GaTAC: A Scalable and Realistic Testbed for Multiagent Decision Making (Demonstration)

Ekhlas Sonu and Prashant Doshi
Dept. of Computer Science
University of Georgia
Athens, GA,30602, USA
esonu@uga.edu, pdoshi@cs.uga.edu

ABSTRACT

In an attempt to bridge the gap between the theoretical advances in multiagent decision making algorithms and their application in real world scenario, we present the *Georgia testbed for autonomous control of vehicles (GaTAC)*. GaTAC provides a low-cost, open-source and flexible environment for realistically simulating and evaluating policies generated by multi-agent decision making algorithms in real world problem domains pertaining to control of autonomous uninhabbited aerial vehicles (AUAVs). We describe GaTAC in detail and shall demonstrate how GaTAC could be used to simulate an example AUAV problem. We expect GaTAC to facilitate the development and evaluation of scalable decision making algorithms with results that have immediate practical implications.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems

General Terms

Experimentation

Keywords

scalability, testbed, autonomous vehicles

1. INTRODUCTION

With advances in multi-agent sequential decision making algorithms, there is a need to move beyond traditional toy problems to scalable problems with real world implications that may be used to evaluate performance of various decision making algorithms. We think that desired problem domains should, (a) be scalable to naturally allow for greater numbers of physical states, actions, observations, and agents while maintaining the plausibility of the problem; (b) be flexible to accomodate different types of multi-agent settings such as co-operative, competitive or mixed; (c) produce solutions that are rich in structure and which have practical implications; and (d) be realistic and have popular appeal. In this paper, we introduce a problem domain that meets these criteria.

Unmanned agents such as uninhabited aerial vehicles (UAVs) are used in fighting forest fires, law enforcement, and wartime reconnaissance. They operate in environments characterized by multiple parameters that affect their decisions, including other agents with common or antagonistic preference. The task is further complicated as the vehicles may possess noisy sensors and unreliable actuators. In such complex and unreliable settings, an autonomous UAV must choose navigational and surveillance actions that are expected to optimize its objective of say, timely reconnaissance of target while avoiding detection. UAV operation theaters may be populated by a single reconnaissance target or a host of other agents including UAVs working together as a team, or other hostile UAVs. Depending on the type of the agents present in the environment, this would involve application of decision-theoretic frameworks such as interactive POMDPs [2] and decentralized POMDPs [1].

In order to facilitate application of multiagent decision making to the problem domains pertaining to AUAVs and its evaluation, we have developed the *Georgia testbed for autonomous control of vehicles (GaTAC)*. GaTAC is a computer simulation framework for evaluating autonomous control of aerial robotic vehicles such as UAVs. It provides a low-cost and open-source alternative to highly complex and expensive simulation architecture. GaTAC uses a free, open-source and multi-platform flight simulator software called *FlightGear*. GaTAC deploys multiple instances of the flight simulator on a networked cluster of computing platforms using a scalable architecture. It is flexible in allowing the interchange of instances of manually controlled vehicles with autonomous ones. It can be extended to include complex scenarios involving multiple UAVs performing complex tasks.

In this paper we describe GaTAC in detail focusing on its architecture and its components and provide an introduction to our demonstration of its applicability on a simple example problem.

2. TESTBED FOR AUTONOMOUS CONTROL

As we mentioned previously, the objective behind the development of GaTAC is to provide a realistic and scalable testbed for algorithms on multiagent decision making. GaTAC facilitates this by providing an intuitive and easy to deploy architecture that makes use of powerful, open-source software components. Successful demonstrations of algorithms in GaTAC would not only represent tangible gains but also have the potential for practical applications toward designing autonomous UAVs. We think that multiagent decision making could make significant contributions in this area.

2.1 Architecture

We show a simplified design of the GaTAC architecture in Fig. 1, where a manually controlled UAV is interacting with an autonomous one. Briefly, GaTAC employs multiple instances of an open-source flight simulator possibly on different networked platforms that communicate with each other via external servers, and an autonomous control module that interacts with the simulator instances using a communication module. GaTAC can be deployed on most platforms including Linux and Windows with moderate hardware requirements, and the entire source code is available. GaTAC is implemented using C++ programming language.

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

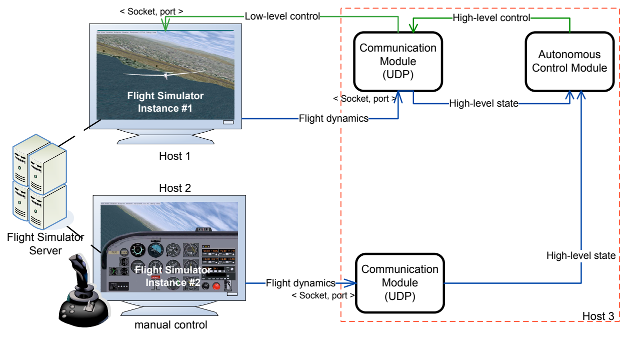

Figure 1: Design of GaTAC showing two networked instances of a flight simulator (FlightGear with 3D scenery from TerraGear), one autonomously and other manually controlled. GaTAC is extensible and more instances may be added.

Each agent may be simulated on a separate instance of GaTAC that may be running on different computing platforms connected through the internet. GaTAC doesn't apply any limit to the problem size or the number of agents. We describe the individual components of a GaTAC instance next.

2.1.1 Flight Simulator

We utilize *FlightGear* [3] as the flight simulator in GaTAC. FlightGear flight simulator project is an open-source, multi-platform, hyperrealistic flight simulator with a goal to develop a low cost sophisticated flight simulator for use in academic and research environments. The entire source code of FlightGear written in C++ is available under GNU General Public License, allowing full extensibility. It provides a flexible platform with options to choose from multiple aircrafts, including UAVs (e.g., Predator), which could be operated manually or guided automatically by external programs. FlightGear uses a generic, six degrees-of-freedom flight dynamics model for simulating the motion of aerial vehicles. It simulates the effect of airflow on different part of the aircraft making it possible to perform the simulation based on geometry and mass information combined with more commonly available performance numbers for an aircraft. FlightGear utilizes realistic 3-dimensional scenery available from TerraGear, which virtually maps many parts of the world including models of the sky.

FlightGear also provides multiple views of the flying aircraft, including external views from different viewpoints and an internal cockpit view which allows for a realistic flying experience. Finally, multiple instances of FlightGear may be run on different hosts and are linked together through external servers located in different countries. This multi-player mode allows for multiple aircrafts to fly simultaneously and see each other if the aircrafts are in visual range. This is a crucial functionality for its use in multiagent systems research.

2.1.2 Communications Module

FlightGear allows remote control of the aircrafts through UDP socket based communication channels. The communication module in GaTAC (see Fig. 1) establishes UDP sockets that are used to communicate with instances of FlightGear. Control data at a low level is sent to FlightGear in order to remotely pilot the UAV. This data includes values for more than 30 flight parameters including the throttle, rudder, elevator and aileron settings. The communications module receives the aircraft's flight dynamics in real time from FlightGear. This includes data about the current latitude, longitude and altitude location of the aircraft, the values of the different flight surfaces, and current fuel level. During flight, the communication module continuously sends and receives data from the FlightGear instance at a pre-specified baud rate. GaTAC associates a communication module with every instance of FlightGear regardless of whether the corresponding aircraft is autonomously or manually controlled. If the aircraft is manually controlled, the communication module simply receives the flight dynamics of the aircraft in order to remain informed about the state of that aircraft. The communication module also provides a way for UAVs to communicate with each other. This may be useful in team settings with communication.

2.1.3 Autonomous Control Module

In order to allow algorithmic control of the aircraft, GaTAC implements an autonomous control module (see Fig. 1). This module implements low-level control actions such as setting values of various flight parameters including throttle, rudder, elevator and aileron settings. Using these low level actions, we have constructed high-level control actions such as *takeoff*, *fly straight*, *change heading*, move to an adjacent grid, etc. We may utilize these actions to construct a set of agent actions for any decision making problem. The GaTAC library is extensible to include additional actions. Additionally, GaTAC allows users to define their own grids of any size.

Because we intend to utilize GaTAC with multiagent decision making frameworks, it implements methods that read *policy tree* files in different formats generated by various algorithms. We have made effort to make GaTAC independent of any particular type of decision-theoretic framework. It may be easily integrated with existing implementations by simply providing it with the behavioral policies generated by the various algorithms for decision making.

3. DEMO

In demo, we show how GaTAC may be used to simulate and evaluate the policies obtained for a few example UAV reconnaissance problem using various decision making algorithms. The evaluation criteria may differ according to various problem. Some of the evaluation criteria may be the number of successful achievement of goal (number of successful reconnaissance), the cumulative reward obtained, etc.

More information on GaTAC including the source code, a demo video and an informative powerpoint presentation may be obtained from the following link: http://thinc.cs.uga.edu/thinclabwiki/index.php/GaTAC_:_Georgia_Testbed_for_Autonomous_Control_of_Vehicles

4. DISCUSSION

GaTAC provides a low cost, open source scalable platform for a satisfactory simulatory experience of a problem domain that has popular appeal, and is extensible. GaTAC represents a realistic testbed for multiagent decision making research, and a first step in our knowledge toward enabling decision-making algorithms to cross over to domains of practical import. We hope that GaTAC could be further improved with inputs from users.

Acknowledgement

This research is supported in part by AFOSR through grant no. #FA9550-08-1-0429 and in part by a NSF CAREER award with grant no. #IIS-0845036.

5. REFERENCES

[1] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of Operations Research*, 27(4):819–840, 2002.

[2] P. Gmytrasiewicz and P. Doshi. A framework for sequential planning in multiagent settings. *Journal of Artificial Intelligence Research*. 24:49–79, 2005.

[3] A. R. Perry. The flightgear flight simulator. In UseLinux, 2004.