Multi-robot Dispatch

Andrew M. Olney
University of Memphis
Institute for Intelligent Systems
aolney@memphis.edu

Abstract

This paper reports the ongoing development of a natural language interface used to implement multi-robot dispatch. A human dispatcher uses the interface to direct multiple robotic tugs performing cargo loading and unloading tasks at an airport. The airport environment is highly dynamic and hazardous, having human ground crews, tugs, and planes as moving obstacles. In this paper we describe the task, current architecture, and development strategy.

Figure 1: An airport tug

1 Introduction

Two research streams have matured to the point that they may be combined for practical, real world applications. The first stream consists of task-oriented natural language interfaces. These systems are multimodal and frequently use an assistant agent that helps the user manage a complex situation ([Allen *et al.*, 1995], [Allen *et al.*, 2000], [Stent *et al.*, 1999]). The second research stream consists of autonomous robots moving at high speeds in hazardous conditions. The recent successes of from DARPA Grand Challenge suggest that these research robots are approaching viability in real world tasks [Montemerlo *et al.*, 2006].

In this paper, we describe a novel problem whose solution draws on both research streams. Our problem is to dispatch and monitor robotic tugs performing cargo loading and unloading tasks at an airport. While previous research indicates that a practical solution is reachable, the dispatch task is of such complexity that a solution is challenging. For example, [Lemon *et al.*, 2003] dispatches a single robot. However, in an airport environment, it is not cost-effective for one person to control one robot when that person could just as easily drive a non-robotic tug. Therefore multiple robots must be controlled by a single human dispatcher, which compounds the complexity of the task. Likewise, the DARPA grand challenge winner [Montemerlo *et al.*, 2006] cannot be seamlessly transported to our domain. Not only is the airport environment full of moving obstacles, but collisions with these obstacles has an unacceptable cost.

The rest of the paper is outlined as follows. Section 2 describes the airport tug task that is managed by the dispatch interface. Section 3 presents an overview of the system's architecture. Section 4 presents the development strategy being used to create the dispatch interface and associated autonomous robots. Section 5 concludes.

2 The Airport Tug Task

Airport tugs are similar in design to riding lawnmowers but are significantly more powerful. They are commonly seen delivering luggage to and from passenger aircraft. However, the airport tug task considered here is somewhat different, because the airport itself is used for commercial cargo rather than passenger transportation. In this situation, each tug pulls a train of trailers carrying containers that fit directly into the cargo hold of the aircraft. The tugs move between the planes and the sort facility where the containers are loaded, with a speed between ten and fifteen miles per hour. Each tug follows an unstructured path that may or may not be marked as a lane. Perhaps most importantly, the tugs are faced with many unpredictable obstructions, including containers, human ground crews, other tugs, and aircraft. The tugs operate at night and in a variety of adverse weather conditions, e.g. rain, fog, and snow. The cost of collision is high: any accident causing injury or damage to an aircraft are unacceptable.

Because the cost of collision is so high, it is desirable to have a human operator overseeing the work of robotic tugs. A human operator is desirable in order to:

- Disable an unsafe robot
- Coordinate and schedule the behavior of multiple robots

- Deal with situations beyond the robot's capabilities
- Act as an intermediary between other employees and the robots
- Create new employment opportunities

A useful existing analogy is currently deployed in some supermarkets. U-scan allows supermarket customers to scan and weigh their own items. A single human supervisor oversees 4-6 stations and deals with customer errors and tasks that are currently hard for computers, e.g. visual identification of produce. Likewise, a human dispatcher for autonomous tugs can assist when needed while delegating simple tasks to the tugs.

3 Architecture

The dispatch natural language interface consists of four components. Speech is transformed to text using Sphinx-4 [Walker *et al.*, 2004]. This text is sent to the Jess rule engine [Hill, 2003], where it is transformed into commands. Each robot is represented as a shadow fact in Jess, which allows Jess rules to be used to monitor the robots. Updates from the system are announced using text to speech from ScanSoft and a 3D interface based on the game Unreal Tournament 2004.

3.1 Speech Recognition

Sphinx-4 is an open-source, pluggable Java implementation of earlier Sphinx speech recognition systems [Walker *et al.*, 2004]. It's high modularity and zero-cost make it highly suitable for human-robot interaction research. In our dispatch natural language interface, Sphinx-4 is used in "push to talk" mode for highest fidelity. We are currently working with the airport personnel to determine the actual noise constraints, but for now we are assuming that a headset microphone with "push to talk" interaction will be sufficient.

The two most significant customizable pieces of a Sphinx-4 application are the acoustic model and the language model. Several existing models of each type are freely available for generic speakers on generic tasks. For high performance in customized tasks, it is desirable to create custom models. For acoustic modeling, Sphinx-4 is somewhat disadvantaged by not being trainable online, i.e. with a speaker enrollment period. For this reason, we are hoping that an speaker-independent acoustic model, when used with a customized language model, will yield acceptable results. Sphinx-4 offers several different language model paradigms, including n-grams, JSGF, and FST. Sphinx-4 also allows online swap out of language models. Currently the dispatch interface uses n-grams in a dictation-style interface while we work with airport personnel to establish the space of tasks and entities involved in tug operations (see Section 4). That information will make it possible to populate the system with JSGF representations.

3.2 Jess

Jess, a Java expert system shell, is a rule engine based on CLIPS [Hill, 2003]. Jess can be used for forward and backward chaining and can easily be connected to ontologies. In the dispatch interface, however, Jess is primarily used for scripting. Scripting is used for two different tasks, command interpretation and robot monitoring.

As stated before, spoken commands are rendered as text and input to Jess. Jess is then used to script commands to the robots. The command structure is imperative [Rayner *et al.*, 2000], but may be modified by logical operations on the Jess facts. This allows a parsimonious blending of logic for reasoning about the world with imperative commands for telling the robots what to do next, rather than forcing a single logical or imperative representation. Although Jess provides a number of built-in operators for scripting, it may be significantly enhanced by customizable user functions that may be defined online. These functions may make atomic a sequence of commands or otherwise provide new building blocks for task performance. Thus the framework allows for both high level control, e.g. "Move cargo from loading bay 1 to aircraft bay 3," and low level control, e.g. "Turn right 90 degrees".

Robot monitoring describes the current status of an "assistant agent" that helps the user manage robot performance. In essence, the robot monitor consists of a number of Jess rules that operate over shadow facts representing the robots. In Jess, shadow facts are a useful feature that allow dynamic real world objects to be represented within the Jess fact space. Shadow facts are attribute-value representations that are updated in real time as the corresponding real world object changes. Using shadow fact representations of robots, rules may be constructed that monitor the state of the robot. For example, if a robot is idle for a long period of time, a rule may be triggered that notifies the user. In the future, notifications will be priority managed so that dialogue interruptions occur only when essential, allowing non-essential notifications to be delayed until the user has completed the current task.

3.3 User Notification

Updates from the system are announced using text to speech from ScanSoft and a 3D interface based on the game Unreal Tournament 2004. Text to speech is handled using Microsoft SAPI [Rozak, 1996], which is supported by all commercial speech engines that run on Windows. This is advantageous because it provides plug and play capability for improved text to speech engines that are being released with increasing frequency. Plug and play also allows the user freedom to exercise personal preference a different speech engine, which should increase user satisfaction.

User notification also takes place through a 3D interface based on Unreal Tournament 2004. Unreal Tournament 2004 is a low-cost, high quality commercial game engine that comes with editing and modification tools and documentation. The Gamebots modification allows agents within the game to be controlled by an external program using network protocols [Kaminka *et al.*, 2002]. Gamebots has been further modified to create a realistic robot simulator (USARsim) for human robot interaction research, specifically unmanned search and rescue (USAR) [Wang *et al.*, 2003]. USARsim has advantages with respect to development (see Section 4) and user notification. For notification, USARsim can display multiple points of view (POV) for a robot, including the robot's camera POV a birds-eye POV centered on the robot. Moreover, USARsim provides a drivable camera that may be

Figure 2: USARsim airport/robot simulation

positioned anywhere in the simulation. As a result, the user can develop a true 3D understanding of the robot's situation.

Once robots are deployed in the real world, the USARsim interface will still be used. However in this case, what is shown to the user is what the robot perceives of its environment, i.e. the robot perceives an obstacle at coordinate (x,y,z), so the USARsim 3D world is updated with an obstacle at that coordinate. In this way, USARsim can provide a 3D representation of the robots' beliefs. Because the robots can have inaccurate beliefs, we propose an interface that, for each robot, displays its front camera video together with its USARsim front camera video, such that the user can be made aware of any major discrepancies between the two. These viewpoints, together with the drivable camera mentioned above, will provide valuable spatial information to the human dispatcher.

4 Development Strategy

A defining characteristic for this project is that the multi-robot dispatch interface is being developed in parallel with the robotic tugs. As a result, development is a moving target of requirements. This naturally suggests an iterative development strategy. Therefore the following steps should not be considered as a pipeline, but rather as landmarks in the development landscape that may be revisited:

- Solve task in simulation
- Move to small scale robots and mock-ups
- Move to robotized tugs and mock-up
- Small scale real trials (one dispatcher)
- Large scale real trials (multiple dispatchers)
- Full deployment

Different teams are currently working on the first two items in parallel. The five other teams are focusing on the autonomous tugs themselves, i.e. vision, localization, navigation, obstacle handling, and cognitive robotics. As these teams progress, new capabilities are added to the robots and others change. In order to limit the disruptiveness of such changes to the dispatch interface, we are working with airport personnel to develop a model of idealized tug behavior. This has been quite difficult to accomplish, because there do not appear to be any written training documents for tug operation.

To acquire the necessary information, we are using current tug operations as a model. We are in the early stages of equipping tugs with handheld video cameras, microphones, and GPS devices. These tugs will be driven as part of normal operations by senior drivers. The data thus recorded, together with satellite imagery we have obtained from Google Earth, will provide a wealth of information concerning actual routes, tug-tug interactions, and driver strategies for dealing with moving objects. This data will also make more apparent the degree of autonomy the tugs can be expected to have, and the corresponding situations where human intervention is likely to be required.

This data will be collected, analyzed, and discussed with the other five teams to obtain a consensus view of the robots' eventual capabilities. The dispatch natural language interface will then be tuned to this idealized model so that user testing (in simulation) can begin. User testing on the idealized model serves not only to validate the natural language interface but also serves to inform the basic robot requirements. As an end-to-end test of the system, the interface and idealized simulation may indicate that low level functionality of the robots needs to be modified. For example, users may expect robot functionality not currently implemented. Likewise, the end-to-end test may reveal that some sophisticated robot functionality may not be needed, since the human operator is easily able to intervene. Such findings would re-prioritize robot requirements and consequently affect the interface. Thus an iterative development cycle would begin anew.

5 Conclusion

This paper reports the ongoing development of a natural language interface used to implement multi-robot dispatch. A human dispatcher uses the interface to direct multiple robotic tugs performing cargo loading and unloading tasks at an airport. The airport environment is highly dynamic and hazardous, having human ground crews, tugs, and planes as moving obstacles. Thus human oversight of robotic operations is essential.

The airport tug task lies at the intersection of research streams in task-oriented natural language interfaces and autonomous mobile robots. However, the airport tug task is of sufficient complexity as to provide practical and research challenges for implementation.

This paper has described the task, current architecture, and development strategy. Immediate future efforts will include data analysis of field work described in Section 4 and user testing of the natural language interface using simulated and idealized robots.

Acknowledgments

The author would like to thank FedEx for its support.

References

[Allen *et al.*, 1995] James F. Allen, Lenhart K. Schubert, George Ferguson, Peter Heeman, Chung Hee Hwang,

Tsuneaki Kato, Marc Light, Nathaniel G. Martin, Bradford W. Miller, Massimo Poesio, and David R. Traum. The TRAINS project: A case study in building a conversational planning agent. *Journal of Experimental and Theoretical AI*, 7:7–48, 1995.

[Allen *et al.*, 2000] James Allen, Donna Byron, Myroslava Dzikovska, George Ferguson, Lucian Galescu, and Amanda Stent. An architecture for a generic dialogue shell. *Natural Language Engineering*, 6, 2000.

[Hill, 2003] Ernest Friedman Hill. *Jess in Action: Java Rule-Based Systems*. Manning Publications Co., Greenwich, CT, USA, 2003.

[Kaminka *et al.*, 2002] Gal A. Kaminka, Manuela M. Veloso, Steve Schaffer, Chris Sollitto, Rogelio Adobbati, Andrew N. Marshall, Andrew Scholer, and Sheila Tejada. Gamebots: a flexible test bed for multiagent team research. *Commun. ACM*, 45(1):43–45, 2002.

[Lemon *et al.*, 2003] Oliver Lemon, Anne Bracy, Alexander Gruenstein, and Stanley Peters. An information state approach in a multi-modal dialogue system for human-robot conversation. In Hans Reiser, Peter Kunlein, and Henk Zeevat, editors, *Perspectives on Dialogue in the new Millennium*, volume 114 of *Pragmatics and Beyond*, pages 229–242. John Benjamins, 2003.

[Montemerlo *et al.*, 2006] Michael Montemerlo, Sebastian Thrun, Hendrik Dahlkamp, David Stavens, and Sven Strohband. Winning the darpa grand challenge with an ai robot. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, Boston, 2006. AAAI.

[Rayner *et al.*, 2000] Manny Rayner, Beth Ann Hockey, and Frankie James. A compact architecture for dialogue management based on scripts and meta-outputs. In *Proceedings of the sixth conference on Applied natural language processing*, pages 112–118, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[Rozak, 1996] Mike Rozak. Talk to your computer and have it answer back with the microsoft speech api. Microsoft systems journal, Microsoft, January 1996.

[Stent *et al.*, 1999] Amanda Stent, John Dowding, Jean Mark Gawron, Elizabeth Owen Bratt, and Robert Moore. The commandtalk spoken dialogue system. In *Proceedings of the Thirty-Seventh Annual Meeting of the ACL*, pages 183–190, University of Maryland, College Park, MD, 1999. Association for Computational Linguistics.

[Walker *et al.*, 2004] Willie Walker, Paul Lamere, Philip Kwok, Bhiksha Raj, Rita Singh, Evandro Gouvea, Peter Wolf, and Joe Woelfel. Sphinx-4: A flexible open source framework for speech recognition. Technical Report TR-2004-139, Sun Microsystems, Inc., November 2004.

[Wang *et al.*, 2003] Jijun Wang, Michael Lewis, and Jeffery Gennari. Usar: A game-based simulation for teleoperation. In *Proceedings of the 47th Annual Meeting of the Human Factors and Ergonomics Society*, pages 493–497, Denver, CO, Oct. 13-17 2003.