beeStanbul
RoboCup 3D Simulation League
Team Description Paper 2012

Baris Demirdelen, Berkay Toku, Onuralp Ulusoy, Tuna Sonmez, Kubra Ayvaz, Elif Senyurek, and Sanem Sariel-Talay

Artificial Intelligence and Robotics Laboratory
Istanbul Technical University
Computer Engineering Department
Istanbul, TURKEY
http://air.cs.itu.edu.tr/beestanbul
sariel@itu.edu.tr

Abstract. The main objective of the beeStanbul project is to develop an efficient software system to correctly model the behaviors of simulated Nao robots in a competitive environment. Several AI algorithms are being used and developed for contributing to robotics research while also advancing the quality of competitions in 3D Simulation League. This team description paper presents important aspects of the overall system design and outlines the methods used in different modules.

1 Introduction

The beeStanbul project from the Artificial Intelligence and Robotics laboratory (AIR lab) at Istanbul Technical University (ITU) is the first initiative from ITU to participate in RoboCup competitions. Earlier projects in the AIR lab were mainly on cooperative multirobot systems. This challenging project was initiated in 2009 to apply the experience, gained from earlier research on multirobot systems [1], to competitive environments as well.

The beeStanbul team consists of undergraduate and graduate students from the Computer Engineering Department of ITU. The main goal of the team is to contribute to the main objective of the RoboCup project by an efficient design of a software system. The designed software system serves as a basis to apply several high-level intelligence, reasoning and learning methods.

beeStanbul team have participated in the RoboCup competitions since 2010. The team won several games during the competitions. The software architecture of the system has been revised and several promising motion types have already been developed since 2010. With the completion of the high-level modules, the team is expected to achieve the project's main goal.

The organization of the rest of the team description paper is as follows. Section 2 presents the software system architecture for simulated Nao robots in the SimSpark simulation environment. Motions and behaviors available for an

agent are presented in Sections 3 and 4 respectively. The developed team-strategy and planning methods are discussed in Section 5, followed by the conclusion in Section 6.

2 System Architecture

The overall software system consists of several modules that interact with each other (Fig. 1). The Server Layer perfoms a two-way communication with the SimSpark server, decodes incoming messages and encodes outgoing messages. In order to carry out these operations, the SimSpark utilities and rcssnet library, provided by SimSpark, are used.

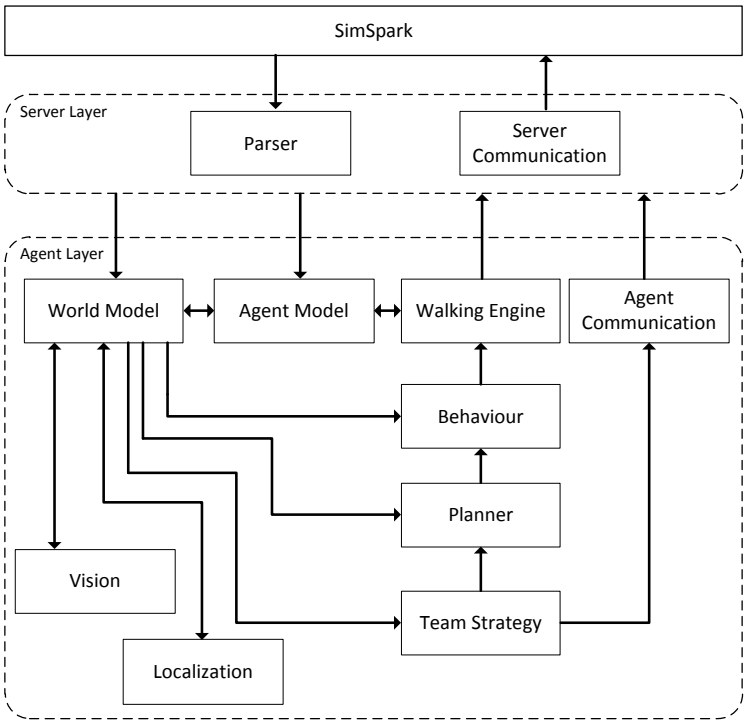

Fig. 1. Overall Software Architecture.

The Agent Layer is responsible for performing the main functionalities of an agent. Each agent maintains its own world model for the environment and the agent model for its own state. The Localization module is responsible for determining the correct pose of the agent using information gained from seen flags. Localization is performed by using triangulation method with Kalman

Filter using distance and angles to a flag. In situations where triangulation is not possible, odometry information is used. The Vision module is responsible for determining the positions of the observed objects in the environment. Based on the observation history, a semantic analysis of the objects (including the opponents) and predictions are made. The agent decides on a behavior based on its agent and world models, the selected team strategy and the assigned role for itself. Walking engine module determines how the agent should walk given a target position and an orientation. Behaviour module is responsible for executing certain commands like move-to-target or dribble-to-target. Commands for the corresponding behavior is sent to the Server Layer to generate the desired effect. Simultaneously, either informative or query messages might be sent to teammates based on the selected team role.

3 Motions

In our earlier implementation, we used the Partial Fourier Series (PFS) model [2] for motions in coronal plane [3]. This implementation includes seven main body motion types and special transition functions for smoothly switching between two arbitrary motion types. Although robots are able to walk fast enough with this model, transitions cause delays in reactive behaviors. Therefore, we have shifted our focus on implementing omnidirectional motions using the motion model given in [4]. The new implementation will include static and dynamic motions. Static motions include standing up or searching the ball with head as predefined motions. On the other hand, dynamic motion patterns are generated dynamically depending on a target. These correspond to different types of walks, turns and alignments. We're planning to optimize the parameters with an optimization algorithm such as Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [5] that is almost a parameter-free algorithm.

4 Behaviors

There is a layered motion selection architecture for motions, behaviors, plans and cases in the team strategy. Modularity and ease of maintenance could be achieved by using such a hierarchy. While motions operate on the lowest level, behaviors are constructed as sequences of low-level motions. Some behavior examples include *move-to-target*, *turn-to-target* and *dribble-to-target*. On top of these behaviors, static plans are constructed in the form of behavior sequences. The team strategy component uses these plans in a modular way. A sample instantiation of a plan is given in Figure 2.

5 Team Strategy and Planning

Team strategy involves four sequential processes to determine the target of an agent [6]. Figure 3 presents the main modules for the team strategy. Initially

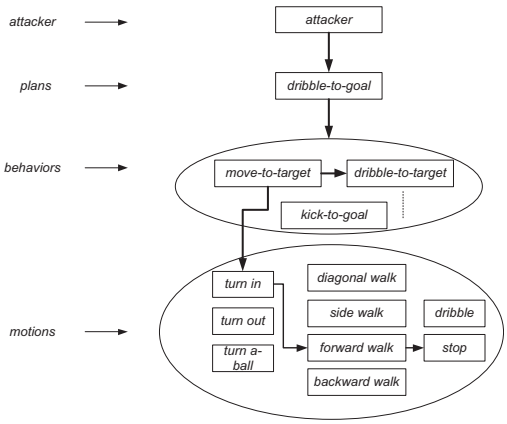

Fig. 2. A sample instantiation of the *dribble-to-goal* plan.

two groups, namely attackers and defenders, are formed by using a group formation strategy. The role of each agent is determined based on these groups. The attackers group involves the forward and the midfielder agents while the defenders group involves only the defender agents. There are three different planners designed for four different roles. Forward planner uses a finite state machine (FSM) that moves to the ball regardless of the teammates and chooses an appropriate action from kick and dribble behaviors. Goalkeeper planner always stays in team's penalty area and tries to form an obstacle against opponents to prevent them from scoring. Final planner is a FSM both used by defenders and midfielders. Only difference in the FSM for both roles is the target calculation. Based on the assigned role of an agent, the corresponding planner is activated. A

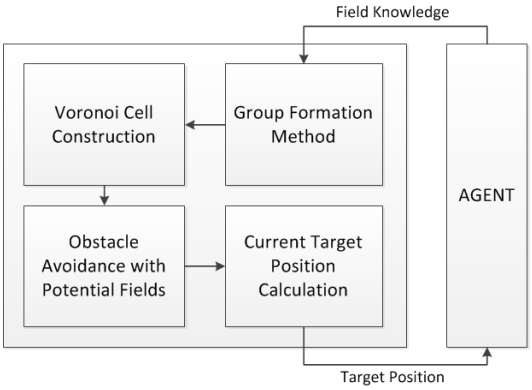

Fig. 3. General structure of the team strategy.

case-based group formation method is used to determine the number of defender agents and midfielder agents dynamically. Since two agents are assigned to the goalkeeper and the forward roles, the remaining seven agents are to be assigned to these roles. Instead of using a predetermined number for these roles, a case-based method is applied to determine the best separation. Equation 1 shows the information that is considered in selecting a case.

$$Case = (BallPos., Score, AgentPos., Numberofmidf., Numberofdef.) \quad (1)$$

The midfielder and defender agents need to position themselves for maintaining

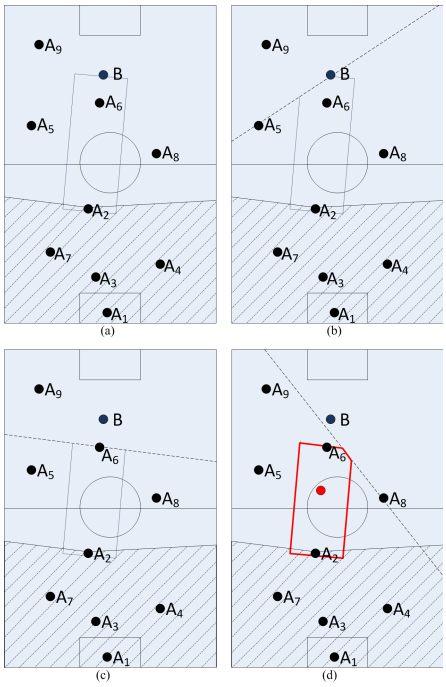

Fig. 4. Step-by-step calculation of the Voronoi cell for an agent (a_2).

close proximity to the forward agent and defending the goal respectively. This is accomplished by a distributed Voronoi cell construction approach in which each agent calculates its own cell independently from that of the others. Therefore, every agent has a differently shaped cell and these can overlap. The time complexity of the method is $O(n^2)$ where n is the number of agents in the team.

Figure 4 illustrates the iterations for calculating the final cell and the corresponding target as the center of this cell for agent #2 (a_2), which is a midfielder and draws its initial cell according to the ball position. As mentioned before, only teammates in the viewpoint of the agent are considered. The area that is

out of a_2's point of view is shown as the shaded area. Figure 4 (a) shows the initial cell construction by considering the ball position (P_B). In Figure 4 (b), (c), and (d), the cell is modified according to the locations of a_5, a_6, a_8 and a_9, respectively. The line for a_9 doesn't have any intersection points with the current cell, so it doesn't make any changes in the cell. The final Voronoi cell of a_2 is shown with the red frame and the center of that cell is marked with a red point in Figure 4 (d).

The method is used for both midfielders and defenders. Defenders create their cells with the same algorithm, but their initial cell is calculated according to the midpoint of the line connecting the ball position and the center of the team's goal position while midfielders use the ball location. After constructing the cell for itself, each agent determines the center of the cell as its new target. Agents become closer to each other by using this strategy, which is more beneficial for attacking in soccer.

6 Conclusion

This team description report outlines different parts of the developed software system for beeStanbul robots. The beeStanbul is an ongoing project and the low-level modules of the architecture were designed and implemented. Current work includes designing the omnidirectional motions. The future focus will be on the use of efficient team coordination strategies with the designed motions and the integration of learning capabilities for robots.

References

1. Sariel-Talay, S., Balch, T.R., Erdogan, N.: A generic framework for distributed multirobot cooperation. Journal of Intelligent and Robotic Systems **63**(2) (2011) 323–358
2. Asta, S., Sariel-Talay, S.: Nature-inspired optimization for biped robot locomotion and gait planning. In: Proceedings of the 6th European Event on Nature-inspired Techniques in Scheduling, Planning and Timetabling. evoSTIM (2011) 434–443
3. Picado, H., Gestal, M., Lau, N., Reis, L.P., Tomé, A.M.: Automatic generation of biped walk behavior using genetic algorithms. In: Proceedings of the 10th International Work-Conference on Artificial Neural Networks: Part I: Bio-Inspired Systems: Computational and Ambient Intelligence. IWANN '09, Berlin, Heidelberg, Springer-Verlag (2009) 805–812
4. Graf, C., Rfer, T.: A closed-loop 3d-lipm gait for the robocup standard platform league humanoid. In Zhou, C., Pagello, E., Behnke, S., Menegatti, E., Rfer, T., Stone, P., eds.: Proceedings of the Fourth Workshop on Humanoid Soccer Robots, IEEE-RAS International Conference on Humanoid Robots (Humanoids-10). (2010)
5. Igel, C., Hansen, N., Roth, S.: Covariance matrix adaptation for multi-objective optimization. Evolutionary Computation **15**(1) (2007) 1–28
6. Ulusoy, O., Sariel-Talay, S.: Distributed team formation for humanoid robot soccer. In: Proceedings of the 4th International Conference on Agents and Artificial Intelligence, Vilamoura, Algarve, Portugal (2012)