Little Green BATS Humanoid 3D Simulation Team Description Paper

Sander van Dijk, Martin Klomp, Jeroen Kuijpers, A. Bram Neijt, Matthijs Platje, Mart van de Sanden, Marijn Stollenga, Alle Veenstra, Fons Vermeulen, Gauke Veenstra, and Jelle Prins

University of Groningen, Artificial Intelligence Department, The Netherlands

Abstract. The RoboCup 3D vice world champion team the Little Green BATS uses several AI methods to control and train their virtual soccer agents. A hierarchical behavior based architecture coordinates all skills. The architecture allows planning and parallel and competing behaviors. Several methods for low level movement generation are available to the agent, like setting joint angle trajectories and sinusoidal control. The latter is used to train the most critical skill of all, running, with a Genetic Algorithm.

1 Introduction

Agents that live in complex environment usually face a difficult problem: they need to show high level intelligence to survive, but the only way to directly interact with the world is through very basic senses and actions. They need a form of abstraction on these senses and actions to be able to act and response to difficult situations in real time. This abstraction has to take place at different levels, because there is no clear distinction between low level behavior and high level behavior. It is even hard to tell what kind of behavior is of a higher level than other kinds, so abstraction should be possible in any degree, at any level and over any kind of abstraction that is already made at lower levels.

To account for this abstraction, we use a hierarchical behavior model in which the agent's intelligence is constructed as a tree of behaviors, each controlling other behaviors at lower levels to supply a new abstraction in the agent's senses and actions. The model is focused on construction of the behavioral tree by hand using human high level knowledge, either top down or bottom up, with the ability to improve the behavior through learning.

In the next section we describe the behavior model. Section 3 will cover the movement methods used by our agents.

2 BATS Hierarchical Behavior Model

Most RoboCup team use a form of layered behavior models, with low level skills like walking, kicking and dribbling defined separately from higher level behaviors

that can use these skills. For our agents we have developed a hierarchical architecture that makes it possible to break the problem up into smaller subproblems at any level in the agent's behavior. The hierarchical behavior model is based on the following key ideas, which are partly common to other hierarchical behavior systems like [4] and [1]:

- A type of behavior is defined by the type of goals it tries to achieve
- A behavior can set subgoals to be achieved serially by other behaviors
- An agent has one highest level goal (e.g. 'win the game' for a soccer agent)
- Behavior selection is done based on a behavior's capability to reach a goal

Therefore, a behavior consists of a sequence of steps. Each step has a subgoal and a set of sub behaviors that can be chosen to achieve these goals. Figure 1 shows a schematic view of such a behavior. Connecting behaviors together like this results in a directed acyclic graph of behaviors with the top level, most abstract behavior at the root and the lowest level, most primitive behaviors at the leafs. The latter behaviors don't perform any behavior selection anymore, but only perform real world actions. Figure 2 shows part of such a behavior graph for the keeper agent.

At every step of a behaviors sequence, several sub goals could be achieved in parallel. Also, multiple sub behaviors can be defined for a single subgoal. These then compete for selection and the sub behavior with the highest capability of achieving the subgoal wins. The behavior tree is constructed during runtime, based on an XML configuration file. This way the agent's architecture and parameters can be changed thoroughly but quickly, without having to recompile the agent's source code.

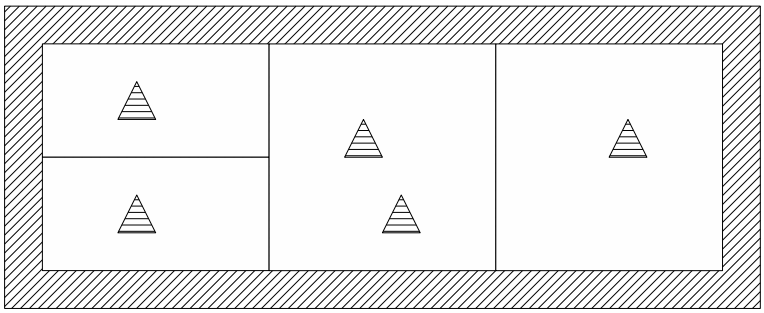

Fig. 1. An example behavior with 3 sequence steps, 2 parallel slots in the first step and 2 competing behaviors in the second slot

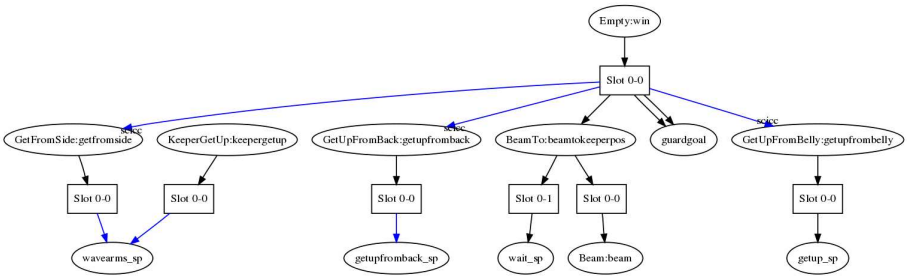

Fig. 2. Part of the behavior structure for the keeper. Blue arrows denote behaviors that are committed to their goal and block until the goal is achieved or no longer achievable. The 'scicc' attribute means that a behavior should commit too if one of his child behaviors is committed, to prevent canceling the behavior at higher levels. Note that behaviors can be placed under different superbehaviors, e.g. wavearms_sp. The BeamTo behavior shows use of a 2 step sequence.

3 Movement Control

The RoboCup 3D humanoid agent is controlled by setting motor joint velocities. The BATS agents have several methods for describing motion at a higher level. These descriptions are translated to joint angles, which are achieved by setting the joint angular velocities as follows:

$$v_i(t) = \gamma(\alpha'_i(t) - \alpha_i(t)) \ , \tag{1}$$

where $v_i(t)$ is the angular velocity for joint i at time t, γ a gain parameter and α' is a goal angle.

3.1 Joint Angle Trajectories

The BATS agents use a very simple scripting language to define trajectories for their joints. The agent plays the script from begin to end, achieving the joint angles set in a line of script using (1) before going to the next line. It is possible to set a certain time to wait before going to the next line and to set the maximum velocity per joint.

This method of movement control is used to create non-cyclical behaviors like standing up and kicking.

3.2 Sinusoidal Joint Control

A lot of human-like movement is cyclical, most notably walking. The BATS agents generate this kind of behavior using a sinusoidal pattern generator. This generator controls joint angles using the following equation:

$$\alpha'_i(t) = \sum_{j=1}^{N} A_j sin(\omega_j t + \theta_j) + C_j \ . \tag{2}$$

For some of the behaviors, like the small, high frequent stepping behavior to position the agent near the ball, the parameters are set by hand. They all are based on the same sinusoidal step-in-place behavior, with an extra out-of-phase sinusoid to create forwards/backwards/sidewards stepping or turning behavior.

The running behavior on the other hand is tweaked using a Genetic Algorithm. The genotype consists of the parameters $A_j, \omega_j, \theta_j, C_j$ with $N = 2$, for 6 joints: the hip, knee and ankle X-axis joints (i.e. LEG2, LEG4 and LEG5) of each leg. Each generation consists of 200 individuals, the first generation is initialized randomly. Tournament selection with a tournament size of 2 is used to select the next generation, after which small mutations are applied to the genotype. Good individuals are found within a few hundred generations.

The running gait found this way was one of the fastest, if not *the* fasted gait in the 3D simulation league in Atlanta, 2007 and was the main cause of the BATS winning the Latin American Open 3D simulation league, 2007.

4 Planned improvements

4.1 Walking speed

Last year, high walking speed proved to be a major winning factor during 3D simulation matches. Teams that could walk fast could prevent the other team from handling the ball effectively. This trend will probably continue, so we will improve the speed of our agents in several ways. First of all we will try to evolve an even faster gait. Also, we are improving the transition between behaviors, so the agent will be able to react faster.

4.2 Stability

The motion controllers discussed in this paper are open loop systems. No sensor data or stability measures are used to control the agent's gait. Although the gaits used have proven to be quite robust, they still have restrictions due to this. Most notably, the transitions between behaviors have to be slow and smooth to prevent unstable postures that the agent can not handle. Also, the agent is unable to react to obstacles it might encounter. We will experiment with different ways to improve the stability of our agents, taking into account several stability measures for humanoid robots [2].

One way to go is to extend our sinusoidal pattern generator, by making the parameters discussed earlier no longer fixed, but adaptive to the current posture and stability of the agent. This way he could adjust the phase of his gait to get back into the right rhythm, or reduce the gait's amplitudes to make his steps more careful. Next to this extension we will also look at total different central pattern generators that use feedback to control the gait, for instance [3].

4.3 Strategy

Like most 3D simulation teams attending the RoboCup 2007 World Championships in Atlanta, our agents used a simple 'run-aim-fire' strategy. We will use our highly modular and expandable architecture to improve this strategy. We expect that teamwork will become more important again in the 3D simulation league, so we will also focus on communication and cooperation.

References

1. Marc S. Atkin, Gary W. King, David L. Westbrook, Brent Heeringa, and Paul R. Cohen. Hierarchical agent control: a framework for defining agent behavior. In *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*, pages 425–432, New York, NY, USA, 2001. ACM.
2. Ambarish Goswami and Vinutha Kallem. Rate of change of angular momentum and balance maintenance of biped robots. *International Conference on Robotics and Automation*, 2004.
3. A.J. Ijspeert. A connectionist central pattern generator for the aquatic and terrestrial gaits of a simulated salamander. *Biological Cybernetics*, 84(5):331–348, 2001.
4. Martin Ltzsch, Joscha Bach, Hans-Dieter Burkhard, and Matthias Jngel. Designing agent behavior with the extensible agent behavior specification language XABSL. In Daniel Polani, Brett Browning, and Andrea Bonarini, editors, *RoboCup 2003: Robot Soccer World Cup VII*, volume 3020 of *Lecture Notes in Artificial Intelligence*, pages 114–124, Padova, Italy, 2004. Springer.