Aerial Robotic Simulations for Evaluation of Multi-Agent Planning in GaTAC

(Demonstration)

Kenneth Bogert, Sina Solaimanpour, Prashant Doshi
THINC Lab, Computer Science Department, University of Georgia
{kbogert, sina, pdoshi}@uga.edu

1. INTRODUCTION

Georgia Testbed for Autonomous Control of Vehicles (GaTAC) [7] is a test bed for multi-agent planning under uncertainty. Previously GaTAC simulations have been performed in FlightGear, a flight simulation software, using realistically simulated aircraft. We enhance GaTAC's versatility by enabling simulations with fully autonomous off-the-shelf robots. Scenarios may include multiple robotic drones interacting with each other while flying over a grid of configurable size. Each drone is controlled via a networked client process. Modules are available for simulations with virtual drones in a robotic simulator and autonomous quadcopters as we show in Figs. 1 and 2. Virtual agents allow for scenarios to be constructed and tested quickly and easily while robotic ones allow for higher impact demonstrations and exhibitions. The networked, modular architecture of GaTAC allows a client program to control either type of agent with no additional overhead for the user.

GaTAC agents may be controlled either interactively by a human or with a pre-specified policy. Policies may be generated by an optimal decision-making framework or manually specified as desired. In our demonstration scenarios, we use an interactive POMDP (I-POMDP) [3] framework or DEC-POMDP [1] to find optimal policies for all agents as appropriate.

2. DEMONSTRATION SCENARIOS

To demonstrate GaTAC's multi-agent simulation capability we choose a simple fugitive reconnaissance game. The fugitive is attempting to reach a safe-house in a known location but is uncertain as to its own position and the location of two UAV's searching for it. We divide the state space into a 5 by 5 grid, similar to sectors in a combat theater, in which these three agents operate (see Fig. 3). The two UAV's know their own location but not that of the other agents. All agents can, however, detect each other when located in the same grid cell. The simulation run is over when either the fugitive is detected or it manages to reach the safe-house undetected.

For simplicity, the agents move in the four cardinal directions (North, South, East, West) only, and all agents know the location of the safe-house. An agent can perform a Listen action which will tell it whether another agent is detected one cell away from it (in the cardinal directions), though not the identity of the other agent.

We will demonstrate a robotic simulation of this scenario live for conference attendees using three quadcopter drones. Additionally, conference attendees will be allowed to interact with similar simulations performed and visualized with virtual agents. They can do so by dictating drone actions with a keyboard while the simulation is running or by designing a policy for use by one of the agents. A preview of the demo is at `http://youtu.be/lRDx-2jsXe8`.

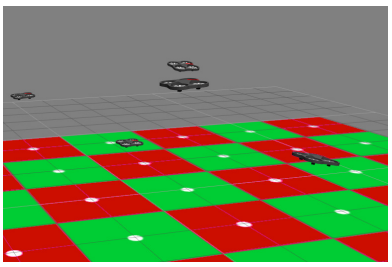

Figure 1: GaTAC simulation with numerous virtual UAVs

Figure 2: GaTAC simulation with two Parrot® AR.Drones

Our scenario exercises the recursive reasoning abilities of a human or artificial agent as they must reason about what the other agents in the system are in turn reasoning about them. This can readily be seen in Fig. 3: as the UAVs know the fugitive is trying to reach the safe-house it makes sense to move toward and listen around this area. The fugitive, however, knowing that the two UAVs are trying to catch it, may react by avoiding a direct path to the safe-house area, thus changing where the UAVs expect to find it and so on. This adds a degree of difficulty and interest not present in single agent or non-strategic multi-agent scenarios.

As an additional scenario, we will present a cooperative game in which both UAVs are required to work together as a team to capture the fugitive. The policies of the UAVs for this demonstration will be generated using a DEC-POMDP solver, with motion of the fugitive modeled as simple noise.

3. ARCHITECTURE

As we see in Fig. 4, GaTAC is a simulation system built on a client/server architecture. Each individual client controls one agent, communicating with the server over UDP using a custom C++ library to send commands and receive percepts. As a result, clients are unaware of whether virtual or physical robotic agents are per-

Appears in: *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May 4–8, 2015, Istanbul, Turkey.*

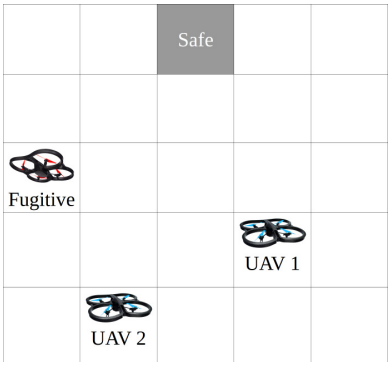

Figure 3: Scenario state space showing the fugitive, safe-house location, and two UAVs

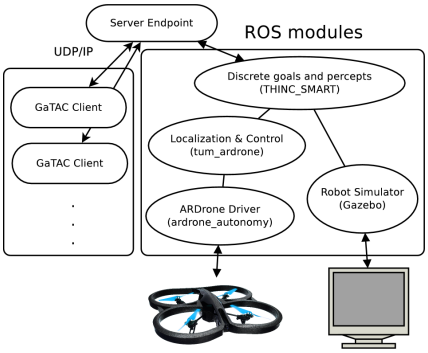

Figure 4: GaTAC clients communicate with our server-based simulation system using UDP/IP. The server may be easily configured to use virtual or real robots with no changes to the clients

forming the simulation. We have implemented a number of clients including random motion, human control, and policy control.

The server coordinates the clients and allows the simulation to be switched between virtual agents and robotic ones. It is built on the Robotic Operating System (ROS) [6] version Hydro, which allows each component to be modularized into a separate package.

3.1 Virtual Robotic Agents

TUM_simulator, a plugin for the Gazebo [5] robot simulator, is used to provide virtual robotic agents with simulated physics, sensors and actuators. Robots are controlled with our custom controller called THINC SMART that is based on the motion of a damped spring. A cropped screenshot of this simulation can be seen in Fig. 1. Varying sizes of maps may be chosen for a scenario and the number of agents (each with their own client) is configurable. Currently up to 12 have been tested but more is possible with suitably powerful computer hardware.

All code used in this demo is open source and will be made available for general use after our demonstration.

3.2 Autonomous UAVs

For simulations involving physical agents we use three Parrot® AR.Drones (both version 1.0 and 2.0) as can be seen in Fig. 2. These small, inexpensive quadcopters are safe for indoor use and have an open-source library available for programming. The ROS package *ardrone-autonomy* provides a driver with which to communicate with each drone. Control is provided by THINC SMART and the package *TUM-ardrone* which implements a PID controller allowing the drone to be reliably moved to a given goal.

The drones come with an array of sensors including downward facing and forward facing cameras, accelerometers, gyroscopes, magnetometers and additionally a GPS module is available. Due to our indoor setting and small map size, however, GPS is not a viable method of localization. Instead each drone uses its available sensors to estimate its location in real time using feature-based SLAM [8] provided by the ROS package *TUM-ardrone*. This package makes use of parallel tracking and mapping (PTAM) [4] as a single-camera based simultaneous localization and mapping (SLAM) solution. A feature map is created, updated, and used for localization to correct for movement errors during flight. To aid with map making in the presence of many moving humans, the drones will be flown constantly oriented towards one side of the map where we will have strategically placed some small non-moving objects.

3.3 Multi-Agent Planning

GaTAC clients can execute policies created by decision-making frameworks that model varied sources of uncertainty. For example, if the combined state space of the agents is fully observable, a DEC-MDP may be used. Depending upon the scenario, policies created by different frameworks may be used to control the agents and their performance directly compared.

Our demonstration scenarios incorporate uncertainty due to the unknown location of multiple agents, imperfect localization of the fugitive, and critically the agent interaction involved. As such, we model the scenarios as a DEC- or I-POMDPs as appropriate. In I-POMDPs, beliefs of the other agents, which a given agent is uncertain about, is a part of the state space. As the other agents are assumed to be reasoning about the beliefs of the given agent as well, this sets up a recursive state space which dramatically increases the difficulty of obtaining an optimal policy. We solve this I-POMDP with point-based value iteration (I-PBVI) [2] and obtain an optimal policy for the subject agent.

Acknowledgement

We gratefully acknowledge partial support from AFOSR through grant FA9550-08-1-0429, an NSF CAREER award, IIS-0845036, and from ONR through grant N000141310870.

REFERENCES

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

[2] P. Doshi and D. Perez. Generalized point based value iteration for interactive pomdps. In *AAAI*, pages 63–68, 2008.

[3] P. J. Gmytrasiewicz and P. Doshi. A framework for sequential planning in multi-agent settings. *JAIR*, 24:49–79, 2005.

[4] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *ISMAR*.

[5] N. Koenig and A. Howard. Design and use paradigms for Gazebo, an open-source multi-robot simulator. In *IROS*, volume 3, pages 2149–2154. IEEE, 2004.

[6] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. ROS: an open-source robot operating system. In *Open-Source Software Workshop, ICRA*, volume 3, page 5, 2009.

[7] E. Sonu and P. Doshi. GaTAC: a scalable and realistic testbed for multiagent decision making (demonstration). In *AAMAS*, pages 1507–1508, 2012.

[8] S. Thrun and J. J. Leonard. Simultaneous localization and mapping. In *Handbook of Robotics*, pages 871–889. Springer, 2008.