UT Austin Villa 3D Sim Base Code Release

Patrick MacAlpine and Peter Stone

Department of Computer Science, The University of Texas at Austin, USA
{patmac,pstone}@cs.utexas.edu

1 Introduction

Within the context of the RoboCup 2050 objectives, the 3D simulation league (which models humanoid Nao robots using realistic physics) is arguably the best positioned for the study of robot skill/behavior learning, 11 on 11 strategic planning, and adversarial modeling. Coupled with the fact that it is relatively inexpensive and easy to get ramped up in simulation as opposed to on real robots, one might expect it to be among the most popular RoboCup leagues. However, there remain relatively few participants, in large part because the programs of the top teams in the league represent several years of effort, and are therefore very difficult to compete against. Meanwhile, the simulation league is now actively considering a shift from the custom and idiosyncratic SimSpark simulation environment to the ROS-compatible Gazebo simulator, which has the potential to make it appealing and useful to the many research groups around the world that currently use Gazebo.

With these considerations in mind, we set out to create an open source release with the following properties:

1. Quick and easy learning curve for newcomers to the RoboCup 3D simulation league, so that within a relatively short period of time they can reach competitive levels.
2. Sufficiently constrained out-of-the-box functionality so that there's still need and incentive for creative research by these newcomers, and there remains incentive for league veterans to continue development of their own, different codebases.
3. Sufficient out-of-the-box functionality and ease of use for non-RoboCup researchers to immediately use it for research on behavior learning, 11 on 11 strategic planning, adversarial modeling, and other AI research topics.
4. Compatibility with the new Gazebo simulator so that it remains useful into the future.

The RoboCup 3D simulation league - as well as the whole RoboCup community, and indeed the larger AI research community as well - stands to benefit greatly from a source code release that has the above properties. But to date, there have not been any open source releases that do have all of these properties – no previous code releases within the league have provided support for both behavior learning and Gazebo compatibility. This source code release fills that hole completely.

2 Code Release Overview

The UT Austin Villa team, from the University of Texas at Austin, first began competing in the RoboCup 3D simulation league in 2007. Over the course of nearly a decade the team has built up a strong state of the art code base enabling the team to win the RoboCup 3D simulation league four out of the past five years (2011, 2012, 2014, and 2015) while finishing second in 2013. The UT Austin Villa base code release, written in C++ and hosted on GitHub,[1] is based off of the 2015 UT Austin Villa RoboCup champion agent.

A key consideration when releasing the team's code is what components should and should not be released. A complete full release of the team's code could be detrimental to the RoboCup 3D simulation community if it performs too strongly and causes other teams in the league to completely abandon their own existing code bases in favor of using the UT Austin Villa code release. In order to avoid this scenario certain parts of the team's code have been stripped out. Specifically all high level strategy, some optimized long kicks, and optimized fast walk parameters for the walk engine [1] have been removed from the code release. Despite the removal of these items, which are described in detail in research publications,[2] we believe it should not be too difficult for someone to still use the code release as a base, and develop their own optimized skills (we provide examples of how to do this with the release) and strategy, to produce a competitive team.

The following features are included in the release:

- Omnidirectional walk engine based on a double inverted pendulum model [1]
- A skill description language for specifying parameterized skills/behaviors
- Getup (recovering after having fallen over) behaviors for all agent types
- A couple basic skills for kicking one of which uses inverse kinematics [4]
- Sample demo dribble and kick behaviors for scoring a goal
- World model and particle filter for localization
- Kalman filter for tracking objects
- All necessary parsing code for sending/receiving messages from/to the server
- Code for drawing objects in the RoboViz monitor for visual debugging
- Communication system previously provided for drop-in player challenges
- Example behaviors/tasks for optimizing a kick and forward walk
- Support for Gazebo RoboCup 3D simulation plugin

What is not included in the release:

- The team's complete set of skills such as long kicks and goalie dives
- Optimized parameters for behaviors such as the team's fastest walks (slow and stable walk engine parameters are included, as well as optimized parameters [1] for positioning/dribbling and approaching the ball to kick)
- High level strategy including formations and role assignment

[1] UT Austin Villa code release: `github.com/LARG/utaustinvilla3d`

[2] `www.cs.utexas.edu/~AustinVilla/sim/3dsimulation/publications.html`

3 Agent Architecture

The agent receives sensory information from the environment, including distances and angles to different objects on the field, and the agent uses this information to build a world model. Building a world model requires the robot to be able to localize itself for which we use a particle filter incorporating both landmark and field line observations. Once a world model is built, the agent's control module is invoked. Figure 1 provides a schematic view of the UT Austin Villa agent's control architecture.

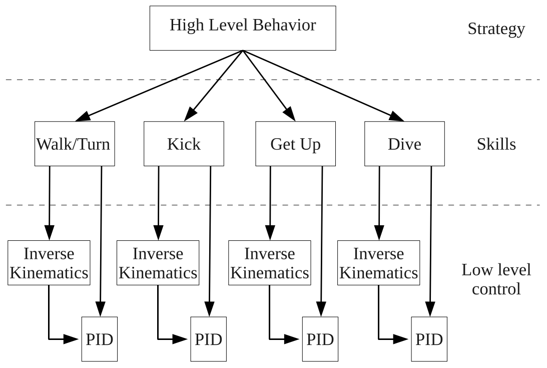

Fig. 1: Schematic view of UT Austin Villa agent control architecture.

At the lowest level, the humanoid is controlled by specifying torques to each of its joints. This is implemented through PID controllers for each joint, which take as input the desired angle of the joint and compute the appropriate torque. Further, the agent uses routines describing inverse kinematics for the arms and legs. Given a target position and pose for the hand or the foot, the inverse kinematics routine uses trigonometry to calculate the angles for the different joints along the arm or the leg to achieve the specified target, if at all possible.

The PID control and inverse kinematics routines are used as primitives to describe the agent's skills. In order to determine the appropriate joint angle sequences for walking and turning, the agent utilizes an omnidirectional walk engine which is described in Section 4.1. Other provided useful skills for the robot are kicking and getting up from a fallen position. These skills are accomplished through a programmed sequence of poses and specified joint angles as discussed in Section 4.2. One of the kicking skills provided in the code release uses inverse kinematics to control the kicking foot such that it follows an appropriate trajectory through the ball as described in [4].

4 Feature Highlights

The following subsections highlight several features of the UT Austin Villa code release. When combined together these features provide a nice platform for machine learning and optimization research.

4.1 Omnidirectional Walk Engine

Agents use a double inverted pendulum omnidirectional walk engine [1] to move. The omnidirectional walk is crucial for allowing the robot to request continuous velocities in the forward, side, and turn directions, permitting it to approach continually changing destinations (often the ball).

The walk engine has parameterized values that control such things as step height, length, and frequency. Walk engine parameters are loaded at runtime from parameter files and can be switched on the fly for different walking tasks (e.g. approaching the ball, sprinting, and dribbling). A slow and stable set of walk engine parameters is included with the release, and these parameters can be optimized to produce a faster walk [1].

4.2 Skill Description Language

The UT Austin Villa agent includes skills for getting up and kicking, each of which is implemented as a periodic state machine with multiple *key frames*, where a key frame is a static pose of fixed joint positions. Key frames are separated by a waiting time that lets the joints reach their target angles. To provide flexibility in designing and parameterizing skills, we designed an intuitive skill description language that facilitates the specification of key frames and the waiting times between them. Below is an illustrative example describing a kick skill.

```
KEYFRAME 1
setTarget JOINT1 $jointvalue1 JOINT2 $jointvalue2 ...
setTarget JOINT3 4.3 JOINT4 52.5
wait 0.08
KEYFRAME 2
...
```

As seen above, joint angle values can either be numbers or be parameterized as `$<varname>`, where `<varname>` is a variable value that can be loaded after being learned. Values for skills and other configurable variables are read in and loaded at runtime from parameter files.

4.3 Optimization Task Infrastructure

A considerable amount of the UT Austin Villa team's efforts in preparing for RoboCup competitions has been in the area of skill optimization and optimizing parameters for walks and kicks. Example agents for optimizing a kick and forward walk are provided with the code release. Optimization agents perform some task (such as kicking a ball) and then determine how well they did at the task (such as how far they kicked the ball) which is known as the agent's *fitness* for the task. Optimization agents are able to adjust the values of parameterized skills at runtime by loading in different parameter files as mentioned in Section 4.2, thus allowing the agents to easily try out and evaluate different sets of parameter values for a skill. After evaluating itself on how well it did at a task, an optimization agent writes its *fitness* for the task to an output file.

Optimization agents can be combined with machine learning algorithms to optimize and tune skill parameters for maximum *fitness* on a task. The UT Austin Villa team uses the CMA-ES policy search algorithm for this purpose. During optimization, agents try out different parameter values from loaded parameter files written by CMA-ES, and then the agents write out their *fitness* values indicating how well they performed with those parameters so that CMA-ES can attempt to adjust the parameters to produce higher *fitness* values. UT Austin Villa utilizes overlapping layered learning [2] paradigms with CMA-ES to optimize skills that work well together.

5 Conclusion

The UT Austin Villa RoboCup 3D simulation team base code release provides a fully functioning agent and good starting point for new teams to the RoboCup 3D simulation league. Additionally the code release offers a foundational platform for conducting research in multiple areas including robotics, multiagent systems, and machine learning. We hope that the code base may both inspire other researchers to join the RoboCup community, as well as facilitate non-RoboCup competition research activities such as a reinforcement learning benchmark keepaway task (a keepaway task is one of the league's challenges at this year's competition).

Recent and ongoing work within the RoboCup community is the development of a plugin[3] for the Gazebo robotics simulator to support agents created for the current RoboCup 3D simulation league simulator (SimSpark). The UT Austin Villa code release provides an agent that can walk in the Gazebo environment. As the development of the plugin continues further support of Gazebo by the UT Austin Villa code release, such as providing getup behaviors, is planned.

A link to the UT Austin Villa 3D simulation code release, as well as additional information about the UT Austin Villa agent, can be found on the UT Austin Villa 3D simulation team's homepage.[4] Further information about the code release can be found in a paper [3] that will be presented orally at this year's RoboCup symposium.

References

1. MacAlpine, P., Barrett, S., Urieli, D., Vu, V., Stone, P.: Design and optimization of an omnidirectional humanoid walk: A winning approach at the RoboCup 2011 3D simulation competition. In: Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI-12) (July 2012)
2. MacAlpine, P., Depinet, M., Stone, P.: UT Austin Villa 2014: RoboCup 3D simulation league champion via overlapping layered learning. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15) (January 2015)
3. MacAlpine, P., Stone, P.: UT Austin Villa RoboCup 3D simulation base code release. In: RoboCup 2016: Robot Soccer World Cup XX. Lecture Notes in Artificial Intelligence, Springer Verlag, Berlin (2016)
4. MacAlpine, P., Urieli, D., Barrett, S., Kalyanakrishnan, S., Barrera, F., Lopez-Mobilia, A., Ştiurcă, N., Vu, V., Stone, P.: UT Austin Villa 2011: A champion agent in the RoboCup 3D soccer simulation competition. In: Proc. of 11th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2012) (June 2012)

[3] `bitbucket.org/osrf/robocup3ds`

[4] `www.cs.utexas.edu/~AustinVilla/sim/3dsimulation/`