Little Green BATS Humanoid 3D Simulation Team Description Paper for Singapore Open 2010

Hans Alvez, Bas Hickendorff, Herke van Hoof, Ciarán Lier, Josje van Lunteren, Paul Neculoiu, Jaap Oosterbroek, Ron Snijders, and Marco Wiering

University of Groningen, Artificial Intelligence Department, The Netherlands

Abstract. The 'Little Green BATS' is a RoboCup 3D simulation team that uses 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. Genetic algorithms have been used to train running.

Currently, we are working on improving our soccer agent by adding feedback to the motor skills ('closing the loop') for improved stability, implementing a learning system for automatic behavior selection and parameter setting, and implementing more dynamic limb control.

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 whether a behavior is of a higher level then another. This means that a behavioral architecture should support the abstraction and encapsulation of any kind of behavior, regardless of its level.

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. After that, localization, current work, and future plans will be covered.

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 use 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 [5] and [1]:

- A type of behavior is defined by the type of goals it tries to achieve
- A behavior can set sub goals 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 sub goal 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 any more, 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 sub goal. These then compete for selection and the sub behavior with the highest capability of achieving the sub goal 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.

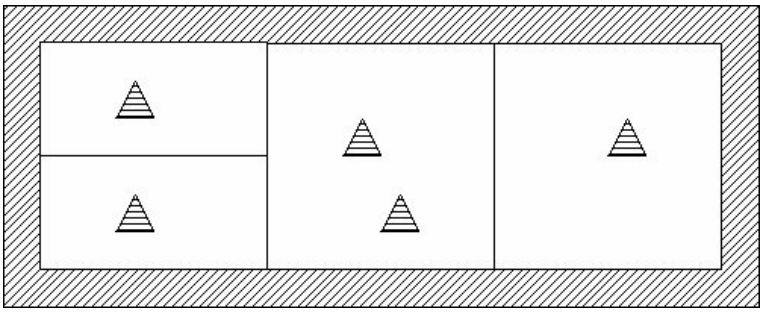

Fig. 1. An example behavior with 3 sequence steps, 2 parallel slots in the first step and 2 competing behaviors in the second slot

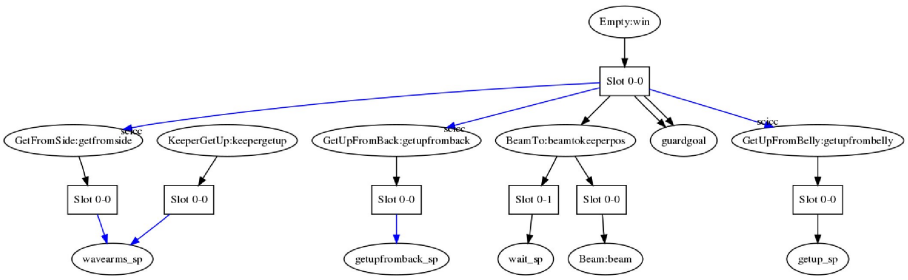

Fig. 2. Part of the behavior structure for the keeper. Blue arrows denote behaviors that are committed to their goal and block until the goal is achieved or no longer achievable. The 'scicc' attribute means that a behavior should commit too if one of his child behaviors is committed, to prevent canceling 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 parameters for the running behavior are set 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. Using this approach, 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 in 2007. For the championships in China we GA to evolve a more stable gait that could also walk in a curve.

4 Localization

One of the major challenges in the competition in Graz, Austria (2009) was the introduction of a limited field of view. Before, the agents were aware of the location of all objects regardless of whether they were in front of them or behind them. Localizing oneself on the field was therefore relatively easy. However, with the introduction of the limited field of view this changed. Localizing oneself and other object became non-trivial, and this troubled many teams, including ours, at the Graz 2009 championship.

In our current implementation, these issues were solved by using the Kalman filter implemented by the Bold Hearts team. This filter is described in [6], and was released by the Bold Hearts team into the public libbats project. This is used by the agent to localize itself and other objects (such as the ball) on the field. Furthermore, it has become necessary to actively move the head to gather information. In our behavior tree, the head of the agent can be directed at an object of interest (the ball) to keep information about this object maximally reliable. It is also possible to explore the world by walking around and moving the head. Depending on the state of the game and on the information currently possesed by the agent, one of these actions is selected.

5 Current Work

Currently we are developing a system improve stability while walking by actively keeping balance. Furthermore, we are working on a learning system that learns the merits of multiple alternatives for the same behavior. Another issue is using inverse kinematics for limb control.

5.1 Learning System

The same goal can be achieved in multiple ways. Because of this, instead of having a single behavior to perform a certain subgoal there might be multiple alternatives available. Furthermore, behaviors might contain a number of parameters to be set. Currently, the selection between alternative behaviors and the setting of parameters is done by hand. This mostly consists of trying out a certain setting and then evaluating this choice by looking at the simulation. The setting that 'looks best' makes it into the behavior tree.

Of course, as the tree grows more complex, this behavior gets more cumbersome. Furthermore, it is quite irreliable as the decision to approve or reject a suggested change to a behavior is sometimes made on just a few matches. To improve the reliability of parameter setting and behavior selection, while reducing the workload for the human observer, we are currently implementing a learning system.

This learning algorithm is modeled after [2], which used a system to estimate the merit of alternative keeper behaviors in the RoboCup mid-sized league. The system we are designing currently will be used just to select the best combination of behaviors and parameters prior to the match. However, as described in [2], an extension of this system enables real-time learning during the match if the chosen strategy appears to be ineffective against the current opponent.

5.2 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 [3].

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 [4].

Another approach under consideration is an active balancing behavior that works in parallel to whatever other behaviors are being selected. The sub-goal to be archieved by this behavior is to move the center of mass of the agent in between its feet. This approach is more modular than the one mentioned above, and fits nicely in the BATS hierarchical agent architecture.

5.3 Limb Control

The system of scripted joint angle trajectories and sinusoidal joint control through fixed central pattern generators has proved to work well in the past. However, sometimes a more dynamic and adaptive approach is desired. For instance, to stop an incoming ball with a hand or foot a fast reaction and dynamic calculation of joint trajectories is needed, not a pre-generated script. In order to do this, we are currently researching different methods for inverse kinematics and limb control. One approach under consideration used neural networks to find the joint angles, given the desired end position and orientation as input.

6 Future Work

In the future, we plan on making improvements to strategy and walking behaviors.

6.1 Walking Speed

In our winning 3D simulation matches, high walking speed was a major winning factor. 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. Thirdly, better stability as described above will hopefully decrease the probability of falling over, thereby making it possible to increase the walking speed.

6.2 Strategy

We expect that teamwork will become more and more important in the 3D simulation league. One of the reasons for this is that is is likely that in the nearby future matches will be played with more players on each side. Because of this, we will also focus on communication and cooperation. A lot of teams are still using a simple 'run-aim-fire' strategy with some simple cooperation between agents. We would like to improve on this by letting the agents by more aware of each other. One of the ways to do this is by using the calling action to transmit information between the agents.

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. T. Van der Zant, M. Wiering, and J. Van Eijck. On-line robot learning using the interval estimation algorithm. In *Proceedings of the 7th European Workshop on Reinforcement Learning*, pages 11–12, 2005.
3. Ambarish Goswami and Vinutha Kallem. Rate of change of angular momentum and balance maintenance of biped robots. *International Conference on Robotics and Automation*, 2004.
4. 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.
5. 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.
6. Sander G. van Dijk and Daniel Polani. Principled construction of perception and action skills for bold hearts 3d 2009. In *RoboCup 2009: Robot Soccer World Cup XI—I. Lecture Notes in Artificial Intelligence*. Springer, 2009.