromatchik, 1994). A fuzzy model of the omni-directional robot control was studied analytically (Watanabe, 1998). However, these approaches had problems such as lengthy effort for control tuning, complicated mathematical models for a real-time trajectory generation, and/or use of a single feedback system for control structure. Also, some of these models offered only theoretical but impractical solutions.

Another significant subject in Robocup 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.

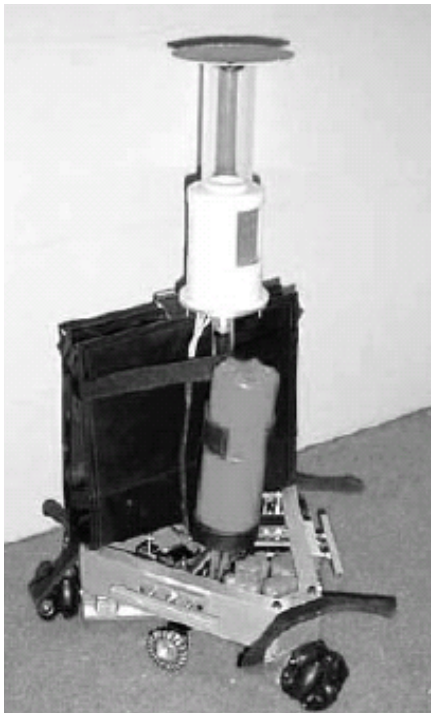


Figure 2: Omni-directional soccer player robot.

3.1 Omni-directional Wheels and Robot Chassis

Omni-directional robots usually use special wheels known as omni-directional poly roller wheels. The most common wheels consist of six spindle like rollers which can freely rotate about their longitudinal axis (Figure 3a) (Watanabe, 1998; Asama, 1995).

(a) (b)

Figure 3: (a) Omni-directional poly-roller wheel, (b) Omni-directional small-roller wheel.

The 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 designed another type of omni-directional wheel which consists of small cylindrical rollers mounted on the main body of the wheel in a mechanism for purposes of the feedback control or odometry sensor. As shown in Figure 3b, 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 (Figure 4). Shaft encoders are mounted on these wheels.

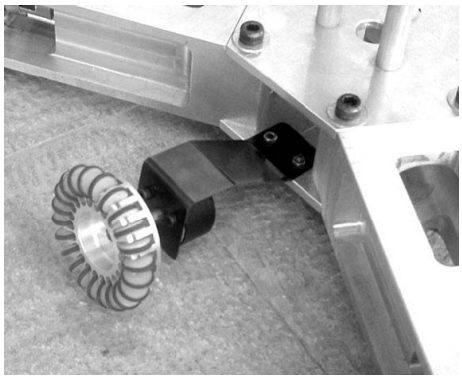
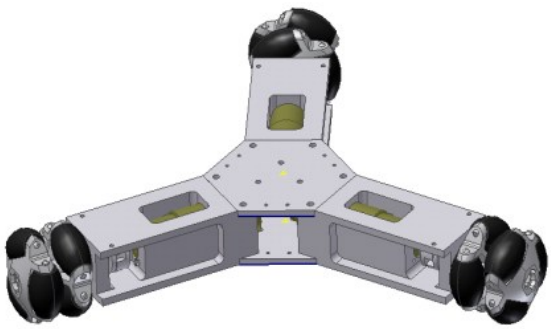


Figure 4: Omni-directional small-roller wheel connected to the body via a flat spring.

A robot with three omni-directional wheels can essentially follow any two dimensional trajectory. Our robot structure includes three big black omni-directional wheels for motion system (Figure 5a), and three small free wheels on which shaft encoders are mounted as feedback mechanism (Figure 5b).



(a)

(b)

Figure 5: (a) Three black omni-directional poly-roller wheels act as actuators, (b) Three free omni-directional small-roller wheels used in a feedback mechanism.

3.2 Robot Kinematics

The schematic view of the robot kinematics with omni-directional wheels is shown in Figure 6.

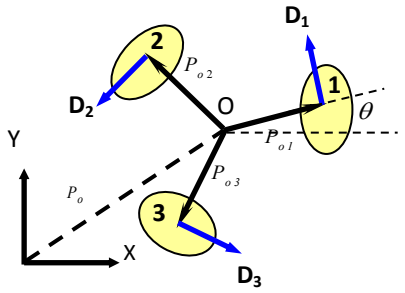


Figure 6: Robot kinematic diagram with local and global coordinate frames.

From the kinematics model of the robot (Kalmar-Nagy, 2002), one can derive the vector of the coordinates of the wheels centers with respect to a local coordinate frame ($\boldsymbol{P}_w$) and drive directions as:

$$\boldsymbol{P}_w = \begin{bmatrix} P_{w1}^T \\ P_{w2}^T \\ P_{w3}^T \end{bmatrix} = L \begin{bmatrix} 1 & 0 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \tag{1}$$

$$\begin{bmatrix} \boldsymbol{D}_{w1}^T \\ \boldsymbol{D}_{w2}^T \\ \boldsymbol{D}_{w3}^T \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{\sqrt{3}}{2} & -\frac{1}{2} \\ \frac{\sqrt{3}}{2} & -\frac{1}{2} \end{bmatrix} \tag{2}$$

where L is the distance of wheels center from the robot center of gravity (O), and vector $\boldsymbol{D}_{wi}$ is the drive direction of the i-th motor. The vector of linear velocities of the wheels ($V_i(t)$, i=1, 2, 3) can be written as:

$$\boldsymbol{V} = \dot{\boldsymbol{P}}_w + \dot{\boldsymbol{R}}(\theta)\boldsymbol{P}_o \tag{3}$$

where $\boldsymbol{R}(\theta)$ is the rotation matrix. Then it can be readily shown that the wheels angular velocity vector, $[\dot{\varphi}_1, \dot{\varphi}_2, \dot{\varphi}_3]^T$, can be written as a function of linear and angular velocities of the robot (i.e., $[\dot{x}, \dot{y}, \dot{\theta}]^T$):

$$\begin{bmatrix} \dot{\varphi}_1 \\ \dot{\varphi}_2 \\ \dot{\varphi}_3 \end{bmatrix} = \frac{1}{r} \begin{bmatrix} -\sin\theta & \cos\theta & L \\ -\sin(\frac{\pi}{3} - \theta) & -\cos(\frac{\pi}{3} - \theta) & L \\ \sin(\frac{\pi}{3} + \theta) & -\cos(\frac{\pi}{3} + \theta) & L \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} \tag{4}$$

where r is the major radius of wheels. Linear and angular momentum equations for the robot can be formulated

as:

$$\sum_{i=1}^{3} F_i \boldsymbol{R}(\theta)\boldsymbol{D}_{wi} = m\ddot{\boldsymbol{p}}_o \;,\; L\sum_{i=1}^{3} F_i = J\ddot{\theta} \tag{5}$$

where $\ddot{\boldsymbol{p}}_o = [\ddot{x}, \ddot{y}]^T$ is the linear acceleration vector of the center of mass with respect to Cartesian coordinate frame, F_i is the magnitude of the force produced by the i-th 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:

$$\boldsymbol{F} = \alpha \boldsymbol{U} - \beta \boldsymbol{V} \,, \tag{6}$$

where $\boldsymbol{U} = \{U_i(t),\ i = 1, 2, 3\}$ is the voltage applied by a supplier to the DC motors. The constants α and β are motor characteristic coefficients and can be determined either from experiments or from the motors catalogue. Substituting (6) into (5) yields:

$$\sum_{i=1}^{3} (\alpha U_i - \beta V_i)\boldsymbol{R}(\theta)\boldsymbol{D}_{wi} = m\ddot{\boldsymbol{p}}_o \;, \tag{7a}$$

$$L\sum_{i=1}^{3} (\alpha U_i - \beta V_i) = J\ddot{\theta} \,, \tag{7b}$$

$$\begin{bmatrix} m\ddot{x} \\ m\ddot{y} \\ J\ddot{\theta} \end{bmatrix} = \alpha \boldsymbol{P}(\theta)\boldsymbol{U} - \frac{3\beta}{2}\begin{bmatrix} \dot{x} \\ \dot{y} \\ 2L^2\dot{\theta} \end{bmatrix} \,, \tag{8}$$

$$\boldsymbol{P}(\theta) = \begin{pmatrix} -\sin\theta & -\sin(\frac{\pi}{3}-\theta) & \sin(\frac{\pi}{3}+\theta) \\ \cos\theta & -\cos(\frac{\pi}{3}-\theta) & -\cos(\frac{\pi}{3}+\theta) \\ L & L & L \end{pmatrix} \tag{9}$$

3.3 Robot Controller

In this work, PID and PD controllers were used for controlling the robot position and orientation respectively. The experiments showed that such system was robust enough for controlling a soccer player robot (Jung, 2001). For obtaining the PID controller gains, one needs to obtain first the whole transfer functions of the system and then use it for initial tuning. Determining overall equations governing the system behavior is not straightforward. Since the equations are a set of coupled nonlinear differential equations, it is very difficult to solve them in a time-efficient fashion. Even if one manages to solve the equations, the resultant PID gains will not be reliable because they will depend on many other parameters such as ground surface friction factor, characteristics of batteries and so on. For many robotic competitions, an efficient and fast tuning method is desired. Therefore, the equations need to be decoupled with the use of the following assumptions:

(1) Omni-directional mechanism is a mechanism which can reach to any position with no rotation (i.e., without loss of generality, one can assume θ = 0) through a straight line. This prescription would help the robot to reach the desired position in the shorter time than that with a two wheel mechanism. It can be also assumed that any curve could be approximated by dividing it into straight line segments in a way that at the end of each

segment, the robot would not need to rotate to follow the next segment.

(2) Whenever it is necessary to rotate (e.g., when the kicker robot needs to be in a particular position), the robot rotates while it is moving in a straight line to reach the target position. This can be regarded as a pure rotation in addition to the first assumption. The pure rotation can be obtained by applying equal voltages to each motor.

(3) In order to find the PID coefficients for the robot position controller, moving through a straight line is very similar to moving through an axis like X-axis (i.e., $y = 0$ in (8)). The voltage obtained from position controller is then added to the voltage found by orientation controller.

Based on the above assumptions, the robot position does not depend on θ. Therefore, for position control, one would assume that θ = 0. In the cases where rotation is required, the voltage obtained from orientation control for each motor is 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 (8). Similarly, for orientation control, a pure rotation is considered, i.e., $x = 0$(or constant), and y = 0 (or constant).

3.3.1Position Control Structure

The overall block diagram of the system is shown in Figure 7. The omni-directional robot control loop contains a PID controller (with the transfer function H_{PID}) and a PD control law, a plant transfer function (H_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, 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.

Two simple motions were considered and solved, namely straight-line motion of the robot, e.g., along 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 but opposite angular velocity while the latter means that all three motors are turning with the same angular velocities.

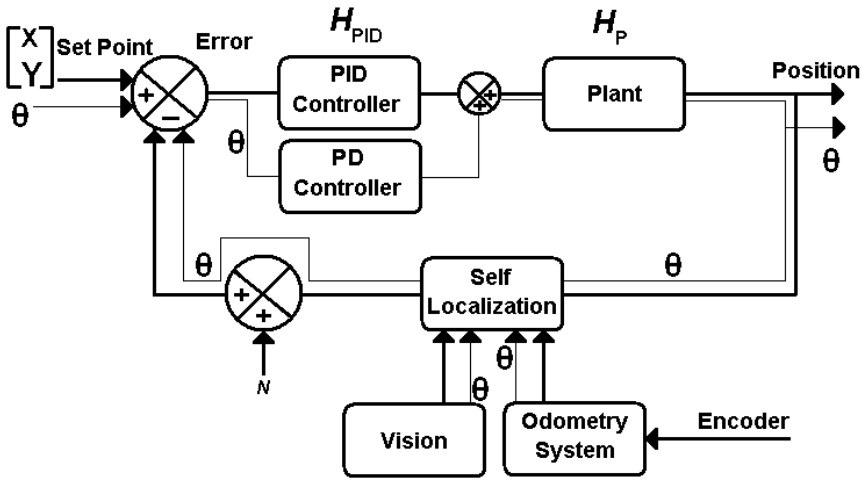


Figure 7: Control diagram of the omni-directional robot.

The orientation will be studied separately in the next Section. 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 (8) with $\theta = 0$, $\varphi = \dot{y} = \dot{\theta} = \ddot{\theta} = 0$, $\dot{\varphi}_2 = -\dot{\varphi}_3$. Equation (8) then reduces to:

$$m\ddot{x} + 3\frac{\beta\dot{x}}{2} = \sqrt{3}\alpha U_2 \tag{10}$$

Applying Laplace transform to (10) with the initial conditions: $X(0) = 0$, $\dot{X}(0)=0$, one obtains:

$$H_P(s) = \frac{X(s)}{U_2(s)} = \frac{\sqrt{3}\alpha}{ms^2 + \frac{3\beta s}{2}} \tag{11}$$

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

$$H_{Total}(s) = \frac{H_{\mathrm{PID}} H_{\mathrm{P}}}{1 + H_{\mathrm{PID}} H_{\mathrm{P}}} = \frac{\sqrt{3}\alpha (K_D s^2 + K_P s + K_I)}{m s^3 + (\frac{3\beta}{2} + \sqrt{3}\alpha K_D) s^2 + \sqrt{3}\alpha K_p s + \sqrt{3}\alpha K_I} \tag{12}$$

Here K_p, K_I and K_D are proportional, integral and derivate gains, respectively. Figure 8 shows the step and noise response curves with various K_p, K_I and K_D values. The following observations can be deducted. The dotted line in Figure 8 shows a step function with an additive white (zero-mean) Gaussian noise (AWGN). In this curve, the noise was applied to the system every 40 microseconds due to the robot processing time. As observed from Figure 8, by increasing K_p and K_I (dash-dotted line and solid line), the system settling time would increase. Also, there are some overshoots in these curves. However, by increasing the K_D value, this effect reduces drastically. In order to find the optimum values for the PID gains, different combinations of the parameters were selected and examined. Eventually, the proper PID gains were obtained for the proposed system as K_p=1, K_I=1 and K_D=10. The response of the system for these values is depicted by thick solid line in Figure 8.

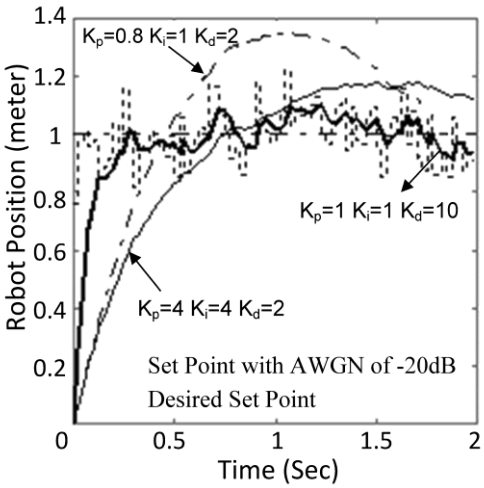


Figure 8: System step response of position control with different values of PID gains.

3.3.2 Orientation Control

Suppose that the robot only rotates about its vertical axis, i.e., Z-axis. Thus: $\dot{\varphi}_1 = \dot{\varphi}_2 = \dot{\varphi}_3$, $U_1 = U_2 = U_3$. Substituting these values into the third equation in (8) leads to:

$$J\ddot{\theta} + 3\alpha\beta L^2\dot{\theta} = 3LU_3 \tag{13}$$

Applying Laplace transform to the above equation yields

$$\theta(s)/U_3(s) = 3L/(Js^2 + 3\alpha\beta L^2 s) \tag{14}$$

Considering a PD controller for this case, the total transfer function for orientation control is given as:

$$H_{Total}(s) = \frac{3L(K_P + K_D s)}{Js^2 + (3\alpha\beta L^2 + 3LK_D)s + 3LK_P} \tag{15}$$

Figure 9 shows the step response of the control system. Experiments showed that the level of noise (measured by noise/signal ratio) in orientation controller was considerably less than that in the position controller (almost 3 times). Therefore, the noise was ignored in tuning PD control gains. Since the experience showed that residual error for orientation control is not of great importance in the given scenario (i.e., robotic soccer competitions), a PD controller will result in desired system response. Therefore, there was no need to apply PID controller for the orientation control. The optimum parameters for PD gains were obtained as K_p = 100, and K_D = 10. The step response for these parameters values is shown by a solid line in Figure 9. The slight overshoot is desirable since the effect of friction (damping the response in our model) was ignored.

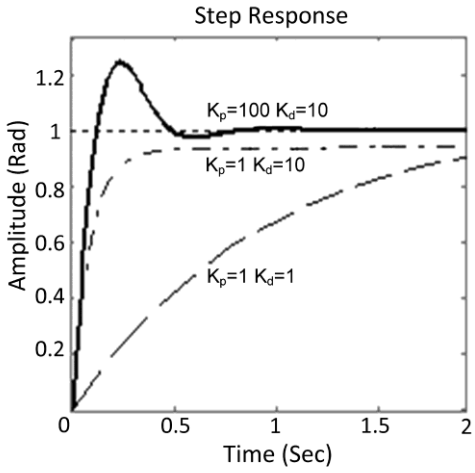


Figure 9: Step response of orientation control for different values of PD gains.

3.3.3 Overall Robot Controller

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

$$\boldsymbol{e} = \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x' \\ y' \end{bmatrix} \tag{16}$$

while the vectors $[x\ y]^T$ and $[x'\ y']^T$ are the desired and the actual position of robot in the field, respectively.

Thus, the position control output can be written as:

$$\boldsymbol{V}_m = K_P \boldsymbol{e} + K_I \int \boldsymbol{e} dt + K_D \frac{d\boldsymbol{e}}{dt} \tag{17}$$

where $\boldsymbol{V}_m$ expresses the output vector of the position controller for the driving units whose components on each driving wheel (V_{mi}) are extracted from:

$$V_{mi} = \boldsymbol{V}_m^{\,T} \cdot \boldsymbol{D}_{wi} \tag{18}$$

For orientation control (using PD law), the orientation error can be calculated using the desired and current head angles of the robot, namely Δ and δ, respectively, as follows:

$$e_{\Delta} = \Delta - \delta \tag{19}$$

The orientation controller output will be then:

$$w = K_P e_{\Delta} + K_D \frac{de_{\Delta}}{dt} \tag{20}$$

The voltage from the orientation controller output will then be added to the voltage obtained from the position control output. Next, the final applicable voltages will be computed as:

$$U_i = v_i + w \tag{21}$$

where v_i is obtained in (18). This voltage is applied to each motor to reach the desired point. Since the system sensitive parts such as electronic board, computer, batteries, etc., may be damaged by rapid rotation of the robot, one needs to apply upper and lower cut-off thresholds for the orientation controller output. Practically, the threshold was set to be ± 10 v. The PID and PD gains were obtained from the two previous cases, and used as first estimation, leaving only fine-tuning to the scene. This was due to the robot working conditions such as friction, and gear boxes clearances and tolerances that were not available in advance and thus not considered in initial modeling. The proper coefficients were then fine-tuned experimentally during each competition. The results showed that for real cases, the maximum changes in the calculated values were bound to ±10% of the original gains values. Therefore, such simplification proved to provide good initial approximation, considerably simplifying final gains tuning.

3.4 Feedback Generation and Self-Localization

The position control method described in the former Section, 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 et al., 1997; Olson, 2000; Talluri & Aggarwal, 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.

We developed and optimized a compound novel method for RoboCup MSL (Middle Size League) in which both visual and odometry information are used to ameliorate a real time, accurate and reliable method (Samani, Abdollahi et al., 2004). Although optimized for soccer player robots, the proposed self-localization method 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 and cheap hardware.

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 com-

plexity 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.

The performance of vision-based self-localization method relies on accurate visual information obtained from the vision module by means of image processing algorithms and techniques. Since goals are of two distinct colours in the play field (yellow and blue), the pixels representing them are distinguished and then their position and angle of observation are extracted accordingly.

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

$$y = y(x_1, x_2, ..., x_n) \tag{22}$$

The variation of y is defined as:

$$dy = y\sum_{i=1}^{n}\left[\frac{x_i}{y}\frac{\partial y}{\partial x_i}\right]\frac{dx_i}{x_i} = y\sum_{i=1}^{n}S_{x_i}^{y}\frac{dx_i}{x_i} \tag{23}$$

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

$$S_{x_i}^{y} = \frac{x_i}{y}\frac{\partial y}{\partial x_i} \tag{24}$$

Figure 10 shows the sensitivity landmark in the field. It is obvious that in certain areas near the corner posts, the sensitivity increases and the accuracy decreases drastically which result in severe errors in these regions. 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.

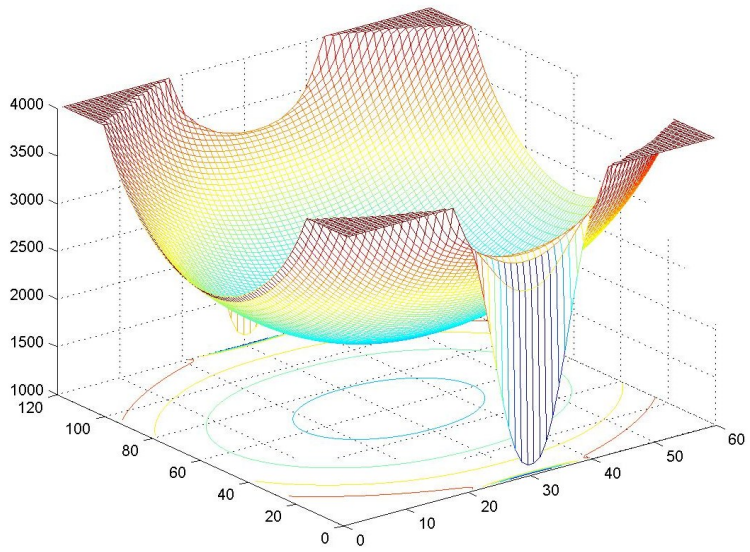
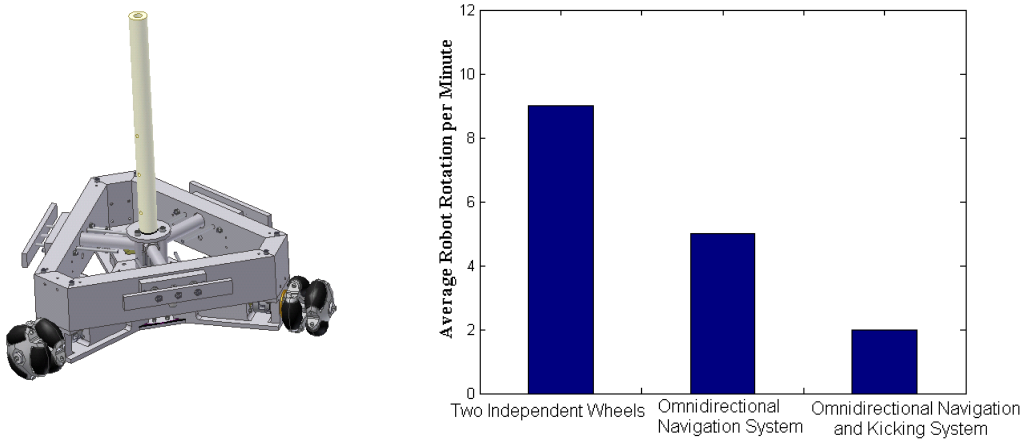


Figure 10: Sensitivity of vision-based self-localization at different location in the field.

3.5 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 helps 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 requires 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 optimize such rotations, we used two extra kicking mechanisms to form an omni-directional kicking system. The position of these kicking systems is shown in Figure 11a. 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. In an experiment, 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.



(a) (b)

Figure 11: (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. Figure 11b shows these numbers for three types of robots, i.e.:

1- Two independent driving wheels mechanism

2- Omni-directional navigation system, i.e. omni-directional wheel mechanism (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 as shown in Figure 11b. 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% percent.

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

3.6 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.

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.

3.6.1 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!

3.6.2 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, 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. Figure 12 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.

3.6.3 AI Core

As it can be seen from Figure 12, 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},* see table 1.
- 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.

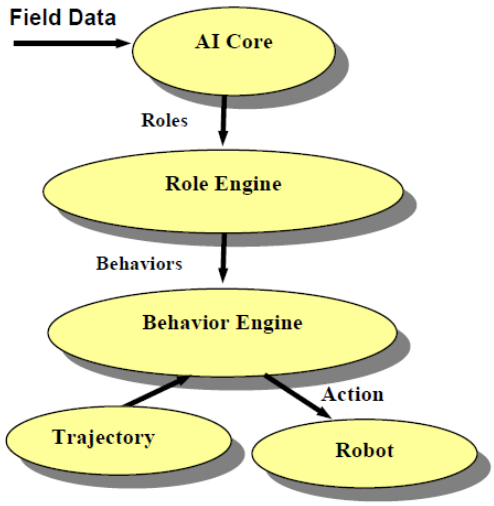


Figure 12: Artificial Intelligence Architecture

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 where different regions are defined in Figure 13.

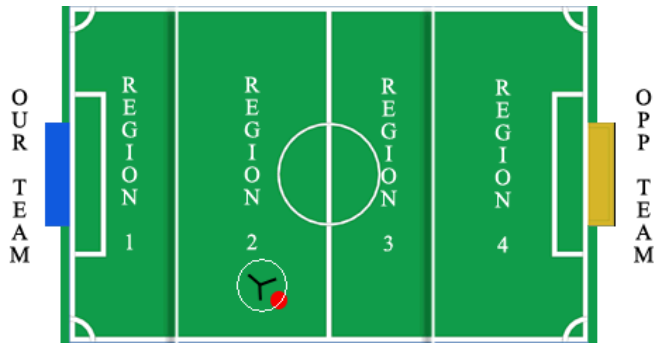


Figure 13: AI core visual module and the defined regions.

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*.

3.6.4 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 behavior returned from the Behaviour Layer (i.e. success or fail) and deciding the action to take according to results returned.

3.6.5 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.

3.6.6 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.

3.7 Experimental Results

In order to evaluate the performance of the position controller suggested in Section 3.3 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 Figure 14) 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 Figure 14).

In the first experiment, the PID constants were set as described in Section 3.3. 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 Figure 14).

In the second experiment, again the PD controller parameters are set to the calculated values for orientation control. The maximum error from the set point angle was $0.03\,\pi$ radians for a complete rotation.

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 Figure 14) while rotating about its Z-axis. The measured errors were between 10 to 12 cm and occurred at points 4, 10, 13 and 17 in curve "A" in Figure 14. 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 Figure 14) 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 14 shows the position of the three robots on the field at different times and Figure 15 presents a picture of the robot tracking on the curves "A", "B", and "C" in the competition field.

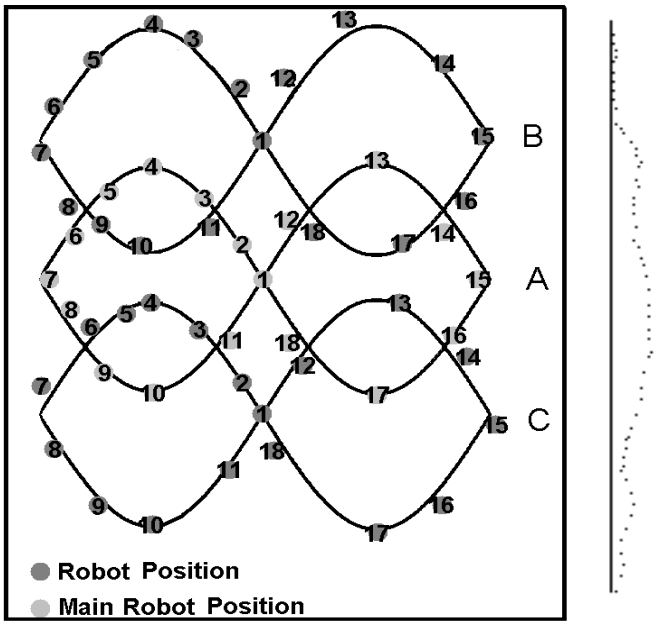


Figure 14: A, B and C depict the robot trajectories. The numbers show each robot position on each curve. The right picture illustrates the straight line followed by the robot

Figure 15: Notations “A”, “B”, “C” are the robots that followed the corresponding “A”, “B”, and “C” curves in Figure 14.

4 Conclusion

In this Chapter, we have discussed the management and technical aspects toward assembling a modern multimodal robotic team. As an example, the related concepts have been presented for a team of Middle size soccer player robots in RoboCup competition. Regarding each aspect, possibilities, challenges and solutions have been discussed. An efficient method for setting up a team consisting of researchers and engineers has been presented with the goal of collaborative cooperation between members of the group.

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. The idea of separating odometry sensors from the driving wheels was successfully imple-

mented. 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. We summarized the proposed PID and PD controllers for position and orientation controls respectively. By using these controllers in the real robot, it was shown that the strategy was appropriate for an omni-directional robot and the combination of methods and techniques described in this Chapter are led to a successful team of soccer player robots.

References

Abdollahi(, A., Samani, H., Ostadi, H., & et .al (2002, 2003 & 2004). I.U.T Persia Team Description Papers. In *International Robocup Symposium Proceedings*.

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, In *IEEE International Conference on Robotics and Automation* (pp. 1925-1930).

Borenstein, J., Everett, H. R., Feng, L., & Wehe, D. (1997). Mobile robot positioning: Sensors and techniques. *J. Robot. Syst.*, 14(4), 231–249.

Cominos, P., & Munro, N. (2002). PID controllers: recent tuning methods and design to specification. In *IEE Proc. on Control Theory and Applications*, 149(1), 46–53.

Fox, D., Burgard, W., & Thrun, S. (1998). Active Markov localization for mobile robots. *Robot. Auton. Syst*., 25(3-4), 195–207.

Hollenbeck, J. R., Moon, H., Ellis, A. P. J., West, B.J., Ilgen, D. R., Sheppard, L., Porter, C. & Wagner, J. A. (2002). Structural contingency theory and individual differences: Examination of external and internal person-team fit. *Journal of Applied Psychology*, 88(3), 599-606.

Jung, M., & Kim, J. H. (2001). Fault tolerant control strategy for OmniKity-III. In *IEEE Int. Conf. on Robotics Automation (*pp. 3370–3375).

Kalmar-Nagy, T., Ganguly, P., & D'Andea, R. (2002). Real-time trajectory generation for omni–directional vehicles. In *American Control Conf.* (pp. 285–291).

Kitano, H. (1997). RoboCup: The robot world cup initiative. In *First International Conference on Autonomous Agents* (pp. 340-347).

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

Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., Osawa, E., & Matsubara, H. (1997). RoboCup: A challenge problem for AI. *AI Magazine*, 18(1), 73-85.

Kitano, H., Siekmann, J., & Carbonell, J. G. (1998). *RoboCup 97: Robot Soccer World Cup I, Lecture Notes in Artificial Intelligence*. Nagoya: Springer-Verlag.

Martinelli, A., Nguyen, V., Tomatis, N., & Siegwart, R. (2007). A relative map approach to SLAM based on shift and rotation invariants. *Robotics and Autonomous Systems*, 55(1), 50–61.

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

Nakano, E., & Koyachi, N. (1993). An advanced mechanism of the omni-directional vehicle (ODV) and its application to the working wheel chair for the disabled. In *Int. Conf. Advanced Robotics* (pp. 277–284).

Olson, C. F. (2000). Probabilistic self-localization for mobile robots. *IEEE Trans. on Robotics and Automation,* 16(1), 55–66.

Paromatchik, I., & Rembold, U. (1994). A practical Approach to motion generation and control omni-directional mobile robot. In *IEEE Int. Conf. on Robotic and Automation* (pp. 2790–2795).

Official website of Robocup: http://www.robocup.org

Samani, H., Abdollahi, A., Ostadi, H., & Ziaie Rad, S. (2004). Design and development of a comprehensive omni-directional soccer player robot. *Int. J. Advanced Robotic Systems*, 1(3), 191–200.

Simmons, R., & Koenig, S. (1995). Probabilistic navigation in partially observable environments, In *Int. Joint Conf. Artificial Intell*. (pp. 1660–1667).

Skrzypczynski, P. (2005). Planning Positioning Actions of a Mobile Robot Cooperating with Distributed Sensors. *Advances in Soft Computing*, 30, 427-434.

Stroupe, A., Sikorski, K., & Balch, T. (2002). Constraint-based landmark localization, In *RoboCup 2002: Robot Soccer World Cup IV* (pp. 239–245).

Talluri, R., & Aggarwal, J. K. (1993). Position estimation techniques for an autonomous mobile robot—A review. In *Handbook of Pattern Recognition and Computer Vision*. Singapore: World Scientific (pp. 769–801).

Thrun, S., Burgard, W., & Fox, D. (1998). A probabilistic approach to concurrent mapping and localization for mobile robots. *Mach. Learning*, 31(1–3), 29–53.

Watanabe, K. (1998). Control of an omni directional mobile robot. In *Second Int. Conf. on Knowledge-Base Intelligent Electronic Systems* (pp. 51–60).

West, M., & Asada, H. (1992). Design a holonomic omni-directional vehicle. In *IEEE Int. Conf. on Robotics and Automation (*pp. 97–103).

Woodward, J., Dawson, S., & Wedderburn, D. (1980). *Industrial organization: Theory and practice*. London: Oxford University Press.

Woodward, J. (1978). Management and technology, *Sociotechnical systems: a sourcebook*. Pfeiffer and Co.