The NeverMore 3D Soccer Simulation Team Description 2011

Yaolong Huang, Chao Ma, Haobo Ma, Yalong Yang, Yang Yu,
Jun Song and Jianrong Wang

Computer Science Department
Software Engineering Department
Tianjin University, Tianjin, China
http://cs.tju.edu.cn

Abstract. In this paper we decribe the main features and new ideas incorporated in *NeverMore* 3D soccer simulation team, which will participate in RoboCup 2011 competition in Istanbul, Turkey. In the design of *NeverMore* agent, we layered the agent into two levels hierarchically. We proposed a new concept *'KeyFrame'* in behavior generation. An HTN-based planner of our agent was also implemented as the strategy level, in which we used logic programming language PROLOG.

1 Introduction

This paper describes the main ideas and the implementation of NeverMore 3D soccer simulation team, which will participate in RoboCup 2011 competition. As one league in RoboCup competition, RoboCup 3D Simulation League has been proved to be an excellent platform for DAI and intelligent robotics research since 2003. Compared with the other leagues, programming humanoid agents in simulation with simspark[1] brings several advantages, such as making simplified assumptions about the world, low debugging costs, and the ability to automate experiments, especially for beginners.

NeverMore is the second 3D soccer simulation team from Tianjin University, but it is totally different from its predecessor TJUnited. The development of NeverMore 3D began in November, 2010. In our original design, we divided our soccer agent into two levels hierarchically, *Strategy Level* as the higher one and *Operation Level* as the lower one.

In the Operation Level, we mainly implement the Cerebellum, which executes agent's behaviors, and the World Model, which maintains the world states. There are three atomic behaviors available in Cerebellum, including walking, turning and kicking. All of these behaviors are designed based on Kinematics and adjusted carefully using database. We propose "Data Processor" as an interface to all other parts of agent to pick up more complex information from World Model. This level is written in C++.

The strategy level, which is also called "Cerebrum", is based on an HTN(Hieratical Task Network)[2] planner. We write this part in PROLOG, and using the interface of SWI-Prolog to C++ to integrate it into the agent. The thinking result of

Cerebrum is a predicate string such as `beam(5.5 7.3, 90)` which will be passed to a simple parser written in C++. This parser will translate the predicate string into an available behavior then pass it to the Cerebellum.

The detailed concepts of NeverMore design are presented in the following sections. Section 2 introduces our overall agent architecture and the specific data processors in World Model. Section 3 presents our "KeyFrame" concept. Section 4 illustrates the design of our HTN planner, followed by conclusion and future work in Section 5.

2 Agent Architecture

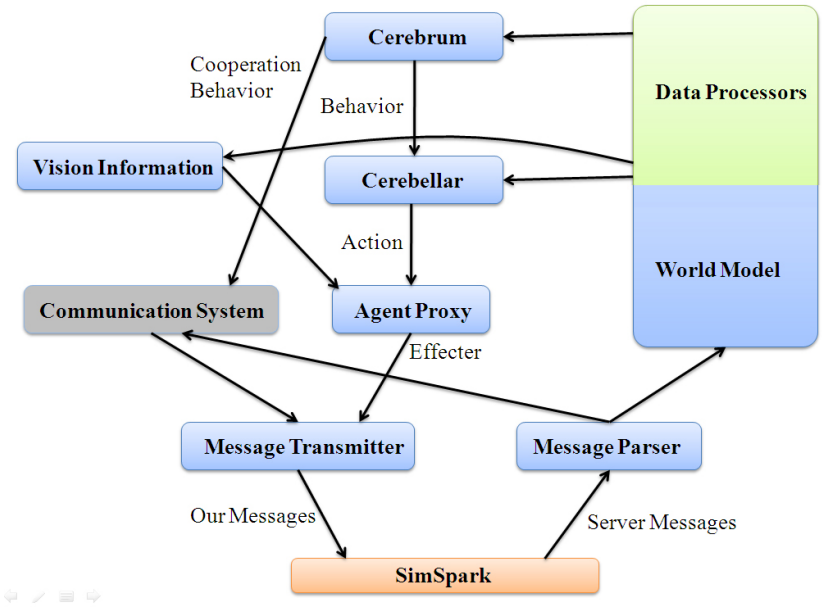

Fig. 1. Interaction between components in NeverMore agent

Based on the idea of Vorst's Layered architecture[4], the NeverMore agent uses a more flexible architecture, as demonstrated in figure 1. Each component is explained as follow:

Message Parser parser that transforms the string representation of the sense message into the internal representation.

World Model and Data Processor Memory for maintaining the internal representation of the environment and extracting useful high level abstraction from this information.

Vision Information Requisition a specific module to control the vision system. Because vision is the most complex and significant information for agent state description and localization, we make this part independent from the thinking process.

Comunication System component used to send and receive the "sentences" to realize the corporation behaviors among several agents, e.g. direct passing.

Cerebellum the execution module of the agent. Cerebellum control the joints of the agent, and receives the behavior from the Cerebrum and execute it with monitoring mechanism.

Cerebrum the thinking component of the agent. It thinks over the next best behavior for the whole team given the world state.

Agent Proxy Joint command from Cerebellum is still a rather high level unit. Agent Proxy takes the command and transform it into the atomic form understandable by server.

Message Transmitter The inverse of Message Parser. It converts the internal form of the agent's command into the string which will be sent to server later.

3 World Model and Data Processor

World Model stores the state information of the environment. In the NeverMore agent, primitive data received from the server is separated from the high level data calculated from the primitive data. Instead of storing data in a Object-Oriented fashion, Data Processor, a view pattern similar to the database system, is adopted.

Data Processor is a mechanism designed for processing the primitive data from the World Model. Each data processor focus on one aspect of analysis of the world state, e.g. positions of the objects and falling state of the agent. In this design, each data processing task can be encapsulated in one place, and adding new aspects is easy compared with the traditional approach, thus the flexibility of the agent is improved.

4 Cerebellum

Cerebellum is the core module of motion generation and management. Its main function is to translate the behavior received from the Cerebrum into an array of actions that can be accepted by the server. In addition, Cerebellum uses the information stored in the World Model to monitor the execution of the behavior.

After receiving a behavior from the Cerebrum, given the current agent state, Cerebellum map the behavior to a sequence of activities each of which consists of a sequence of key frame. With the activities and the current agent state(current frame), the Frame Manager, which will be explained later, finds a proper target key frame, and calculates the transition from the current frame to the target key frame.

4.1 Frame Manager

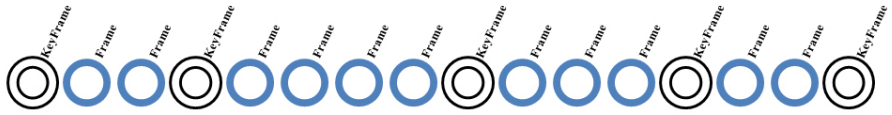

Fig. 2. Frames and Key Frames

Frame Manager takes care the transition from the current frame, defined as follow, to the target frame.

Frame a kind of abstraction of the agent's current state. A behavior can be represented by an array of frames, in which there are some significant ones that are keys to success of execution of the behavior(figure 2), e.g. the turning point of a joint. These significant frames are defined as *Key Frames*. The frames between Key Frames can be computed by fitting some known functions such as linear functions or trigonometric functions.

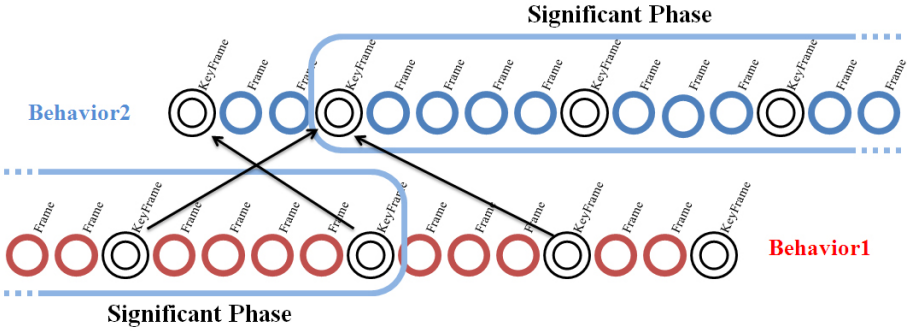

Fig. 3. Transition from Behavior 1 to Behavior 2

An implementation of a behavior can be described as a series of actions that make the agent reach each key frame and at last arrive at the target frame we expected. This action series is also called *Activity*. With key frame system, we can generate or debug the behavior more easily. We do not need to concern about

the total process of the behavior but only the states between Key Frames instead. Since the key frames cut the behavior into short segments, it will be easy for us to find a function to fit angle change of each joint within the specified period. In addition, it can help us to link the key frames of two behaviors to realize a smooth transition from one behavior to the next one without interruption.

Figure 3 shows how the Frame Manager implements the transition between two behaviors. Frame Manager searches all Key Frames from the current behavior to the next behavior with the shortest path. We have built a function to calculate the difference between two Frames. For each behavior, there is a *significant phase*. The transition from the current behavior to the next behavior should be finished before the significant phase of the next one, because it would be meaningless if we just reach the tail of a behavior by transition, which may cause the agent to do some unexpected things.

5 HTN-based Cerebrum

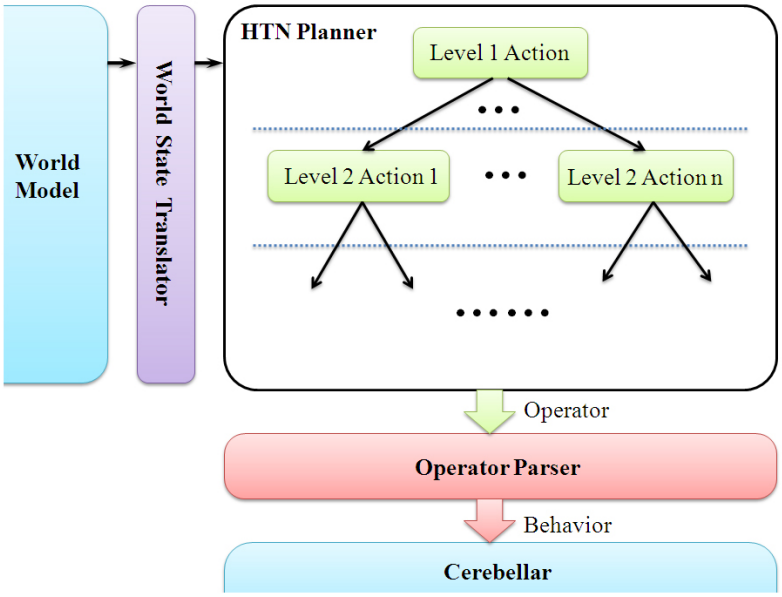

Fig. 4. The components contained in the cerebrum

The Cerebrum is responsible for planning the action sequence which will be executed by the agent. It consists of a PROLOG planner, a operator parser and a world state translator. The planner will be invoked every planning *step*, which

we define as the interval between the beginning of the execution of the current behavior and the next behavior. As illustrated in figure 4, at the beginning of each step, the PROLOG planner use the environmental information provided by world state translator to think over an operator which corresponds to a behavior in Cerebellum, then it pass the operator to the operator parser to convert into a behavior.

In order to rapidly incorporate our strategic knowledge about soccer game into the agent, we decide to use an HTN planner instead of traditional STRIPS-based[2] planner. HTN is more suitable for the problem domain where top-down approach is used, such as robot soccer. In our HTN planner, there are four levels of actions, including Team Level, Group Level, Single Agent Level and Operator Level. Given the current world states, the actions in the higher level will be decomposed into a sequence of actions in the next level by lazy evaluation[3]. The planner assumes the actions of lowest level, which we call operator, can be directly achieved by operation level.

As mentioned above, the HTN planner is implemented in SWI-PROLOG[5]. By using PROLOG, which is a scripting language, the development and debugging time greatly decreases. In addition, because PROLOG is a logic programming language, we can model the game strategy in a logical way, which can be directly feed into the planner.

6 Conclusion and Future Work

In this paper, we addressed some key concepts in NeverMore agent design and explained some of its detailed implementations. Currently, although we have finished the key parts of the agent, many improvements still need to be done in the future. The follow-up of NeverMore agent development will mainly focus on the following aspects:

- More efficient and stable skills for our agent should be designed, e.g. walking with higher speed. With the help of Key Frame System, machine learning techniques can be applied to motion optimization.
- More knowledge and rules should be integrated to the agents Cerebrum so that more complete and powerful strategies can be constructed from the thinking process.
- Consideration of indeterminism of behavior execution should be added to the HTN planner so that we can handle some unexpected effects in the game.
- The communication module should be implemented to realize the synchronization between agents' state.

References

1. Oliver Obst and Markus Rollmann: Spark - A generic simulator for physical multi-agent simulations. Technical report, Universit t Koblenz-Landau (2004)

2. Stuart Russell and Peter Norvig: Articial Intelligence: A Modern Approach. Prentice Hall (1995)
3. Oliver Obst and Joschka Boedecker: Flexible coordination of multiagent team behavior using HTN planning. In Ansgar Bredenfeld, Adam Jaco, Itsuki Noda, and Yasutake Takahashi, editors, RoboCup, volume 4020 of Lecture Notes in Computer Science, pages 521528. Springer (2005)
4. Philipp Vorst: Readylog agents for the robocup 3d soccer simulation league, Masters thesis, RWTH Aachen (2006)
5. Jan Wielemaker: An overview of the SWI-Prolog programming environment. In Fred Mesnard and Alexander Serebenik, editors, Proceedings of the 13th International Work shop on Logic Programming Environments, pages 116, Heverlee, Belgium (2003)