UT Austin Villa RoboCup 3D Simulation Base Code Release

Patrick MacAlpine and Peter Stone

Department of Computer Science, The University of Texas at Austin

July 4, 2016

AUSTIN VILLA
ROBOT SOCCER TEAM
THE UNIVERSITY OF TEXAS AT AUSTIN

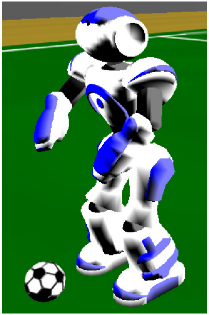
UT Austin Villa 3D Simulation League Team History

- Have competed every year since 2007 except for 2009

- Won RoboCup world championships in 2011, 2012, 2014, 2015, and 2016, second place in 2013

RoboCup 3D Simulation Domain

- Teams of 11 vs 11 simulated autonomous robots play soccer
- Realistic physics using Open Dynamics Engine (ODE)
- Robots modeled after Aldebaran Nao robot (5 robot type variations)
- Robot receives noisy visual information about environment
- Robots can communicate with each other over limited bandwidth channel

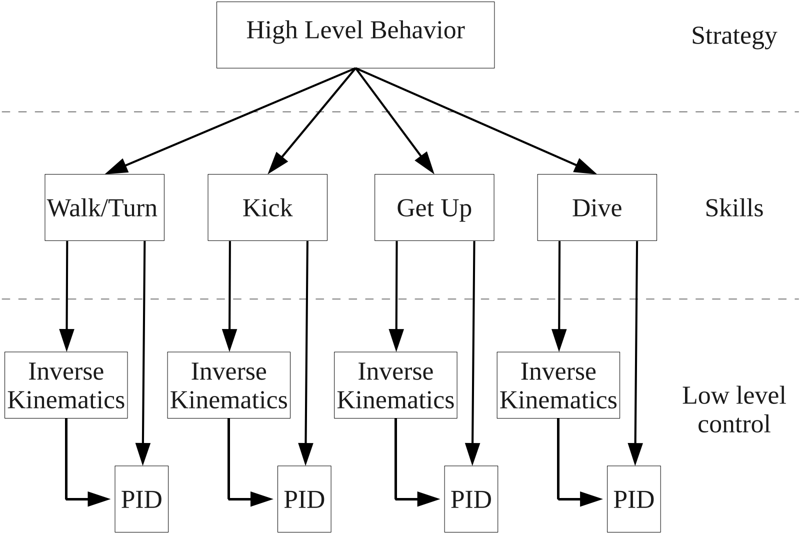


Code Base Release

- Code base almost a decade in development
- Written in C++
- Used in teaching curriculum as part of undergraduate autonomous multiagent systems course
- Released on GitHub

Code Release URL:
https://github.com/LARG/utaustinvilla3d
(Google "utaustinvilla3d")

Agent Architecture



Included in Code Release

- 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 debugging objects in the roboviz monitor
- Communication system previously provided for use in drop-in player challenges
- Omnidirectional walk engine based on a double inverted pendulum model
- A skill description language for specifying parameterized skills/behaviors
- Getup behaviors for all agent types
- A couple basic skills for kicking one of which uses inverse kinematics
- Sample demo dribble and kick behaviors for scoring a goal
- Example behaviors/tasks for optimizing a kick and walk

Included in Code Release

- 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 debugging objects in the roboviz monitor
- Communication system previously provided for use in drop-in player challenges
- Omnidirectional walk engine based on a double inverted pendulum model
- A skill description language for specifying parameterized skills/behaviors
- Getup behaviors for all agent types
- A couple basic skills for kicking one of which uses inverse kinematics
- Sample demo dribble and kick behaviors for scoring a goal
- Example behaviors/tasks for optimizing a kick and walk

Included in Code Release

- 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 debugging objects in the roboviz monitor
- Communication system previously provided for use in drop-in player challenges
- Omnidirectional walk engine based on a double inverted pendulum model
- A skill description language for specifying parameterized skills/behaviors
- Getup behaviors for all agent types
- A couple basic skills for kicking one of which uses inverse kinematics
- Sample demo dribble and kick behaviors for scoring a goal
- Example behaviors/tasks for optimizing a kick and walk

Included in Code Release

- 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 debugging objects in the roboviz monitor
- Communication system previously provided for use in drop-in player challenges
- Omnidirectional walk engine based on a double inverted pendulum model
- A skill description language for specifying parameterized skills/behaviors
- Getup behaviors for all agent types
- A couple basic skills for kicking one of which uses inverse kinematics
- Sample demo dribble and kick behaviors for scoring a goal
- Example behaviors/tasks for optimizing a kick and walk

Included in Code Release

- 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 debugging objects in the roboviz monitor
- Communication system previously provided for use in drop-in player challenges
- Omnidirectional walk engine based on a double inverted pendulum model
- A skill description language for specifying parameterized skills/behaviors
- Getup behaviors for all agent types
- A couple basic skills for kicking one of which uses inverse kinematics
- Sample demo dribble and kick behaviors for scoring a goal
- Example behaviors/tasks for optimizing a kick and walk

Included in Code Release

- 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 debugging objects in the roboviz monitor
- Communication system previously provided for use in drop-in player challenges
- Omnidirectional walk engine based on a double inverted pendulum model
- A skill description language for specifying parameterized skills/behaviors
- Getup behaviors for all agent types
- A couple basic skills for kicking one of which uses inverse kinematics
- Sample demo dribble and kick behaviors for scoring a goal
- Example behaviors/tasks for optimizing a kick and walk

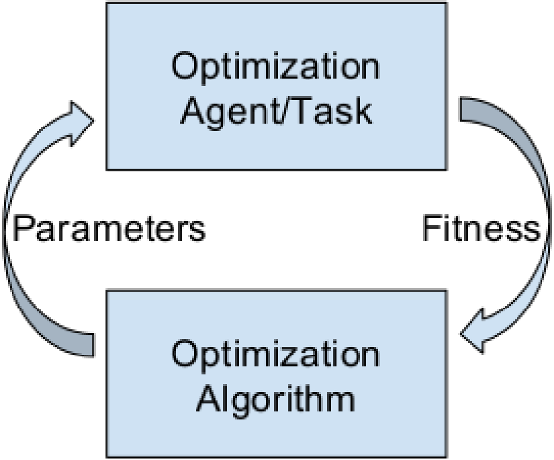
Included in Code Release

- 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 debugging objects in the roboviz monitor
- Communication system previously provided for use in drop-in player challenges
- Omnidirectional walk engine based on a double inverted pendulum model
- A skill description language for specifying parameterized skills/behaviors
- Getup behaviors for all agent types
- A couple basic skills for kicking one of which uses inverse kinematics
- Sample demo dribble and kick behaviors for scoring a goal
- Example behaviors/tasks for optimizing a kick and walk

Initial Walk Parameters

- Omnidirectional walk based on double inverted pendulum model
- Designed and hand-tuned to work on the actual Nao robot
- Provides a slow and stable walk

Optimization Infrastructure



- Optimization algorithm produces candidate parameters to evaluate on optimization task
- Optimization task evaluates parameters and returns fitness to optimization algorithm

Obstacle Course Walk Optimization Task

Red ’T’ = *GoToTarget* parameters, yellow ’S’ = *Sprint* parameters

- Optimizing parameters for omnidirectional walk engine (step height, frequency, balance, etc.)
- Agent rewarded for distance traveled toward magenta target

Kick Optimization Task

- Kick consists of series of joint angle poses specified by a skill description language
- Optimize joint angle values
- Agent rewarded for distance and accuracy

Dribbling and Kicking the Ball

Red 'T' = *GoToTarget* parameters, yellow 'S' = *Sprint* parameters,
cyan 'P' = *Positioning* parameters, orange 'A' = *Approach* parameters

UT Austin Villa 2014: RoboCup 3D Simulation League Champion via Overlapping Layered Learning
Patrick MacAlpine, Mike Depinet, and Peter Stone. AAAI 2015.

Not Included in Code Release

- The team's complete set of skills such as long kicks and goalie dives
- Some optimized parameters for behaviors such as top speed walking
- High level strategy including formations and role assignment

Optimizing Robot Morphologies

- Optimized six leg anchor joint positions, no power or mass changes
- Achieved running speed of ≈ 3 m/s

P. MacAlpine, M. Depinet, J. Liang, and P. Stone. UT Austin Villa: RoboCup 2014 3D Simulation League Competition and Technical Challenge Champions, 2015.

Keepaway

Keepaway team maintains possesion of the ball while also keeping ball inside shrinking red boundary box

Support for Gazebo RoboCup 3D Simulation Plugin

Click to start

Gazebo robot simulator maintained by the Open Software Robotics Foundation

Summary

- Released base code of a champion 3D simulation team written in C++ on GitHub

Summary

- Released base code of a champion 3D simulation team written in C++ on GitHub

- Serves as a resource for current teams and good starting point for new teams

Summary

- Released base code of a champion 3D simulation team written in C++ on GitHub
- Serves as a resource for current teams and good starting point for new teams
- Especially useful for research in machine learning and multiagent systems

Summary

- Released base code of a champion 3D simulation team written in C++ on GitHub
- Serves as a resource for current teams and good starting point for new teams
- Especially useful for research in machine learning and multiagent systems
- Platform for performing research extending outside of RoboCup community

Other 3D Simulation Code Releases

- magmaOffenburg (Java 2014)
- libbats (C++ 2013)
- Nexus (C++ 2011)
- TinMan (.NET 2010)

None provide Gazebo support or optimization task infrastruture

Contributors

- Frank Barrera
- Samuel Barrett
- Yinon Bentor
- Nick Collins
- Mike Depinet
- Josiah Hanna
- Todd Hester
- Shivaram Kalyanakrishnan
- Jason Liang
- Adrian Lopez-Mobilia
- Patrick MacAlpine
- Michael Quinlan
- Art Richards
- Andrew Sharp
- Nicu Stiurca
- Peter Stone
- Daniel Urieli
- Victor Vu

More Information

RoboCup 3D Simulation Homepage:
http://www.cs.utexas.edu/~AustinVilla/sim/3dsimulation/
(Google "UT Austin Villa 3D Simulation")

Demo Behavior