The Sixteenth International Symposium on Artificial Life and Robotics 2011 (AROB 16th '11),
B-Con Plaza, Beppu,Oita, Japan, January 27-29, 2011

A Visual Debugger for Developing RoboCup Soccer 3D Agents

Yosuke Nakamura and Tomoharu Nakashima

Osaka Prefecture University, 1-1 Gakuen-cho,Nakaku,Sakai,Osaka
(Tel : 81-72-254-9351; Fax : 81-72-254-9351)
(nakashi@cs.osakafu-u.ac.jp)

Abstract: In this paper, we introduce a visual debugger that helps us develop soccer agents for RoboCup Soccer 3D Simulation. The visual debugger enables us to graphically monitor the internal state of a soccer agent and the soccer field such as joint angles, the position of objects, and text messages. We employ a server/client framework where the debugger acts as a server while the agent acts as a client. A soccer agent connects to the debugger using TCP/IP and sends the information about the field and the internal status. The information that is sent from the soccer agent to the visual debugger consists of three parts: visible objects of the soccer field, the joint angles of the soccer agents, and text messages from the agents. These are shown in separate components on the screen of the debugger. The debugger draws the current pose of the soccer agent from the information on the joint angles that is sent from the soccer server. Text messages are used as a debugging message. The developer of soccer agents are allowed to check if the developed agent works properly through the screen of the visual debugger. A soccer agent that is manually controlled using a game-pad is also included as a part of the debugger. Each of the above features is explained in detail.

Keywords: RoboCup, soccer robot, multi-agent system

I. INTRODUCTION

Soccer simulation league is one of the oldest leagues in the RoboCup competitions. The main aim of the soccer simulation league is to develop decision making systems that achieve intelligent behavior in both high and low levels. It is expected that successful decision making systems are translated to that of real robots. The developers of soccer agents in the simulation league have proposed various techniques from both top-down and bottom-up manners. For example, Stone[1] proposed a layered approach to achieve a complex decision system. FC Portugal proposed a flexible formation model that is based on the ball position[2]. These are categorized as top-down approaches. On the other hand, Stone et al.[3] proposed a neural network-based approach for learning low-level skills. Sean et al.[4] also showed that the team formation is successfully evolved by a genetic programming. A fuzzy reinforcement learning method is proposed for a ball intercept task by Nakashima et al.[5].

In the early years of the RoboCup soccer simulation league, soccer matches were played on a virtual two-dimensional field. The first prototype of 3D simulation was proposed by Oliver[6] in 2003. The community of the 3D simulation has rapidly grown up since then. In 2005 the first competition of 3D simulation was held in Osaka, Japan. The soccer agents were modeled as a sphere object with a kick device. In 2007 bipedal humanoid robot model was employed for soccer agents. This made the development of soccer agents very challenging because not only intelligent decision making but also the movement of joints have to be considered when implementing even lower level skills. The standard way to check the behavior of developed agents is to use the soccer monitor, which is included in the package of the soccer server[7]. Although the soccer monitor is useful to check the movement of joints of soccer agents, it is not possible to check the internal status of the soccer agents such as vision, ground force, body rotation, and their decision making process. Furthermore the soccer monitor does not record the movement of soccer agents during the course of the game. Thus it is not possible to play back the match even if we want to retrospectively watch the behavior of the soccer agents. These things make the development of 3D soccer agent very difficult. Even if we use files to replay a match, the internal status of agents such as sensory information and the intention of agents cannot be shown and we only have to guess them from the behavior of the agents.

In the 2D simulation league, several high functional debuggers have been proposed. For example, soccerwindow2[8] has useful features such as monitoring internal status, modeling team formation based on the ball position. The developer of soccer agents can also check the decision making process inside the soccer agents through the messages that are sent from the soccer agents to soccerwindow2. Furthermore, soccerwindow2 has a play-back function where the developer can check the behavior of the soccer agents at the past time steps. SoccerScope[9] has similar useful features as soccerwindow2, though SoccerScope can analyze the game logs and give statistics of the game such as the number of successful pass, ball possession rate, and the total number of pass, dribble or shoot. In contrast, in 3D simulation league,there are no such tools available in public. Therefore, we have developed a visual debugger as a development support tool that connects to an agent, displays sensor information and checks the agent's behavior.

II. ROBOCUP SOCCER 3D SIMULATION LEAGUE

RoboCup Soccer 3D simulation league is the project where autonomous agents play soccer in a virtual 3D soccer field. The first prototype of 3D simulation was proposed in 2003. In 2005 the first competition of 3D simulation was held in Osaka, Japan. In this competition, the soccer agents were modeled as a sphere object with a kick device. In 2007, a bipedal humanoid robot model was employed for soccer agents. This made the development of soccer agents very challenging because not only intelligent decision making but also the movement of joints has to be considered when implementing even lower-level skills. The humanoid robot which was used in the RoboCup world competition 2008 is given in Fig. 1. This robot model is based on the Nao[10] which is used in the standard platform league. In the RoboCup world competition 2009, one team consisted of three robots. Six-vs-six games were played in RoboCup 2010.

Fig. 1. Humanoid robot

Each soccer agent is autonomously controlled in the 3D simulation league. A game is divided into two halves and each half has 300 seconds according to the official rule. Since one time step is 20 ms, a half consists of 15000 steps. During the simulation, the soccer server sends a message to agents every time step. The message includes joint angles, the value of gyro sensor, the value of foot force resistance sensor, the position of visible objects, and say-messages from other agents. The gyro sensor is equipped in torso. It senses the orientation of the torso in the global coordinates. The foot force resistance sensor informs about the force that acts on a foot. This sensor is equipped in the bottom of feet. A soccer agent can also receive the position of visible objects (the ball, other agents, goal posts, and landmark flags) from the soccer server. An agent is equipped with the camera on its head and the server sends the position of objects which are in the sight of the camera. The angle of sight had been 360 degrees until the world competition 2008 and agents could see all objects from everywhere. However, in the 2009 competition, the angle of the sight was changed to limit to 120 degrees and thus agents have to turn neck and search for objects to receive the position of objects from as broader area as possible. An aural sensor gets one message from both teams every 3 time steps. Agents make a decision based on the above information. Furthermore, the decision made by an agent must be converted to the necessary changes to the joint angles by the agents themselves. Agents send joint change rates to the server. There are 22 joints in a Nao robot. To control this robot perfectly, an agent program is required to control 22 joint angles every time step.

The RoboCup Soccer 3D Simulation Server is available in a Sourceforge project[7]. The package includes the soccer server, the sample agent, and the soccer monitor. The soccer monitor is the standard way to check a behavior of agents. But the monitor shows the soccer field in the third person viewpoint and we can only check the state of joints from their pose. Therefore, a software tool that shows the internal state of agent is required. In this paper, we develop a visual debugger as a development support tool that connects with an agent, displays the value of sensors and checks the agent's behavior. We also develop a soccer agent that can be controlled by a game-pad as a part of the debugger. In the next section, we introduce the visual debugger in detail.

III. VISUAL DEBUGGER

1. Overview

The visual debugger is connected to a soccer agent via TCP/IP and receives information on the field and the agent's internal status every cycle. The functions that the visual debugger provides make it easier for us to develop agents. A screen shot of the visual debugger is shown in Fig. 2. The soccer agent can send a one-line message to the debugger at each time step. The message contains the information on the state of the field and the agent graphically. The visual debugger graphically shows the information on the display.

Fig. 2. The visual debugger

2. Debug Message

The one-line message from the soccer agent includes four pieces of information, that is, game state, visual information, joint angles, and gyro information. We explain each of the above information in detail.

- **Game State**
  The game state information consists of the game time, the playmode, each team's name, and each team's score.
- **Visual Information**
  The agent receives the visual information from the soccer server in three dimensional polar coordinates. The visual information is then sent to the visual debugger as a three dimensional vector in Cartesian coordinates. This information provides the position of visible objects.
- **Joint Angles**
  Joint angle includes the joint angles of the agent at the current time step. One message format corresponds to one joint angle.
- **Gyro Information**
  Gyro information includes the value of the gyro sensor. The gyro sensor shows the angular velocity of agent's torso.

3. Visible Objects in the Soccer Field

The debugger shows visible objects in the soccer field within the agent's view cone as explained in subsection 3.2. The agent sends the debugger a message that contains field information about whether each object is in the eyesight of the agent or not and if it is visible, its position is also included in the message. The position of visible objects is given in the local coordinates of the agent. The debugger converts it into the absolute coordinates (i.e., the origin is set to the center of the field) and draws it in the overhead view of the soccer field. Figure 3 shows an example of the overhead view of the soccer field. In Fig. 3, the arrow in the rightside of the field indicates the position and direction of the agent's torso, two lines from an arrow to rightside and to top of the figure shows the eyesight of the agent, the white circle indicates the ball, the light gray square indicates a teammate player, and the dark gray square indicates an opponent player. The debugger draws each of the above objects when it is in the agent's eyesight. The position of landmarks such as flags and goal posts are described by gray circles or black circles. The positional relation among landmarks are pre-specified by the soccer server according to the official rule of the league. Therefore, if at least two landmarks are visible to the agent, the position of landmarks which are out of sight can be identified. Lighter circles in Fig. 3 are visible, while black circles are out of sight.

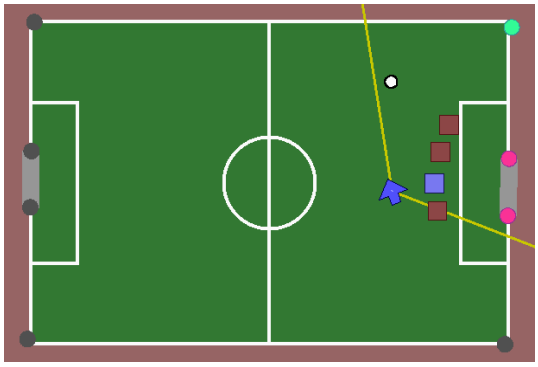

Fig. 3. Visible object in the soccer field

4. Pose of the Agent

The debugger can also draw the pose of the agent and the positional relation between agent's feet and the ball as shown in Figs. 4 and 5. In Fig. 4, the left image shows each joint position from the frontal view and the right sagittal view. In both images the pose of the agent is shown by nodes and edges. The top circle indicates the head, the under circles indicate neck joint and torso. Square nodes indicate leg, heel, and toe joints. In the left images, the right and left circles means arm joints. These figures are drawn based on the message from the agent that contains the joint angles of each time step. Shaded area is the eyesight of the agent. In the left image of Fig. 4, the horizontal axis and the soccer field are assumed parallel, and in the right image of Fig. 4, the vertical axis and the soccer field are orthogonal. The developer of the agent checks the pose of the agent from the two images in Fig. 4.

The positional relation between agent's feet and the ball is shown in Fig. 5. Figure 5 is the mapping of each joint position to the horizontal plane. The white circle indicates the ball, the black circles indicate the nodes of the agent; in this figure they represent the upper body of the agent, and the black squares indicate the lower body of the agent. When the ball is in the eyesight of the agent but is too far to draw in the display, a white triangle locator appears to indicate the direction toward the ball.

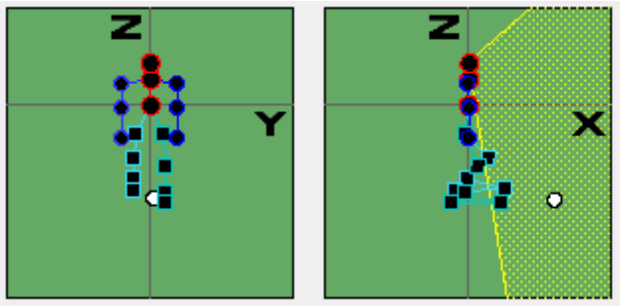

Fig. 4. Pose of the agent

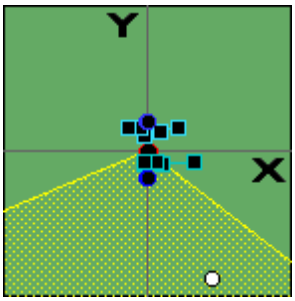

Fig. 5. Position of feet and ball

5. Replay Function

The debugger can replay the game without running the soccer server by loading a text file which includes all debug messages sent by the agent during a game. We call this text file "debug message file" in this paper. The main difference between a debug message file and a log file is that the debug message file contains the internal information from the viewpoint of an agent while a log file just records the joint angles each time step. While replaying a game from a debug message file, extra operations such as pause, go-to-next-step, or go-to-previous-step are available. By using these functions, we can examine an agent's behavior in detail during a game. These functions are also available while a soccer match is played.

6. Manually-Operable Agent

The behavior of an agent sometimes requires the interaction with other agents. Passing the ball to a teammate and avoiding opponent agents that are approaching the agent are examples of such behaviors. Even though there are released binaries of those teams who participated in previous competitions, it is hard to examine those behavior using the released binaries because soccer agents are autonomously moving and never manually controlled. To overcome this problem, it is helpful to prepare an agent that can be manually controlled. For this purpose, we have developed a manually-operable soccer agent. To control the agent, we use a USB Game-pad with 12 buttons and 2 sticks. Figure 6 shows the game-pad for manually operating the agent.

Fig. 6. Game-pad

IV. CONCLUSION

In this paper, we introduced a visual debugger that helps develop a soccer agent of the RoboCup soccer 3D simulation league. We also showed a manually-operable agent by game-pad for debugging. Using the debugger, the visible object on the soccer field and agent's internal state are graphically shown. The debugger can also replay the game on itself by loading a debug message file so that we can check agent's behavior in detail. The source code of the visual debugger will be available in the near future. We hope the debugger makes the agent development more efficient for all teams of soccer simulation 3D league.

REFERENCES

[1] Stone P (2000), Layered learning in Multiagent Systems: A Winning Approach to Robotic Soccer. MIT Press

[2] Reis L P, Lau N, Oliveira E C (2001), Situation Based Strategic Positioning for Coordinating a Team of Homogeneous Agents. Balancing Reactivity and Social Deliberation in Multi-Agent Systems, LNCS 2103:175–197

[3] Grasemann U, Stronger D, Stone P (2007), A Neural Network-Based Approach to Robot Motion Control. RoboCup 2007: Robot Soccer World Cup XI:480–487

[4] Luke S, Hohn C, Farris J, et al. (1997), Co-Evolving Soccer Softbot Team Coordination with Genetic Programming. RoboCup 1997: Robot Soccer World Cup I:398–411

[5] Nakashima T, Udo M, Ishibuchi H (2003), A Fuzzy Reinforcement Learning for a Ball Interception Problem. RoboCup 2003: Robot Soccer World Cup VII:559–567

[6] Kogler M, Obst O (2003), Simulation League: The Next Generation. RoboCup 2003: Robot Soccer World Cup VII: 458–469

[7] Spark - A generic physical simulator, Sourceforge project, http://sourceforge.net/projects/simspark/files

[8] rc-tools, released web page, http://rctools.sourceforge.jp/pukiwiki/

[9] SoccerScope2003, released web page, http://ne.cs.uec.ac.jp/~newone/SoccerScope2003

[10] Nao, Aldebaran web page, http://www.aldebaran-robotics.com