RoboCup 3D
Simulation League Cyclone3D Team Description Paper 2017

Seyed Behdad Musavi Ahmadi Neishaboori, Amirhosein Izadjoo, Hanieh Sadat Izadpanah, Farzad Jafari Aghdam, Mohammad Nasirifar, Shiva Sadat Mahdavian
Shahid Beheshti University

{sbmahmadi , amirhoseinizadjou , hsizadpanah , Farzadjafari1996 , far.nasiri.m , mahdavianshiva}@gmail.com

Abstract. This paper describes the ideas, and research carried out through novel algorithms used by the Cyclone3D soccer simulation team aiming to optimize the performance of humanoid soccer agents. Our agent's performance is currently based on several physical and AI algorithms used for localization and skills such as walking and kicking which will be briefly explained in this paper.

1. Introduction

Shahid Beheshti University has had a team in soccer simulation 2d. Following these successes, work on RoboCup 3D soccer simulation league kicked off in the form of introducing a team to the 3D soccer simulation league which have begun researching since 2014.The main challenge in 3D soccer simulation is the control of a humanoid robot while satisfying both the physical constraints of the robot and the official rules of a soccer game. In order to achieve this goal a number of physical, AI and learning algorithms have been used which can be optimized through the usage of AI methods. By using simulated robot agents instead of actual NAO robots we obtain several advantages such as being able to experiment on the robots in shorter time and less money expenses. Thus the RoboCup organization's goal of using 11 humanoid soccer bots to defeat an 11-man football team can be achieved much sooner.

This paper is structured as follows: section 2 the system architecture is discussed, followed by agent's behavior and skills in section 3 and future works in section 4.

2. System Architecture

Our overall system architecture (Fig. 1) can be broken-down into several modules that interact with each other. The server communication module is in charge of establishing a two-way connection between the Simspark simulation server and our agent while decoding incoming messages into the agent's world model and encoding outgoing ones so that they are ready to be sent to the server. Our agent layer is responsible for the agent's basic skills and behaviors including localization, standing up, walking and kicking. Our agent's final actions during a cycle are determined using information provided by the server which each agent can access through its own world model. These actions are returned to the connection class ready to be encoded and sent back to the server.

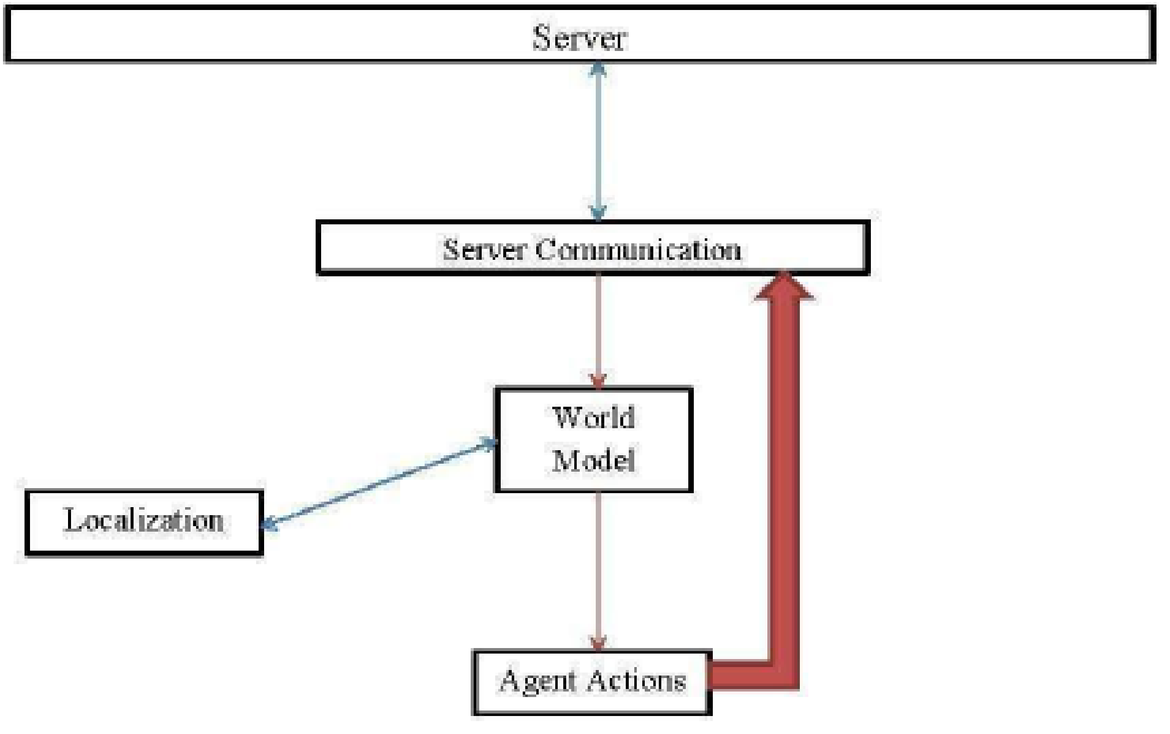

Fig. 1. Overall base architecture

3. Agent Behavior and Skills

In this section we will discuss the methods and algorithms used in order for our agent to localize itself and other objects. We will also discuss how walking skill functions are.

3.1 Localization

This section describes the different methods we used for self-Localization and ball localization

3.1.1 Self-Localization

These methods are used by our agent in order to obtain its position. We prefer the first method and the second method is used if our agent cannot see more than two flags.

3.1.1.1 Localization with two flags or more

Our flags heights are always constant therefore we can always obtain our agent's location by using the coronal plane. If there are more than two flags in our agent's sight, we will pair them up and use this process on each pair and finally in order to obtain our position we will use weighted average on the results of this process.

$$(X - X_1)^2 + (Y - Y_1)^2 + (Z - Z_1)^2 = {r_1}^2$$
$$(X - X_2)^2 + (Y - Y_2)^2 + (Z - Z_2)^2 = {r_2}^2$$

3.1.1.2 Localization with one flag and body direction

The server sends us the position of a flag. We use these flag positions and a gyroscope to find out what our body direction is. Then we use this body direction and flag position to localize our agent's position. Using a gyroscope for a long time will result in a high error.

3.1.2 Ball localization

In order to score a goal, we have to obtain the location of the ball. There are only two situations, we either have the ball in our sight or we don't. In the first condition the server gives us the coordinates of the ball and in the second condition we try to keep the ball in our sight while trying to have a better view on court so that we can see more flags when looking at the ball.

3.2 Walking

Creating a stable, fast and at the same time flexible walking is perhaps the biggest problem in 3D soccer simulation. Currently we use a static walking method based on physical factors such as Center of Mass (COM) and Zero Moment Point (ZMP) which was first proposed by [1] for biped humanoid robots during an IEEE Robotics convention. By satisfying the constraints noted in this method we receive a set of trajectories, afterwards we use inverse kinematics to find out the required angular velocity in order to achieve these trajectories.

3.2.1 Walking cycle

A complete walking cycle is composed of two phases: double-support and single support. The double support phase is when both feet are on the ground and during the single support phase one foot remains steady on the ground while the other foot swings from the rear to the front. In order to achieve stability during these phases we need to satisfy specific constraints which will be discussed in the next section.

3.2.2 Trajectories

In order to achieve a stable walking a set of constraints need to be fulfilled which lead to specific trajectories. This will be discussed in this part. Note that the trajectories we obtain are only for the sagittal plane since we are using a static walking method which was first proposed by [1]. We obtain the following constraint for our ankle trajectory:

$$\theta_a(t) = \begin{cases} q_{gs}(k), & t = kT_c \\ q_b \quad , & t = kT_c + T_d \\ -q_f \quad , & t = (k+1)T_c \\ -q_{ge} \quad , & t = (k+1)T_c + T_d \end{cases}$$

$$x_a(t) = \begin{cases} kD_s & , \quad t = kT_c \\ kD_s + l_{an} sin\ q_b + l_{af}(1 - cos\ q_b) & , \quad t = kT_c + T_d \\ kD_s + L_{ao} & , \quad t = kT_c + T_m \\ (k+2)D_s - l_{an} sin\ q_f - l_{ab}\left(1 - cos\ q_f\right), & \quad t = (k+1)T_c \\ (k+2)D_s & , \quad t = (k+1)T_c + T_d \end{cases}$$

$$z_a(t) = \begin{cases} hg_s(k) + l_{an} & , \quad t = kT_c \\ hg_s(k) + l_{af} sin\ q_b + l_{an} cos\ q_b, & \quad t = kT_c + T_d \\ H_{ao} & , \quad t = kT_c + T_m \\ h_{ge}(k) + l_{ab} sin\ q_f + l_{an} cos\ q_f, & \quad t = (k+1)T_c \\ h_{ge}(k) + l_{an} & , \quad t = (k+1)T_c + T_d \end{cases}$$

And the following constraints are obtained for our hip trajectory.

$$z_h(t) = \begin{cases} H_{h\,min}, & t = kT_c + 0.5T_d \\ H_{h\,max}, & t = kT_c + 0.5(T_c - T_d) \\ H_{h\,min}, & t = (k+1)T_c + 0.5T_d \end{cases}$$

$$x_h(t) = \begin{cases} kD_s + x_{ed}\quad, & t = kT_c \\ (k+1)D_s - x_{sd}, & t = kT_e + T_d \\ (k+1)D_s + x_{ed}, & t = (k+1)T_c \end{cases}$$

$$x_h(t) = \begin{cases} kD_s + \dfrac{D_s - x_{ed} - x_{sd}}{{T_d}^2(T_e - T_d)}\left[(T_d + kT_e - t)^3 - (t - kT_e)^3 - {T_d}^2(T_d + kT_c - t) + {T_d}^2(t - kT_c)\right] \\ \qquad + \dfrac{x_{ed}}{T_d}(T_d + kT_c - t) + \dfrac{D_s - x_{sd}}{T_d}(t - kT_c), \qquad t \in (kT_c, kT_c + T_d) \\ kD_s + \dfrac{D_s - x_{ed} - x_{sd}}{T_d(T_e - T_d)^2}\begin{bmatrix}(t - kT_c - T_d)^3 - (T_c + kT_c - t)^3 - (T_c - T_d)^2(T_c + kT_c - t) \\ -(T_c - T_d)^2(t - kT_c - T_d)\end{bmatrix} \\ \qquad + \dfrac{D_s - x_{sd}}{T_c - T_d}(T_c + kT_c - t) + \dfrac{D_s + x_{ed}}{T_c - T_d}(t - kT_c - T_d), \qquad t \in (kT_c + T_d, kT_c + T_C) \end{cases}$$

Note that on level ground Ɵh is constant and equal to: 0.5

3.2.3 Inverse kinematics

We need to pass our obtained trajectories to the inverse kinematics function in order to receive our joints angular velocities. This function receives input in the form of a matrix containing the foot position compared to the torso therefore we must convert our global foot position to match this requirement. In order to do this, we subtract the global hip position from the global foot position.

3.3 Kicking and Genetic Algorithm

3.3.1 Fast Kick and Force Kick

We used two different methods for implementing kick. One of them is fast kick and the other one is force kick. In force kick we need more time but the ball will go a long distance. Fast kick is faster but its range is low.

3.3.2 kick optimizing

After discussing different types of kicking, we concluded that the best way to develop a kick is using learning or optimization methods. So we checked different learning and optimization methods (RL, Imperialist Competitive Algorithm, PSO, Genetic Algorithm) and we came up with Genetic Algorithm. Our shot has three steps in which the first two steps are static and the third step will be determined by using genetic algorithm. To determine the third step, we had about 2048 chromosomes as our starting population (21 of these shots were written by team members and the rest of the shots were made randomly). We applied genetic algorithm for 112 generations and after this we got to a point that the algorithm didn’t made any better chromosomes.

Because of low speed of learning we made a network on a cluster to continue the process in parallel form. Without a good fitness function, we faced a problem. Although we have a large population, Fitness converges to the local peaks with a low rating. So we needed to improve the genetic algorithm. For example, we use Tournament Selection method but it didn’t give us what we expected.

As a result, we search for a new method to prevent convergence. To prevent our problem from converging to the local peaks or it can come back of this situation, we need to keep our chromosomes as distinguished as possible. In other word we must have more uniform distribution.

So beside our rate function which was based on the distance that the ball will go, we consider another rate which is based on uniqueness of the chromosomes in the population which they are in. As the result, chromosomes can go to the next generation if they have higher rate of others and also be more different than the rest of the chromosomes.

If we define fitness the distance that the ball will go and the rate of uniqueness of the chromosomes, we still have convergence problem. This new function doesn’t guarantee the uniform rate of chromosomes in future generation. According to all of this, we should use a method to ensure us the arbitrary distribution of chromosomes which we want follow a certain probability distribution in all generations. For creating the generations which follow the considered distribution, we use Metropolis algorithm beside Grey Wolf Optimizer algorithm.

4. Path Planning

We needed a plan to go from a point to another in which we can leave the opponent agents. For this problem we decide to use “The Boustrophedon Cell Decomposition”. The “Boustrophedon Cell Decomposition (BCD)” is a method used in artificial intelligence and robotics for configuration space solutions. Like other cellular decomposition methods, this method transforms the configuration space into cell regions that can be used for path planning.

Cellularization algorithms is not suitable because of the mass of information we exchange each cycle. So we made another plan with Bézier Curve. A Bézier curve is a parametric curve used to model smooth curves that can be scaled indefinitely. "Paths", as they are commonly referred to in image manipulation programs, are combinations of linked Bézier curves. Paths are not bound by the limits of rasterized images and are intuitive to modify.

We used Bézier Curve and a simple go-to-point algorithm together. In each cycle we draw a line between our selected agent and the point where we want it goes, called line L. We get the distance of each opponent agent from line L. Assume that the nearest opponent agent to L has distance D and has distance P than our selected agent. If $P < D$ we go to our target point in a straight line. Otherwise we use Bézier Curve method to identify our path.

5. Future Works

We are currently working on our walking system using genetic algorithm. But it’s not completed yet. But we believe that it’ll be completed before the competition. Also we’re currently working on our skill changing and plans.

6. References

1.Huang, Qiang, Planning Walking Patterns for a Biped Robot, IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION, june 2001, VOL. 17, NO. 3

2. Spong, Mark, NOVEMBER 2005, Robot Modelling and Control, ------------, Wiley, 1st Edition

3. Jazar, Reza, Theory of Applied Robotics: Kinematics, Dynamics, and Control, --------, SPRINGER, 2nd Edition

4. J.J. Alcaraz-Jim´enez, D. Herrero-P´erez, and H. Mart´ınez-Barber´a, Motion Planning for Omnidirectional Dynamic Gait in Humanoid Soccer Robots, JOURNAL OF PHYSICAL AGENTS, JANUARY 2011, VOL. 5, NO. 1

5. Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kazuhito Yok oiand Hirohisa Hirukawa, The 3D Linear Inverted Pendulum Model: A simple modeling for a biped walking pattern generation, Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems, Maui, Hawaii, USA, Oct. 29 - Nov. 03, 2001

6. Patrick MacAlpine, Daniel Urieli, Shivaram Kalyanakrishnan and Peter Stone, UT Austin Villa 3D Simulation Soccer Team Description Paper 2011

7. Shahriar Asta, Tuna Sonmez, Onuralp Ulusoy, Alper Alimoglu, Mustafa Ersen, Ozdemir Sozmen, and Sanem Sariel-Talay, beeStanbul RoboCup 3D Simulation League Team Description Paper 2011

8. N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, “Equation of state calculations for fast computing landscape,” J. Chem. Phys., vol. 2, pp. 1087–1093, 1953.