beeStanbul
RoboCup 3D Simulation League
Team Description Paper 2010

Bahadir Acar, Shahriar Asta, Sukru Avcuoglu, Maltam Hosseinzadeh Sarabi, Itauma Itauma, Zehra Kavasoglu, Can Oztokmak, Ozdemir Sozmen, Nurdan Topaloglu, 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. The challenging and the most time consuming part of the project was the design phase of the motions. However, these motion models were successively developed towards the achievement of the main goal. 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 AIR 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 [1–3], 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 contributing for the main objective of the RoboCup project by presenting an efficient software system implementing several promising approaches which will be successful during the main competition. The designed software system will serve as a basis to apply several high-level intelligence, reasoning and learning methods as well as to improve semantic predictions for reasoning.

The organization of the rest of the paper is as follows. Section 2 presents the software system architecture for simulated Nao robots in the SimSpark simulation environment. Localization technique applied for robots is presented in Section 3. Section 4 outlines the developed semantic analysis for reasoning on

decision conditions. The input for the vision module to perform this analysis is the incoming vision information from the server. Different behaviors and corresponding motions are illustrated in Section 5. Section 6 presents the designed planning strategy. The distributed coordination and team strategy that is employed by each agent using its own agent model is discussed in Section 7, followed by the conclusion in Section 8.

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.

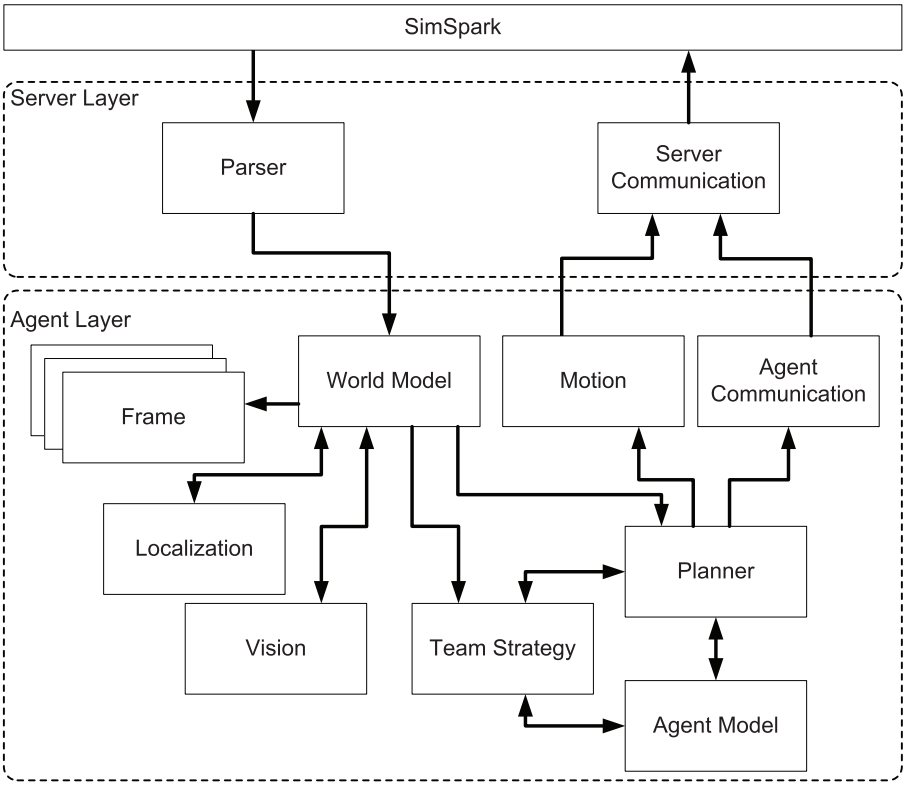

Fig. 1. Overall Software Architecture

The Agent Layer is responsible for performing the main functionalities of a robot. 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 robot given an observation history. The short term memory (in accordance with the observation history) of the robot is maintained by ten consecutive frames in the world model. The Vision module is responsible for determining the positions of the seen objects in the environment. Based on the observation history, semantic analysis of the objects (including the opponents) and related predictions are made. The robot decides on an action based on its agent and world models, the selected team strategy and the assigned role for itself. The motion command 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 role. The planner for the goalie is different than that of the field players.

3 Localization

The global pose of a robot is determined based on landmark-based triangulation. If three flags are observed, the robot localizes itself correctly based on the standard formulation. When only two or less flags are seen, the previous calculated location information is used in the triangulation. Therefore, dead-reckoning from no-flag zones to locations in which flags are seen is performed successively by considering previously calculated pose information. A sample triangulation is given in Fig. 2, in which the robot localizes itself by using distance information from one corner flag and two goal flags. Even when only two flags are seen, the robot localizes itself by using its short term memory. A more robust localization method by using Kalman Filter ([4], [5]) was implemented and it will be used in the final version of the system.

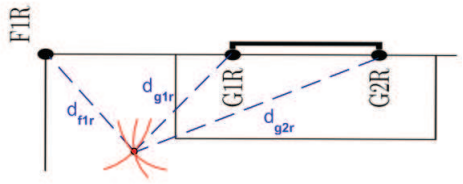

Fig. 2. The robot localizes itself based on the incoming distance values. When the flags cannot be observed, the prediction is made based on the history.

4 Vision Semantics

The Vision module is responsible for identifying different objects in the field of view. Based on the observation history, semantic analysis of the objects (including the opponents) is made, and beliefs are formed. Semantic analysis is needed to infer some useful information for the planner to efficiently select the best action in the current situation. This useful information is related to some boolean flags (e.g., whether the robot is facing the opponents goal), ball location and its semantic state (e.g., the ball is coming from the right side). Predictions are made for the movement direction of the objects (i.e., the ball and the opponents) and their velocities. To make such an analysis, an observation history (short term memory) is stored by means of the registered frames (environment information) for up to ten cycles. The estimations are made by using the Kalman Filter Model [4] for every object to be tracked. By means of these estimations, passive stability of the robot is improved and the required underlying structure for reflexive behaviors is formed (e.g., selection of the dive right behavior for the goalie when the ball is coming towards the right side). Although the relative locations are sent by the server, the global coordinates of the objects are calculated by using localization information. The future work includes cooperative object localization by using incoming information from other teammates.

5 Motion

Primitive behaviors that make up high-level plans and team strategies are determined for robots. These behaviors are encoded as consecutive sequences of joint angles to generate different motions for robots. Some videos of these behaviors are available online [6]. All goalie behaviors, kick and stand up behaviors are generated by a detailed analysis and tuning of the joints on the simulator. Hence, the joint angles for different phases of these behaviors are scripted.

Three different types of kick behaviors were developed: a side kick (Fig.4), a tunable straight kick and a shoot. These behaviors were generated by fitting relevant joint angles to sinusoidal phase function. The tunable (for short-long distance) straight kick is designed for dribbling and passing to another teammate. The shoot directly targets to score a goal. The side kick is designed to be used whenever the position of the robot and the goal location is appropriate for scoring a goal from a side position. Left and right side versions of all these kicks are available. The side change (Right-Left) for a behavior can be performed by switching the behaviors of the foot and changing the sign of the values for only joint angles which change in the y axis.

A sample motion model is given for the shoot behavior as follows. The motion is divided into four different phases. The joint angle calculations for the first two phases are performed by sinusoidal fitting at specified intervals. The target angle of a joint, θ_{target}, is evaluated by the following equation where the $\sin(wt + p)$ is a sinusoidal signal with phase p, a is the amplitude, θ_{shift} is a shift value.

$$\theta_{target} = a * \sin(wt + p) + \theta_{shift} \quad (1)$$

The angle change that will be applied to a joint is calculated based on the error between the target joint angle and the current joint angle by the following equation.

$$\Delta\theta_{joint} = (\theta_{target} - \theta_{current}) * 0.1 \quad (2)$$

The third phase is designed in a different way than the previous phases since the robot foot needs to be accelerated in this phase. An experiment was performed to determine the maximum angular velocity for the knee joint. The obtained experimental values are illustrated in Fig. 3. Based on these results, the joint angle changes to accelerate the foot at this step are calculated by

$$\Delta\theta_{joint} = w_{max} * (p_{kick}/100) \quad (3)$$

where w_{max} is the maximum angular velocity and p_{kick} is a tunable parameter to determine the speed of the shoot. After this phase, a sinusoidal fitting phase as in initial phases is applied to end the motion.

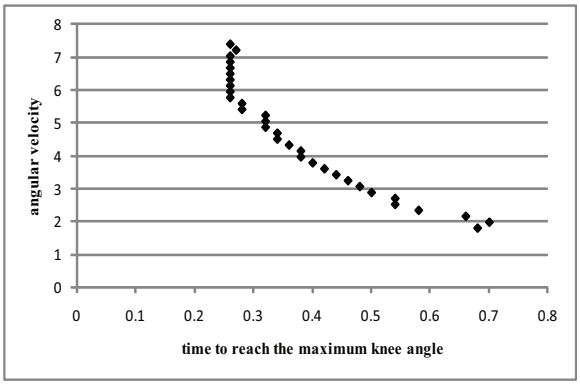

Fig. 3. Experimental results to determine the maximum angular velocity for the knee joint.

An active balancing walk model [7] was investigated and performed for walk behaviors. In the initial phase, the walk trajectories of the robot are determined by a Center of Mass (COM)-based model and, then, the joint angles are inversely calculated to perform the specified walk pattern. However, in the current implementation, walk behavior joint angles are adopted from Robotoos 3D Team joint angle parameters [8] which are provided as text files. Move forward, move backwards and sidewalk behaviors are developed by using these parameters.

A stop behavior is designed to make robots switch from its current behavior to another stably. For the cases when stability cannot be maintained, two different stand up behaviors (i.e., from the front and the back) are developed. These stand up behaviors were generated by directly tuning relevant joint angles for fast and stable recoveries. Incoming perceptual inputs related to both gyro and foot

Fig. 4. The robot position after the side kick

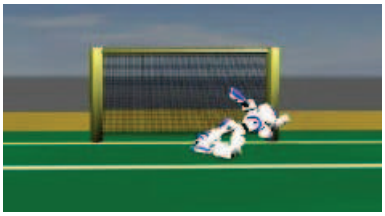

Fig. 5. A goalie dive position

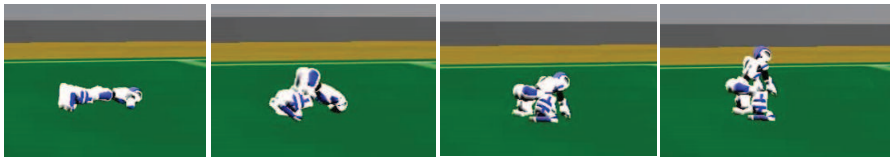

Fig. 6. Different phases of the stand up behavior

pressure values are used to detect whether the robot is fallen down. The side of which it is fallen down can be decided by using gyro information. Foot pressure value is also used to determine whether the stand up behavior is succeeded. Sample gyro values for y and z axes are given in Fig. 7, illustrating changes in their values when the robot falls down (left side) during walking. In this instable case, x axis gyro values do not change, therefore, not illustrated in the figure.

Goalie has dive behaviors (Fig.5) to protect either sides of the goal which can be accompanied by stand up behaviors (Fig.6). Furthermore, a goalie position for protecting the goal was determined and scripted as a motion. However, integration of this motion into the goalie planner has not been performed yet. Current versions of these behaviors were successfully implemented in SimSpark. The future work includes improving these behaviors by applying reinforcement learning techniques on them.

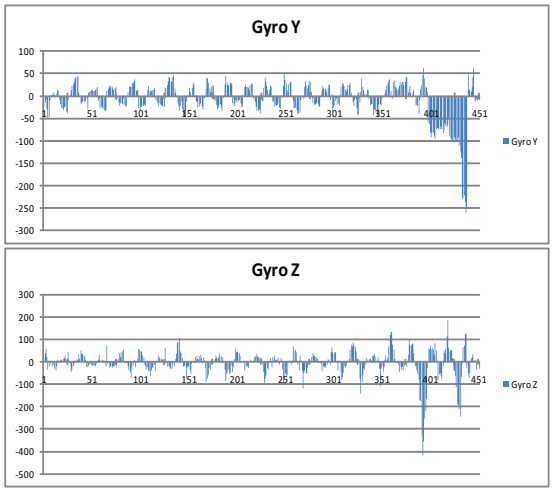

Fig. 7. Changes in the gyro values (in y and z axes) when the robot falls down (left side) during the walk behavior.

There are three head motions which can be executed simultaneously with the body motions. These are track object, search and reset.

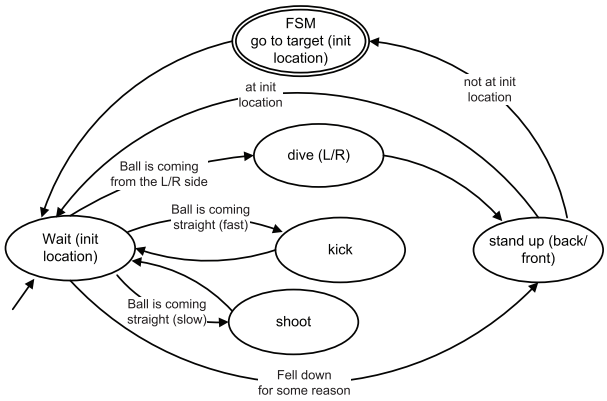

Fig. 8. Goalie FSM

6 Planning

A priority-based action selection strategy is adapted in the design of the planner. This approach is used to determine the most appropriate action for the robot in a given situation. Although the priorities of the actions are fixed, their confidence values are determined based on a list of predicates which change during runtime. These predicates have different impacts on the confidence values of the actions. Reflexive behaviors have usually priorities higher than more complicated behaviors. As a simple example, the stand up behavior has a higher priority than the search behavior.

The goalie planner, on the other hand, is designed as an FSM, illustrated in Fig. 8. According to this model, the goalie never leaves the goal location unless it falls down or the server state is *goalkick* for the team. It has a specific goalie position and two-sided dive behaviors. The goalie planner uses global localization and estimated ball position information from the world model to decide on an action.

A case-based playbook is being designed for the 3D simulation league. A case library is constructed for different situations of the world state. By applying a case-based reasoning approach, this library will be extended in the future. An HTN type of hierarchical playbook decomposition strategy will also be integrated into this approach.

7 Team Strategy

Since one of the players is directly assigned to the goalie role, its own planner is used for its strategy. The remaining two robots may be in one of the following roles: attacker, supporter or defender. This decision is simply made based on the estimated ball position and velocity, and the global localization information. Based on the selected team role and the current situation, the appropriate high-level behaviors are executed based on the priority-base planner. Besides the regular behaviors, different behaviors (e.g., ball search, flag search, etc.) are developed for the robots to correct their world model including localization and vision semantics information.

Role assignment is performed by a single-item auction method [1] where a role is assigned based on the bids by the robots. An auction phase is performed whenever needed. In some situations, in which the roles are obvious, there is no need to offer an auction. This is due to the communication limitations of the simulator. The role assignment and bid calculation is based on the area that the field player is in (Fig. 9) and confidence values for the specific behaviors. An HMM-based opponent modeling method is also being designed for recognizing high-level plans of opponents.

8 Conclusion

This team description report outlines different parts of the developed software system for beeStanbul robots. The beeStanbul is an ongoing project and promis-

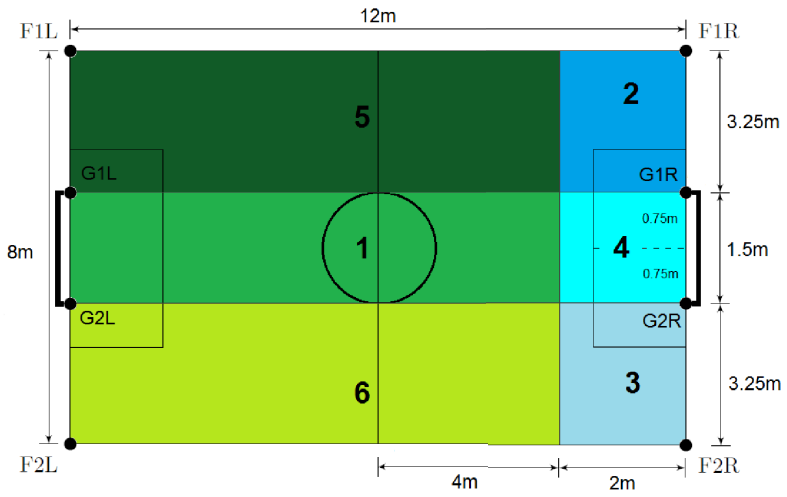

Fig. 9. In different zones, the strategies are selected accordingly.

ing results have been obtained in the current version. The future work includes improvements on some motions and their integration to the system. After revising some of the high-level team strategies, the base code will become a test-bed to apply high-level intelligence and learning capabilities for soccer robots.

References

1. Sariel, S.: An Integrated Planning, Scheduling and Execution Framework for Multi-Robot Cooperation and Coordination. Phd thesis, Istanbul Technical University, Turkey (2007)
2. Sariel, S., Balch, T., Erdogan, N.: Multiple traveling robot problem: A solution based on dynamic task selection and robust execution. IEEE/ASME Transactions on Mechatronics **14**(2) (2009) 198–206
3. Sariel, S., Balch, T., Erdogan, N.: Naval mine countermeasure missions: A distributed, incremental multirobot task selection scheme. IEEE Robotics and Automation Magazine **15**(1) (2008) 45–52
4. Welch, G., Bishop, G.: An introduction to the kalman filter. Technical report, Chapel Hill, NC, USA (1995)
5. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. The MIT Press (2001)
6. beeStanbul Media Files 2010. http://air.cs.itu.edu.tr/beestanbul
7. Graf, C., Härtl, A., Röfer, T., Laue, T.: A robust closed-loop gait for the standard platform league humanoid. In Zhou, C., Pagello, E., Menegatti, E., Behnke, S., Röfer, T., eds.: Proc. of the 4th Workshop on Humanoid Soccer Robots at the 2009 IEEE-RAS International Conference on Humanoid Robots. (2009) 30 – 37
8. Robotoos Team Website. http://www.robotoos.com/ (2010)