Caspian 3D team description

Bardia Aghabeigi, Mohammad Jafar Abdi, Farbod H. Manouchehri, Pouyan Ziafati, Araz Mashhadi Alipour, Mohammad Reza Kangavari

Intelligent Systems Lab, Computer Engineering Department
Iran University of Science and Technology, Tehran, Iran
{bardia, mohammad, farbod, pouyan, araz, Kangavari}@caspian.iust.ac.ir
http://caspian.iust.ac.ir

Abstract. In this paper we introduced our soccer 3D simulation team structure, main ideas and fundamentals used to support them. We have used fuzzy behavior based approach and reinforcement learning methods as well as developing a trainer tool to visually control and train our multi-agent team. Also we have described our design patterns and multi-layer architecture used in our system.

1 Introduction

Simulated environments are a commonly used method for researching artificial intelligence methods in physical multi-agent systems. Simulations are especially useful for two different types of problems: (1) to experiment with different sensors, actuators or morphologies of agents and (2) to study team behavior with a set of given agents. Additionally, the connection between both types of problems is an interesting research problem [1].

According to our past three years of experience in developing a team of intelligent agents, we have come to a customized methodology and approach to construct an extensible, reusable and robust framework. In this paper we describe this framework and also our academic research areas.

In *focusing areas* section we describe how the AI methods help us solve our problems. In the *system design* part we introduce how we use a component-based system in a multilayered architecture. Also we introduce our developed tool to both test and train our agents.

2 Focusing areas

In this section two types of methods will be described. One is fuzzy behavior-based approach which is used in decision making process. The other one is reinforcement learning method that is used in skill development. To achieve these high level goals, a trainer tool will be developed to both train the agents and also track their behaviors.

2.1 Decision-making

We have used a *behavior-based* approach to control the agents. In our approach, we also employed the power of fuzzy logic. The use of fuzzy logic has two advantages: (i) the ability to express partial and concurrent activations of behaviors; and (ii) the smooth transitions between behaviors [2]. In fuzzy behavior-based control, each behavior is synthesized by a rule base controlled by an inference engine to produce a multivalued output [3]. The implementation of complex behavior generation for artificial systems can be extremely difficult due to the very high level complexity involved. This problem can be partially overcome by decomposing the global tasks into simpler well specified behaviors which are easier to design and tuned independently of each other [4]. Each of these behaviors can be seen as a fuzzy controller, mapping the states as inputs to the control values as outputs. At the higher level, a planner generates a global plan for the agent. Each plan consists of several *meta-rules* or *context rules* that together direct the agent to achieve its goal. Each meta-rule consists of a context and a behavior in the form of:

IF *context* THEN *behavior*,

meaning that *behavior* should be activated with a strength given by the truth value of *context*, a formula in fuzzy logic. When more than one behavior is activated, their outputs will have to be fused. Behaviors that compete for controlling the agent must be coordinated to resolve potential conflicts. Fuzzy behavior coordination is performed by combining the fuzzy outputs of the behaviors using an appropriate operator. The defuzzification is used to select a final crisp action ultimately used for control [3]. Context rules allow us to express different patterns of behavior combination. The combined use of fuzzy behaviors, fuzzy context rules, and fuzzy command fusion provides an extremely flexible solution to the behavior coordination problem [5]. For example in the 3D simulation, the agents have four control values to control their effectors. Power for drive, direction for drive, power for kick, and angle for kick are four control values of an agent in the current version of server (In the future, the power and direction for drive may be changed to power of left and right motors). Each behavior in the lowest level generates a fuzzy set for each of these control variables. Then the fuzzy sets of all behaviors are combined (by means of an appropriate fuzzy operator) to produce a fuzzy set for each control variable. Then defuzzification is performed and a crisp value for each control variable is obtained and sent to the effectors. In this way, the agent can consider different behaviors in parallel, and apply the best control values to its effectors. This modular fuzzy control scheme can speed

up the development process of the agent, because each behavior can be seen as a fuzzy controller and implemented independently of the other behaviors. To design each fuzzy controller we can combine the fuzzy logic with learning algorithms and structures. For example we can use neuro-fuzzy systems such as *ANFIS* to design the fuzzy controller that has the capability to learn its rules and tune its membership functions.

2.2 Least-error, adaptive skills

According to changes in the environment, like server versions or the noise, we have decided to use learning methods to reach a stable state. The description of our method is as follow: a skill is a sequence of basic commands (such as drive or kick) that transform the current situation to a new situation in next steps. The result situation is an element of terminal states set (s^f). The skill ends if either a terminal state is reached ($s^t \in s^f$), or if the time exceeds a certain limit. Since each skill has a clearly defined goal, the task is now to find a sequence of basic commands that does the job. This can be done either by formal programming or by reinforcement learning methods. The main idea of RL is that the agent should incrementally improve its decision policy such that the learning goal is fulfilled better and better. In this case we use Real-Time Dynamic programming methods [6] that solve the learning problem by approximating an optimal value function by repeated control episodes. Since the state space is continuous, a feedforward neural network is used to approximate the value function [7].

For example in intercept-ball skill, our rewarding system is as follow: we consider two types of rewards to the learning agent. One is task reward , r^t , that is given when the learning agent can successfully intercept the passed ball, since the task reward is rarely given to the agent we use an intermediate reward , r^i , that is given to the learning agent when it can reduce the distance between itself and the passed ball.

3. System design

According to our previous experience, we consider our team effort be in a robust framework to support feature variations and extensions and also a rapid approach to code and test components. For this we have our *software architecture* and *development tools.*

3.1 Software architecture

The architecture of the system is the place to show how we are thinking of a complete human-like agent simulation. Keeping in mind how a real soccer player interact with

his world using the SR (Sense Response) scheme, we model a sense-think-act scenario with the packages and layers of architecture as follow:

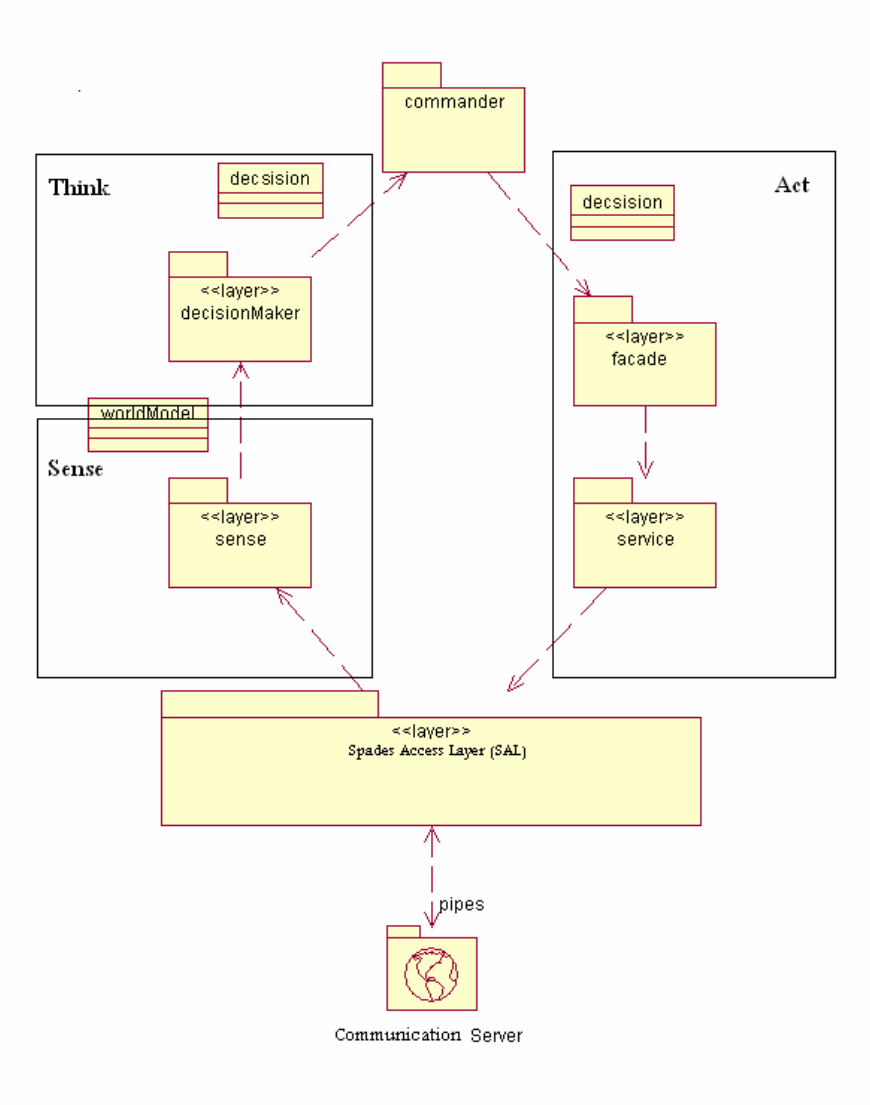

Fig. 1. Layers of architecture making a full-featured spades enabled soccer player agent

This is a two-way system; one for setting up the actuators (act package) and another for gathering raw data into the system and making decisions according those information (the sense and think packages).

In the *act* package does the desired service for the agent using the *façade* design pattern. The agent skills reside in the *service* layer with the emphasis on on-demand adaptations, that is easily change the behavior of a player when new versions of server released. The team strategy is implemented in the *façade* layer; the main idea is to picking and assembling specific subset of *services* (skills).

The other important aspect of the system is understanding the state of the play, tracked with a state-machine, and making appropriate decisions (These decisions are those which are fed to the actuating part.). The *sense* and *think* packages together do this fundamental task.

3.2 Development tool

Besides using defined team coding rules and standards, a time-saving and high-performance utility, named *agent trainer*, is used to both test the just-coded components of the system and train the agents. The main parts of the *agent trainer* are: a monitor to watch out how the agent is thinking and a joystick-enabled input to force and move playing objects in the simulation.

References

1. Oliver Obst and Markus Rollmann. SPARK -- A Generic Simulator for Physical Multiagent Simulations. In Gabriela Lindemann, Jörg Denzinger, Ingo J. Timm, and Rainer Unland, editors, Multiagent System Technologies -- Proceedings of the MATES 2004 , pp. 243–257, Lecture Notes in Artificial Intelligence 3187, Springer, September 2004
2. A. Saffiotti. Fuzzy Logic in Autonomous Robotics: behavior coordination. Proc. of the 6th IEEE Intl. Conf. on Fuzzy Systems, pp. 573-578. Barcelona, Spain, 1997
3. Paolo Pirjanian and Maja Mataric, "A decision-theoretic approach to fuzzy behavior coordination", IEEE International Symposium on Computational Intelligence in Robotics and Automation. November 1999
4. E. Aguirre and A. González, A study of the influence of the combination method in fuzzy behavior coordination. Septiembre, 1998
5. Paolo Pirjanian and Maja Mataric, "Multiple Objective vs. Fuzzy Behavior Coordination", Lecture Notes in Computer Science on Fuzzy Logic Techniques for Autonomous Vehicle Navigation, eds. D. Drainkov and A. Saffiotti. 1999
6. A.G.Barto,S.J.Bradtke,and S.P.Singh. Learning to act real-time dynamic programming. Artificial Intelligence, (72):81 138, 1995
7. Martin Riedmiller:"Concepts and Facilities of a neural reinforcement learning control architecture for technical process control "Neural Computation and Application Journal (1999)8:323-338, Springer Verlag London