15th ICCRTS
"The Evolution of C2"

Adapting the RoboCup Simulation for Autonomous Vehicle Team Information Fusion and Decision Making Experimentation

Topic(s)
Modeling and Simulation
Experimentation and Analysis
C2 Architectures and Technologies

Douglas S. Lange, SPAWARSYSCEN Pacific
Phillip Verbanscics, University of Central Florida

Point of Contact
Douglas S. Lange
Space and Naval Warfare Systems Center, Pacific
53570 Silvergate Ave Rm 1432
San Diego, CA 92152-5147

(619)553-6534
doug.lange@navy.mil

Abstract

The RoboCup Simulator is a popular experimentation platform for research in artificial intelligence, from areas such as computer vision, machine learning and multi-agent systems. The primary purpose of the simulator is to provide a software analogue to the physical robot RoboCup leagues, providing an easy method of entry into RoboCup without the need for expensive hardware. This simulator provides a common framework for researchers to compare their approaches and many graduate students and professors utilize the simulator for their experiments. We wanted a platform that would encourage researchers outside the government to produce the kinds of algorithms and software that would easily transition into solutions for teams of autonomous vehicles for military scenarios. To accomplish this, we began modifying the RoboCup soccer game step-by-step to incorporate rules that simulate these scenarios. These changes affect the field of play, agent-to-agent communication rules, the number of balls, what constitutes a foul, and the number of teams present in a game. An initial version of the software that incorporates some of these changes is now available with the final version scheduled for completion during the summer of 2010. We anticipate publishing the final version as open source software.

Introduction to RoboCup

Soccer is currently one of the best analogues available for military and other team operations. Individuals within a team must make decisions to ensure team victory without being able to stop and script out plays (with some exceptions). It is a network centric activity, where any player can fill most any role at any time, move wherever they want and whenever they want. Teamwork is based on principles of play, individual assessment of the tactical situation, and communication among players. It's pretty much all there.

"RoboCup™ is an international research and education initiative. Its goal is to foster artificial intelligence and robotics research by providing a standard problem where a wide range of technologies can be examined and integrated" [RoboCup 2002]. Artificial Intelligence (AI) researchers recognize the value of soccer as a platform for experimenting with autonomous systems, cooperative behavior, and self-synchronization. Researchers develop autonomous robot players and compete against each other and benchmarks in order to test their technology. Many graduate students have written their dissertations based on their RoboCup™ results.

For those that want to test algorithms without building entire robots, the RoboCup™ Simulator is a popular experimentation platform. Many researchers in machine learning and autonomous decision making utilize the simulator to evaluate their algorithms. The simulator provides a software analogue to the physical robot RoboCup leagues, providing an easy method of entry into RoboCup™ without the need for expensive hardware.

The Autonomous Vehicle Problem

Soccer is a great platform for experimentation, but the structure of the game isn't quite right for testing out behavior for some military scenarios. Soccer is a symmetric game. Both teams notionally have the

same objectives, similar resources, and tools available to them. The difference is made in the individual skill and the tactics utilized.

Warfare is asymmetric. Two teams, if that is all that there are, typically have different goals. It may be as simple as invader vs. defender. But things get more complicated than that still. Often multiple teams act against another group of teams in alliances. There are also neutral agents in the way.

The goal of this project was to envision a particular class of scenarios of interest and determine what steps would be needed to evolve from the RoboCup™ simulator to a simulator that played a game that would test out concepts for that scenario. We then began to modify the simulator to play the new game.

In our scenario, we are defending a high value unit in a maritime environment using a set of autonomous vehicles against a force of armed vehicles, where there may also be neutral vehicles present.

The Stepwise Plan

One of the games in the simulation is a keep-ball game. A team of agents tries to keep the ball away from another team of agents. Our game starts there. Our view is from the defensive side.

First, we play with multiple balls. We represent hostile-intent by possessing a ball. The attacking team can come at us with more than one weapon and more than one vehicle with hostile intent. This is relevant to the next change as well in that we allow fouling a player who has the ball, but not one that doesn't have the ball. So if our agent can get to and contact a ball carrier, that is a good thing, and reduces the number of opponents. However, we are not allowed to touch a player who does not have the ball (therefore, not armed or lacking hostile intent). The attacking players can pass the balls around, requiring our autonomous vehicles to track not only location but changes in the threat.

In our game we need to have three teams. Our friendly forces are one team, the enemy forces are another, and the third team constitutes neutral players. Neutral players won't get the ball, and our team needs to make sure we don't foul them. Players don't all wear uniforms so when we decide a player is neutral we need to track that information.

The next change is that we are to defend a goal. In the normal soccer simulation, there are two goals, but this is an asymmetrical game. We're protecting a goal representing a high-value unit and get no benefit from attacking another goal. The other team wins by shooting their ball (representing a weapon) hitting our goal.

Now the changes get a bit more complicated. Our field is three-dimensional, and the goal can be hit from any direction. It is in the center of our three dimensional playing field. This field has a plane through it representing the surface of the ocean. Some players are able to move only on the surface, some on the surface or beneath it, and some strictly above the surface. Players can communicate in the same region that they inhabit. Additionally, players above the surface can communicate with those on

the surface plane. Players on the surface can communicate with those beneath the surface plane, but subsurface players cannot communicate with those in the air. The laws of physics as related to the ball are not followed. The ball can move in the three dimensional field just as it would in a two dimensional field.

The contours of the three regions start simple, but ultimately become more complicated. We are dealing with small enough areas so that we can assume a cylinder, but ultimately, the area above the surface is bounded by a curved surface that is a the portion of a sphere that the field intersects. The bottom is as well, essentially sandwiching the field between two spheres.

Finally, the field moves. The field is defined as having a radius along the surface plane extending from the goal, but the goal can have a course and speed. If the high-value unit we are protecting is a ship, this would be the case. If it is an oil platform or other static unit, then it would obviously stand still.

Our effort has been to modify the RoboCup simulation stepwise, preserving the remaining characteristics of the simulation so that algorithms similar to those developed for the original simulation could be developed and tested. Many of those algorithms are relate to information fusion, decision making, and communication, all problems relevant to the military environment and autonomous agents in particular.

RoboCup Modification

Shift to C# Programming Language

To make the code more accessible and cross-platform without recompilation, a shift was made to the C# programming language. With C#, code compiled into an executable can be run in Windows, or, *nix based systems with Mono. Additionally, the cross-language features enable easier accessibility to algorithms implemented in any .NET language.

The shift to the C# language also makes extending the server much easier, as several programming burdens (such as memory management) are eliminated.

Class Changes

A large portion of the code-base was devoted to legacy code from previous server versions, all the way to the original LISP implementation of the RoboCup server. This included such items as communication formats for all previous server versions and formats for logging files for all previous versions. These have been all compressed and brought up to solely the most recent version of RoboCup Server.

Communication has been shifted to an external library and has been changed from a LISP like string based system to an XML format, to better enable cross-platform use. Further, an option has been enabled to switch between a binary encoding or XML encoding of the messages.

Extension to Multiple Teams

Changing to multiple teams extended the existing two individual teams to a list of teams, identifiable by unique team ID (as determined by position in the list) or the unique team name. This created the need for additional parameters for the soccer server. These parameters are: Max_teams (the number of teams possibly allowed), Max_players (the maximum number of player per team), and Hide_teams (removes team names from visual info about players who are not teammates if set to true). The first two teams will be set to left and right sides, while the remaining sides will be set to neutral with respect to side of the field.

The extension to multiple teams allows the concept of multi-faceted game as opposed to one side versus another. For example, the game of keepaway can now be extended to have neutral players on the field, which create obstacles for the other players to avoid.

Extension to Multiple Balls

Similar the changes for multiple teams, the extension to multiple balls changed the situation from a single ball a list of balls which must now be identified by a unique ID. In previous versions of the sever, commands regarding the ball (e.g. kick) would only contain parameters concerning the action itself (i.e. power, angle, etc.), however now the particular ball being acted upon must also be addressed. Actions concerning a ball now contain an additional parameter which defines which ball the action is affecting (e.g. for a kick it is now <power, angle, ballId> instead of <power, angle>). This change is reflected in the change in communication format for the commands.

This extension to multiple balls challenges agents. While previously they did keep track of multiple objects (players), they did not have to consider the implications of acting on different objects and only had to consider the one object of importance, the ball. By changing to multiple balls, agents must now consider which ball to act upon and determine the relative importance of the different balls.

Extending the Server

There are four key areas to consider when extending the server: Referees, Players, the Stadium, and Communications. The following briefly explains these and then briefly outlines a client. For more complete information on the server, see the official RoboCup Soccer Server Manual [RoboCup Simulator].

Referees

Referees represent the primary means of implementing new games and scenarios into the server. The Referee class, as the name implied, enforces rules and structure onto the game. To implement new Referees, one creates a new child class of the base Referee class and implements the appropriate methods. Then, add an instance of the new Referee to the Referee list (done in the Constructor of the Stadium).

Players

The Player class contains all the methods and actions that a player may perform. These actions are activated via received commands and then the player calls the necessary methods from the Stadium to

perform the actions. Additionally, all updates to the player are done through methods within the Player class.

Stadium

The stadium is where all the action takes place. It is responsible for maintaining and advancing the state of the game. The primary methods to look at are those that are extended from the Timetable class. These represent the key steps that the game must go through, primarily dealing with receiving/sending communications, querying the Referees about the game state and advancing the next cycle of the simulation.

Communications

Everything players do or receive in the RoboCup simulator depends on the communications. Messages must be passed back and forth over UDP sockets to remote clients, which represent the agents. Any feature that is added to the server or command that influences the server must be contained in the communications code. This code has been transferred to its own CLS-compliant library, to make it so that extension is much easier. Rather than updating both the server and client code, both can share the communications library. Communication is done through either XML or binary encoding in .NET. It is separated into Monitor and Player sections, where the Monitor is just an observer, while the Player area is for active agents. Additionally, the library is sub-divided into Commands (which come from clients) and Messages (which travel to clients).

Summary

We have begun the modification of RoboCup simulation software to create a game that matches the asymmetric scenarios that we are interested in from the standpoint of autonomous vehicles in a maritime environment. The modifications above discuss show the progress we have made in the stepwise modification plan. The remaining steps are anticipated to be completed during the summer of 2010, after which we intend to publish the result back as open source software and encourage researchers to see how their algorithms work in our new environment.

References

[RoboCup 2002] RoboCup, “RoboCup Brief Introduction”, http://124.146.198.189/Intro.htm, 7 August 2002, accessed on 28 January 2010.

[RoboCup Simulator] RoboCup, Simulator User’s Manual, http://sourceforge.net/apps/mediawiki/sserver/index.php?title=Users_Manual/Overview, accessed on 28 January 2010.