Aria3D Research Proposal for 2007 Competitions

Arman Sarrafi, Golnaz Ghiasi, Kourosh Meshgi, Moloud Shahbazi

Department of Computer Engineering and IT
Amirkabir University of Technology
P.O. Box 15875-4413, Tehran, Iran
{sarrafi, golnaz.ghiasi, meshgi, moloud.shahbazi}@gmail.com

Abstract. In this paper we will present a description of the architecture of our 3D soccer player agents in Aria team and the algorithms we have used for controlling the robots. Since it is a new brand in RoboCup soccer simulation, our development mainly focuses on the low-level controllers for the simulated robots. We have tried to make it possible for the robots to walk smoothly without collapsing. Here, we present a hierarchical architecture for our agents and describe how we narrow it down from high level decisions to joint control commands.

1 Introduction

Soccer simulation is an excellent test bed for researchers in the fields of Robotics and Artificial Intelligence. Humanoid 3D soccer simulation is a new league in RoboCup which benefits from a more realistic model for the simulation of soccer player robots. As a result, low-level control mechanisms has been much more important compared to 2D and 3D-spheres competitions. At the current stage, the problem of making a team of these robots mostly falls in the field of robotics and needs an insightful view to kinematics and dynamics[1].

In Section 2 we will present our proposed architecture. Then, in Section 3 we will mention the algorithms we are going to use through the different layers of our agents' architecture. We will conclude in Section 4.

2 Multi-layer Architecture

Facing this view to RoboCup competitions agent programming, we divide it into four major levels and thus with conquering each of them the main target will be achieved. Robocup is a multi-agent competition. Therefore, the views are:

1. Multi-agent Level View
2. Movement Level View
3. Joint Level View
4. Rotation Level View

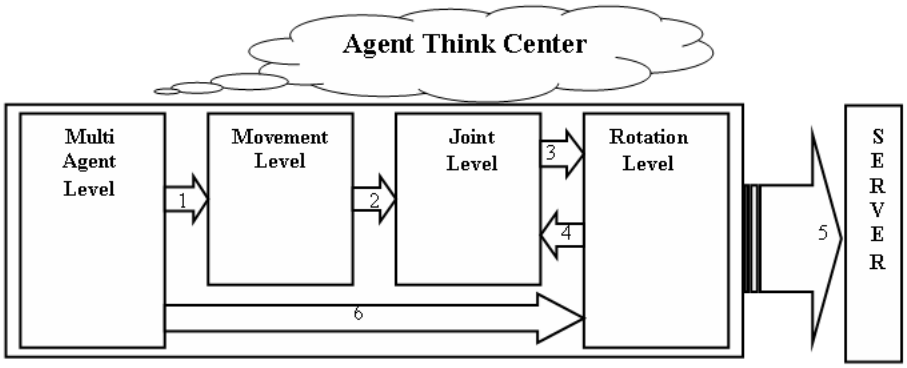

Fig. 1. Hierarchy of Agent Think Center: 1- Decision 2- New Position 3- Joint Angle 4- Body State 5- Angular Velocity[i] 6- Movement Power

Generally, the following sequence happens during a game: A robot makes a decision regarding of field status, ball position, other robots position and formation, etc. This decision is very high-level (i.e. robot decides to cut through a pass between two other robots). This decision makes robot to move from current position (x_1, y_1) to a target position (x_2, y_2). Several rotations in robot joints must be done to able it to pass this way, i.e. a sequence of angles is prepared to joint to let the robot run to target with a definite speed. In order to close or open a joint from x_1 to x_2 it must be supplied with angular velocity and its the final data one must achieve to attend in a match.

To implement these levels, knowing that major decision of robot must be transformed to low level commands which will be executed in server sequentially, the following data structures are used: 1.In Multi Agent Level, robot decides to move from current position to another and the power this movement must have. Regarding this decision, the positions of field must be defined uniquely. On the other hand the state of robot can be movement with ball or movement without ball. The final structure is in the below form. Coordination in field (X, Y) Position of robot regarding the center of the field Direction of body θ The angle between the orthogonal vector of robots chest and the line along the field Power of movement (P); Its a constant value for movement of body without ball which needs balancing, and may be more or less for action that interact with ball such as dribble, shoot ... So the quadruple (X, Y, θ, P) is the product of this level that is delivered to lower level. 2.In Movement Level, the current position of robot will be determined and amount of required movement to reach the target which is designated in upper level can be calculated then. This level calculates Δx, Δy and $\Delta\theta$ and prepare $(\Delta x, \Delta y, \Delta\theta, P)$ to lower level (note that P is transited from upper level). Current state of robot is obtained using visual information of robot and is used as (X_0, Y_0, θ_0) in the calculation. To determine this information visual information will be treated as follows:

3.The purpose of Joint Level is to find a sequence of joint angles that move robot for $(\Delta_X, \Delta_Y, \Delta_\theta)$. Keeping the robot balanced during this sequence is necessary. Found sequence defines change of angles for all joints during a period of time. The main task of this level, in fact, is to fill the sequence of angle changes for joints with the values that previously executed algorithms, for example to make a robot walk a sequence of angles must be applied to all joints. The output of this level will be in the following structure: The soccerbot has 18 joints that 6 joints of them has double move form (universal joints), Thus can be considered as 2 joints. So finally the state of robot can be described with an array of length 24 which the i th element indicates the change of angle in joint i. This array, named x, is initiated with zero which means no change. This array and the P parameter will be sent to lower level in the form $(P, \alpha[24])$. In this level, balance of robot must be maintained regarding the forces which are applied on robots from different directions. To calculate α from x, y and z there are some equations that will be implemented later with balance equations.

4.The goal of Rotation Level is to translate robot change sequence to a chain of commands to each joint. These commands are in the form of angular velocity to joints. Regarding to input of this level $(P, \alpha[24])$, angular velocity can be inferred from the combination of P and $\alpha[i]$, In this layer there is a circular queue for each joint and the value of angular velocity (including zero) is stored in it and the rear and front of each queue that are the same for all queues are controlled centrally. This feature enables the robot to correct its moves, which will be explained later. In each cycle, server reads a command from head of the queue and applies it to corresponding joint. In this layer final angles of all joints as will described in move patterns section are calculated and saved, forming the Robot Body State.

3 Algorithms

3.1 Algorithms of Multi Agent Level

Robocup competitions is multi agent competition, thus multi agent algorithms are needed to decide in these environments. These algorithms are not the point of present work and we will use the 2D and 3D common algorithms of previous years. Obviously, the resulted decisions must be mapped to the (X, Y, θ, P) quadruple.

3.2 Algorithms of Movement Level

At this stage, robots must be aware of their positions in order to determine their distance to goal position and type of required moves. This milestone can be accomplished with assist of robot vision. As it was said before, the output of this level is the triple (X_0, Y_0, θ_0) that indicates the current position of robot and the next changes could be identified regarding target position (X_g, Y_g, θ_g) .

Localization of Robot can be done using agent vision and distance parameter. Global position of flag i: $(X_{f_i}, Y_{f_i}, Z_{f_i})$

Global position of robot: (X_M, Y_M, Z_M)

Relative position of flag i received from server: $(d_{v_i}, \theta_{v_i}, \phi_{v_i})$

$$(X_{f_i} - X_M)^2 + (Y_{f_i} - Y_M)^2 + (Z_{f_i} - Z_M)^2 = d_{v_i}^2$$

We can choose 3 flags out of 8 existing flags of the field to calculate: Determining the global angle of the robot with ground as zero level: First we obtain angle of joint from robot joint sensors. Then we calculate robot global angle. Finally we generate all of the robot position parameters. Head angle regarding to X, Y, Z axis: (θ_G, ϕ_G)

Relational position of flag I and robot:

$$(d_{R_i}, \theta_{R_i}, \phi_{R_i}) = SphericalPolar((X_{f_i}, Y_{f_i}, Z_{f_i}) - (X_M, Y_M, Z_M))$$

$$\theta_{R_i} = Arctan((Y_{f_i} - Y_M)/(X_{f_i} - X_M))$$

$$\phi_{R_i} = Arctan((Z_{f_i} - Z_M)/\sqrt{(Y_{f_i} - Y_M)^2 + (X_{f_i} - X_M)^2})$$

$$\theta_G = \theta_{R_i} - \theta_{v_i}$$

3.3 Algorithms of Joint Level

At this stage robot follows a sequence of moves regarding the requested changes. This sequence, as explained before, can be generated by experience (by generate and test approach) or in algorithmic manner. In the present work, we try to find this sequence in algorithmic way. In each of these ways, joints are fed with sequential values. In this text we model this problem as a Search Problem and then suggest a way to solve it, but not in many details in solution: Initial State The soccerbot has 18 joints that 6 joints of them has double move form (universal joints) , that can be considered as 2 joints. So the state of the robot can be described with these 24 joints. To describe these joints, we consider their angle from the stand up position, called relaxed position from now on, as if we look at robot form its side while its faced to right, the counter clock wise moves are assumed positive while clockwise ones are negative. Angles are also measured in this system.

The other defining parameter of robot state is its direction and current position (as defined in multi agent level implementation). If we assume the initial state of robot as $(x, y, \theta) = (0, 0, 0)$, the current position will be (x_0, y_0, z_0). So, the initial state of search problem includes:

1. displacement of angles from the relax position ($\alpha[i] i = 1, 2, ..., 24$)
2. coordination of robot (x_0, y_0)
3. direction of robot (θ_0)

Operators Assign a valid value to one or more joints. Invalid values are specified in below: Out of Range values: Each joint has some limits in movement that is able to move far to these limits. For example the thigh joint is limited to the range of $[-90, +90]$ and cant accept values of out of this range. Physical Interference of another Object: In the case of unpredicted incident to ground or collision with other parts of robot, the more is evaluated as invalid and cant be performed. Losing Balance: The move is invalid if it results to losing balance or incident to ground (an unpredicted move, it excludes standing up procedure).

Cost Function The length of sequence obviously longer sequences results in more time cost. We assume that other limitations like batteries do not constrain our moves on joints.

Goal Test The goal state of robot including $\alpha_f[i] i = 1, 2, ..., 24, X_f, Y_f, \theta_f$ is compared with current state. This requires calculation of current Robot Body State regarding last operation. There are several last issues on determining initial and goal state:

1. There is no coordination change during in-place action(like standing up) $(x_0, y_0) = (x_f, y_f)$
2. All paths through the goal must be saved, in order to replace them with original ones if conditions changed. For example for walk pattern: we use the sequence with more stable (balanced) states in noisy environment and shorter step in surfaces with less friction.
3. Long distance and curve paths must be split to smaller fixed-length movements parts and then translated to their pattern.
4. The real goal test is applicable by small changes in simulator.
5. The goal state must be a relax state ($\alpha[i] = 0 i = 1, 2, ..., 24$) to ease cascading patterns.

This search problem is looking for compatible joint angle values which can make the desired goal. So the problem can be modeled as a series constraint satisfaction problem (CSP) that work in a special framework. This framework has its own rules that the most important one is maintaining the balance of robot. Finding an integrated approach will be one of our further works. Resulted patterns are stored in tables in Joint Level. When an input comes from upper layer, after being split into different pattern (i.e. diagonal move = rotate direct move rotate) and extracted from corresponding table, and will be sent to lower layer in form of sequence of joint angle changes. As said before, maintaining robot balance and calculating angles from their coordination is possible through robot kinematics. Merging the equations derived from forward and reverse robot kinematics, the following equations are resulted:

In order to control the robot, proper angles are needed to reach a desired state. Assume we want to move point P (it is placed at the end of 2 connected moving arms in 2D scene). Now we want to calculate angles of arms, θ_1, and θ_2 as shown in figure 1. First we assume θ_1, θ_2 are defined and calculate coordination of P:

$X_P = r_1 cos\theta_1 + r_2 cos(\theta_1 + \theta_2)$

$Y_P = r_1 sin\theta_1 + r_2 sin(\theta_1 + \theta_2)$

We can use transformation matrix in place of above equation.

Consider hand of robot with 3 joints. One of these joints has got 2 parameters, so there are 4 parameters $\theta_1, \theta_2, \theta_3, \theta_4$ We will apply these rotations, one by one,

1. First rotation of arm joint 3 (θ_3) around x axis
2. Second rotation of arm joint 1 (θ_1) around y axis
3. Third rotation of arm joint 2 (θ_2) around z axis
4. Forth transmission in amount of length of upper-arm through x axis (r1)
5. Third rotation of arm joint 2 (θ_4) through y axis
6. Sixth transmission in amount of length of lower-arm through x axis (r2)

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_3 & \sin\theta_3 & 0 \\ 0 & -\sin\theta_3 & \cos\theta_3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \cos\theta_1 & 0 & -\sin\theta_1 & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta_1 & 0 & \cos\theta_1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \cos\theta_2 & \sin\theta_2 & 0 & 0 \\ -\sin\theta_2 & \cos\theta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times$$

$$\begin{bmatrix} 1 & 0 & 0 & r_1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \cos\theta_4 & 0 & -\sin\theta_4 & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta_4 & 0 & \cos\theta_4 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & r_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Fig. 2. Transformation matrices

We can have one matrix by reversing the production of above matrices, by multiplying this matrix to position of shoulder and the position of

3.4 Algorithms of Rotate Level

In this level, angular velocity of each joint will be calculated from P and $\alpha[i]$, and then adds to joints queue. Fail detection algorithms in this level prevent total failure in robot actions. In this level, Robot Body State is calculated after applying

angle change theatrically. Real Robot Body State is measured through sensors and then compared with the first one. In the case of not matching (Matching function uses fuzzy algorithms), real state which is derived from sensory information (not balancing, falling down, ...) is stored as robot current body state and send as a feed back to Joint Level in order to finding new angles which robot can recover its state with (standing up from lying on back and front state, bowing to one side to maintain balance, ...)

4 Conclusion

In this paper we presented what we have done for developing a humanoid soccer player agent so far, and what we are going to do in the future. After developing these ideas successfuly, the next step will be moving to a higher level of abstraction for the agents.

References

1. James M. Jeanne: Developing Adjustable Walikng Patterns for Natural Walking in Humanoid Robots. Dr. Adrian Stoica, Jey Propulsion Laboratory, 2004