A Flexible Architecture for a Multimodal Robot Control Interface

Carl Burke Lisa Harper Dan Loehr

The MITRE Corporation
{cburke, lisah, loehr}@mitre.org

Abstract

Despite increased activity in robotics, relatively few advances have been made in the area of human-robot interaction. The most successful interfaces in the recent RoboCup Rescue competition were teleoperational interfaces. However, some believe that teams of robots under supervisory control may ultimately lead to better performance in real world operations. Such robots would be commanded with high-level commands rather than batch sequences of low-level commands. For humans to command teams of semi-autonomous robots in a dynamically changing environment, the human-robot interface will need to include several aspects of human-human communication. These aspects include cooperatively detecting and resolving problems, making using of conversational and situational context, maintaining contexts across multiple conversations and use of verbal and non-verbal information. This paper describes a demonstration system and dialogue architecture for the multimodal control of robots that is flexibly adaptable to accommodate capabilities and limitations on both PDA and kiosk environments.

Introduction

Robotics research has recently experienced a surge of interest due to a growing awareness that robots can work collaboratively with humans to perform tasks in situations unsafe for humans. The 1997 Mars Sojourner rover was tasked to act as a "mobile remote geologist" and conducted soil experiments in several different terrains (NASA 1997). Teleoperated robots assisted at the site of the World Trade Center in New York City after the September 11 attack. Robots were able to penetrate into areas of rubble debris in cavities too narrow and dangerous for humans and dogs (Kahney 2001). Finally, the US Government's Defense Advanced Research Projects Agency (DARPA) has invested substantial funding toward a vision in which robots will support future combat systems. Future operations are envisioned to be populated by teams of semi-autonomous robots dispersed through the battlefield and controlled remotely by one or several human operators on mobile platforms.

Despite this increased activity in robotics, relatively few advances have been made in the area of human-robot interaction. In RoboCup Rescue 2001 held in Seattle Washington, the best contenders in the competition relied upon teleoperation (joystick-style control) by human controllers (Eyler-Walker, p.c.). Though ultimately supervisory control of teams of semi-autonomous robots is a very promising avenue for future research in robot search and rescue, this technological approach does not yet reach the level of competence of teleoperation. In the 1997 Mars mission, a large operations team was required to command a single robot. The robot had little navigational autonomy and because of communications latency, commands had to be sent in batches of several low-level commands. Recently, NASA has been concerned with human-machine interactions that are commanded by high-level commands rather than sequences of low level commands. A grapefruit-sized Personal Satellite Assistant (PSA) is being developed to operate aboard the Space Shuttle's flight deck. It will navigate using its own navigation sensors, wireless network connections, and propulsion components. Rainer et al. (2000a, 2000b) describe an architecture for a spoken dialogue interface for communicating with the PSA.

An alternative approach to human-robot interaction by Fong et al. (2001) bridges teleoperation with "collaborative control". In this model, humans and robots act as peers exchanging information in dialogue to achieve goals. Instead of controlling the vehicular robot solely by direct (manual) control, the human specifies a set of waypoints that the robot can achieve on its own. One problem observed with waypoint driving is that robots may encounter obstacles for which its vision system is inadequate to assess. In such a circumstance, the robot can query the human about the nature of the obstacle and receive assistance.

In this paper we describe a dialogue management architecture we are developing for two different physical interfaces: a Personal Digital Assistant (PDA)-based dialogue interface and a touch screen kiosk interface to a robot. We plan to extend the PDA-based interface toward a team-based search and rescue task. Currently, both kiosk and PDA modes support single user, single robot dialogue in a limited navigation "RoboCup Rescue" style arena with a simulated victim and various obstacles. Using touch gestures and speech on the PDA interface, users may task the robot, operate camera controls and issue navigational commands. Later we plan to extend to supervisory control of a small team of robots.

Our primary research interest is the development of a dialogue system architecture robust enough to tolerate continuous operational use, flexible enough for porting to different domains and tasks and devices, and able to support multiple, simultaneous conversations involving one or more humans and one or more cooperative robot entities. The dialogue management architecture we are developing is based on the TRINDIKit (Task oRiented Instructional Dialogue Toolkit) (TRINDI 2002) framework, although we have introduced a number of implementational changes in the re-engineering of a TRINDIKit architecture.

Original System Concept

Our architecture was first assembled for development of a demonstration system called Mia, the MITRE Information Assistant. Mia is an information kiosk equipped with a touch screen and a microphone, and stocked with information about MITRE's internal research projects for use as a visitors' guide to a MITRE trade show (see Figure 1). Our intent was to prototype a system in which novice users would be able to step up to the kiosk and immediately be able to speak and point with no coaching or training. In fact, our system met this goal well though we discovered a number of shortcomings during design and development.

Mia was built as a set of independent modules communicating across SRI's Open Agent Architecture (OAA). The Graphical User Interface (GUI) was written in Tcl/Tk. The GUI (see Figure 2) handled push-to-talk for the speech recognizer, maintained a text menu of possible user utterances, showed a map of the overall trade show layout with the ability to zoom in on specific rooms, and displayed pre-recorded output videos of the animated agent speaking and gesturing. Speech recognition was performed by Nuance while gesture input was limited to selection of exhibit rooms and booths on a touch screen. When speaking, Mia appeared as a talking head, as in Figure 2; when gesturing, Mia appeared as a full-body agent, as in the bottom of Figure 1.

In this demonstration system, all output was pre-recorded into discrete video and sound clips using a custom head imported into DEC's FaceWorks and onto a body provided by the Engineering Animation, Inc.'s Jack ergonomic toolkit. Speech synthesis was accomplished via Lernout & Hauspie's concatenative speech synthesizer RealSpeak and synthesized speech was essentially spliced onto the Faceworks head using TechSmith's DubIt. Ideally, we would have generated system output in real time, but our goal was to focus solely on an initial assessment of TRINDIKit's potential for building a multimodal dialogue manager. This system will be described further below.

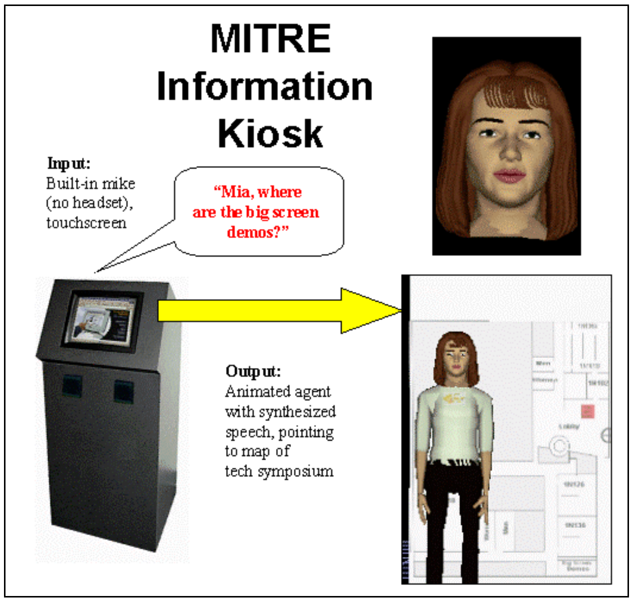
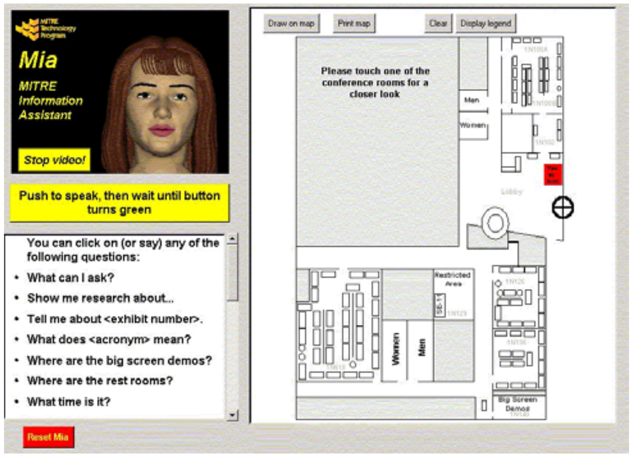

MITRE Information Kiosk
Input:
Built-in mike (no headset), touchscreen
"Mia, where are the big screen demos?"
Output:
Animated agent with synthesized speech, pointing to map of tech symposium

Figure 1: MITRE Information Assistant (Mia)

Robot Control Tasks

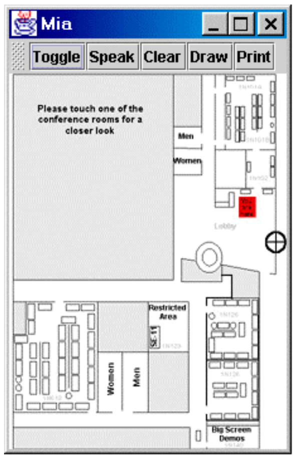
After completion of this first phase, we devised new goals with respect to a robot control task. Our primary goal was to re-use most, if not all, components in a configuration such that we could run virtually the same set of user tasks on both a PDA and kiosk. The difference in available screen space between interface configurations poses a significant challenge, so we are constructing our interfaces for the kiosk (Figure 2) and the PDA (Figure 3) from task-oriented components which can be assembled as desired.



Please touch one of the conference rooms for a closer look

You can click on (or say) any of the following questions:
- What can I ask?
- Show me research about...
- Tell me about <exhibit number>.
- What does <acronym> mean?
- Where are the big screen demos?
- Where are the rest rooms?
- What time is it?

Figure 2: Kiosk Graphical User Interface



Please touch one of the conference rooms for a closer look

Figure 3: PDA Graphical User Interface

With this component-based approach we are able to reuse graphical components such as the trade show map between different configurations, even when it is impractical or undesirable to use identical interfaces.

Our second goal was to extend our system to use robots and humans in teams. We developed two new tasks for the control of three Pioneer 2-AT robots (Figure 4).

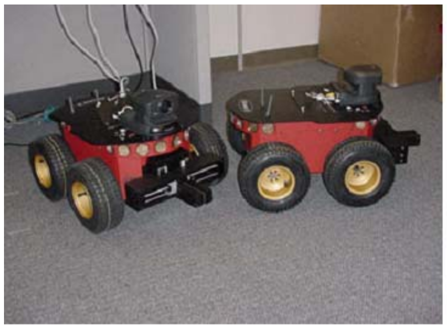


Figure 4

The first task is to assist guiding the robot to a particular location in the arena. Though the robots have the capability to navigate autonomously, we envision the robot encountering obstacles that may require human intervention and assistance. The second task is to provide access to cameras affixed to one of several robots, so that a user can dynamically request various camera views. Figure 5 shows an example dialogue script from such an interaction.

(H) Show me the view from this robot
(R) Okay
(H) Go to this location <pointing on the PDA map>
(R) Okay
(H watching incoming video feed on PDA) Stop here
(R) Okay
(H) Pan left 180 degrees
(R) The maximum I can pan is 95%. Would you like me to turn left 180 degrees and show you the view?
(H) Yes. (Because the video head can only pan approximately 95 degrees left and right)
(R) Okay (executes 180 degree turn and displays view on camera)

Figure 5: Sample Dialogue

These tasks were selected as a starting point for evaluating the usability of a conversational multimodal interface in a future RoboCup Rescue event.

Robot Control Needs Assessment

The robot tasks of navigation and camera control are perhaps less rigid than the question-answer task supported by Mia. However, the same kind of core dialogue functionality is needed. These aspects include cooperatively detecting and resolving problems, making using of conversational and situational context, and making use of verbal and non-verbal information. The robot task as extended for RoboCup Rescue will require one additional aspect not considered for our interim technology symposium demonstration: we will need to be able to maintain contexts across multiple conversations. The human controller will need to be able to communicate with each robot individually and as a team. Furthermore, we expect that robots not participating directly in an exchange may need to over-hear other exchanges. For this reason, our dialogue management toolkit must provide for the maintenance of multiple conversational states. Other considerations with regard to specific implementation issues will be discussed below.

Original Dialogue Management Architecture

TRINDIKit itself provides the basic infrastructure of our dialogue manager. It provides structured data types and the means to define an Information State (IS) from those types, a language for defining the modules of a Dialogue Move Engine (DME), and a language for controlling the application of individual modules to the job of dialogue management. With all TRINDIKit provides, it does not implement a theory of dialogue. For that we used the GoDiS (Gothenburg Dialogue System) (Larsson et al 2000) system, which implements the Questions Under Discussion model in TRINDIKit. We were able to adapt existing GoDiS dialogues to our kiosk domain in a very short time.

In order to integrate TRINDIKit into the kiosk using the OAA, we used TRINDIKit's concurrent mode, which incorporates support for use of the OAA. While this seemed to be a natural choice, and allowed more natural definition of module interactions, it also raised several problems, as discussed below.

Speed

TRINDIKit in concurrent mode ran very slowly. Although modules run independently in concurrent mode, updates to IS were still transmitted to each module individually. Updates were sent whether they were used by that module or not, and all other processing waited until that module finished its work. Furthermore, the original TRINDIKit code was uncompiled SICStus Prolog was not compilable using original source code.

Data Consistency

TRINDIKit does not exercise good controls over asynchronous modifications to an Information State. At one point we had to build artificial delays into our system to work around these limitations. The dialogue manager we built for Mia was based on GoDiS, which requires very structured turn-taking. In several cases, however, the interactions with the user flowed better if these responses were automatic. Processing was sufficiently slow that our GUI's automatic acknowledgement often arrived and was processed before TRINDIKit was finished cleaning up from the previous utterance. As a result, it was possible to change the IS twice before the DME could respond to one change, and the system lost track of the dialogue state. We feel this is an unacceptable outcome. It's also unwise to depend on faster execution to solve the problem; consistency of data has to be assured throughout the design of the system.

Inconsistent Semantics

We encountered situations where constructs of the GoDiS plan language were interpreted differently depending on the depth of the plan. With the proliferation of small languages implemented by different sets of macros, it was difficult to track down bugs in the rules and conversation scripts. This was made more difficult by the nature of Prolog. Clauses that fail do not normally generate any error messages, because failure is a normal aspect of program execution. Unfortunately, database bugs and misspelled names often caused unexpected failures, causing the system to generate either no response or a response that looked reasonable but was in fact incorrect. We feel it's necessary to provide explicit notification of certain kinds of failure, such as failure to find a named variable, failure to find a matching value in a table, and so on.

Lack of Multimodal Support

Neither TRINDIKit nor GoDiS provides any direct support for multimodal processing. The primary interface driving the development of these systems was language; there is no separation of events by source, no temporal tagging of input events, and no provision for assessing temporal relationships between different inputs. For Mia, we worked around this by using touch to generate events that were indistinguishable from the output of the speech recognizer agent. For the robot control task, where we would like to allow the user the freedom to refer to locations or paths in a more natural way, this workaround doesn't let us get at the problem in a meaningful way.

Revised Dialogue Management Architecture

We are moving ahead with the design for a dialogue manager for robot control. From our experience with the dialogue manager in Mia, we're convinced of the advantages of a kit-based approach. We feel that TRINDIKit was a good first cut at it, and hope that our efforts will lead to a second, somewhat better iteration.

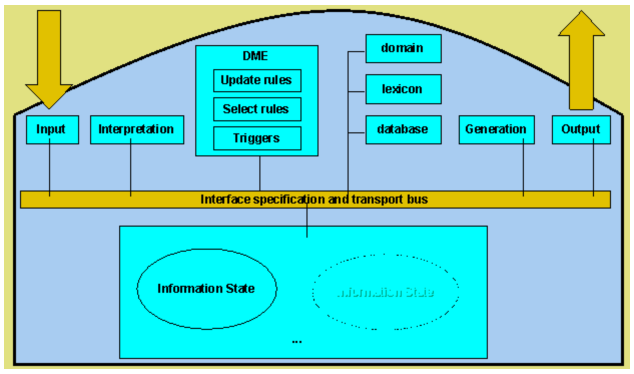


Figure 5: Dialog manager architecture

Our dialog management architecture, with the GoDiS suite of modules, is shown in figure 5. It comprises a set of software modules (some generic, some domain-dependent) communicating with and through a distributed information state; it provides a basic set of structured datatypes for implementing the information state; it provides languages for writing rules and algorithms linked closely with the information state, to implement theories of dialog within a DME; and it contains tools to support the development and testing of dialog systems. While the overall design is similar to the TRINDIKit architecture, our kit differs in several aspects which incorporate the lessons learned from our experience with TRINDIKit.

Distributed Information State

We've chosen to model all of our module interactions as if they were asynchronous. This provides the cleanest separation of modules, and the cleanest conceptual integration with the asynchronous requirements of robot control. Our approach to solving this problem is to define an explicit interface definition language, which will be used to define every module's interface with the outside world. We explicitly include the information state structure in this interface definition, perhaps as a module in itself. Since TRINDIKit does not include a separate language for specifying module interfaces, we are designing our own. This language is analogous to CORBA Interface Definition Language, but with less concern for the physical implementation.

There are a large number of protocols and infrastructures that have been developed to support communications between agents, each with characteristics optimized for particular tasks or emphasizing desired functionality. We intend to support small standard set of operations that have wide applicability across programming languages and communication protocols.

Controlled Extensibility

Our interface specifications will need to be translated into specific computer languages before they can be executed. The translation will vary depending on the underlying protocol used to communicate between modules. While we want to support the widest possible audience, we don't want to get bogged down in the construction of translators for every possible set of implementation language and protocol. Our approach is to exploit an existing standard set of translation software, namely XML and XSLT processors such as Xalan. We are specifying a dialect of XML for modules interface definitions, and a small set of templates for realizing interfaces with specific combinations of programming language and protocol. Additional templates can be written as necessary to extend the kit to other languages and protocols without requiring modification of the kit itself.

The same approach extends to the specifications of DME rules, module synchronization and control, and the definition of new "languages" for the kit. We have defined what well-formed formulas look like in our kit's scripting language: what names look like, the types of expressions that are possible, how expressions and statements are aggregated to form programs. What is left unspecified is the exact sequences of expressions that form statements in any particular script language. Those are specified using templates analogous to XML DTDs, which gives us the flexibility to define new constructs as needed.

Rule Engine

The DME rules in TRINDIKit have strong similarities to rules in expert systems. We plan to implement these rules in both a sequential form, equivalent to the current TRINDIKit, and in an expert system form which may be more efficient. We expect that there will be differences in operating characteristics between those two styles, and we want to identify and quantify those differences.

One issue we must address in our design is unification. While logic variables are natural for modelling discourse given the history of the field, most languages typically used to implement robot software do not support it directly. Our kit must ensure that sound unification procedures are provided for every language it supports, so that semantics are common across all realizations of a script. We must also provide for backtracking or iteration through the set of alternatives in a straightforward way.

Control and Synchronization

Our primary focus is multimodal communication, potentially multiparty as well. We are extending TRINDIKit's triggers to include support for consideration of temporal relationships between events, both within and across modes.

Integrated Environment

An ideal toolkit would have an integrated set of tools for designing, testing, and debugging dialogues. We would like to support static and dynamic analysis of dialogues, recording and playback of dialogues, graphical dialogue design tools, a "validation suite" of tests to support extension of the toolkit to new programming languages and agent protocols, and above all, the ability to plug-in as-yet-undefined capabilities as needed.

Future Work

Our two current goals are 1) to reach the same level of functionality as the TRINDIKit/GoDiS system, using our approach; and 2) use the same dialogue manager from two different interfaces. Significant effort has been devoted to defining our mutable language capability. This capability provides both a reasonable transition path from TRINDIKit scripts and a means for specifying module interfaces and information state structure using a common XML representation.

Our intent is to provide support for several different transport mechanisms to explore the limitations of our approach. To date, we have completed an initial interface definition specification and have developed templates to realize those interfaces with the OAA. DARPA's Galaxy Communicator is the second transport mechanism we will be considering. Time and resources permitting, we will examine some additional transports with differing characteristics, such as CORBA, Java Remote Method Invocation (RMI), or Linda.

We have devoted considerable time to up-front consideration of scripting languages, portable code generation, and module communications, and are now beginning the task of implementing our versions of the TRINDIKit scripting languages. Our target realization for these scripts is a combination of Java code and expert systems that can be executed within a Java program. As the kit becomes more mature, we would like to add additional languages such as Prolog and C to support both the computational linguistics community and the developers of software systems.

Finally, we plan to experiment with different configurations of DM software on small devices. It isn't clear to us yet how we should best allocate functions across devices, particularly when those devices have relatively meagre resources to devote to dialogue management. While a small footprint is not one of our primary goals, it would be instructive to discover just how small and efficient a kit-based dialogue manager can be.

Conclusion

We have described our evolving architecture (based on the TRINDIKit framework) for a flexible dialogue manager capable of robust, multimodal, multiparty control of robots. A dialogue manager constructed with the toolkit is used to drive both PDA and kiosk interfaces.

References

Fong T., C. Thorpe, and C. Baur (2002), Robot as Partner: Vehicle Teleoperation with Collaborative Control, Workshop on Multi-Robot Systems NRL, Washington, D.C.

Galaxy Communicator (2002) Web Site. http://communicator.sourceforge.net.

JESS (2002) Web Site. http://herzberg.ca.sandia.gov/jess/

Kahney, Leander (2001) *Robots Scour WTC Wreckage.* Wired Magazine, http://www.wired.com/news/print/0,1294,46930,00.html

Larsson, Staffan, Robin Cooper, Stina Ericsson (2000) *System Description of GoDis*. Third Workshop in Human-Computer Conversation, Bellagio, Italy.

NASA (1997) *Past Missions – Mars Pathfinder*. NASA, http://www.jpl.nasa.gov/missions/past/marspathfinder.html

Rayner, M., B.A. Hockey, and F. James. (2000a) *Turning Speech into Scripts.* AAAI Spring Symposium on Natural Dialogues with Practical Robotic Devices.

Rayner, M., B.A. Hockey, and F. James (2000b) *A compact architecture for dialogue management based on scripts and meta-outputs.* Proceedings of Applied Natural Language Processing (ANLP).

Perzanowski, Dennis, A. Schultz, and W. Adams, (1998) *Integrating Natural Language and Gesture in a Robotics Domain*. Proceedings of the IEEE International Symposium on Intelligent Control: ISIC/CIRA/ISAS Joint Conference. Gaithersburg, MD: National Institute of Standards and Technology, pp. 247-252.

Triesch, Jochen and Christoph von der Malsburg (1998) *A gesture interface for Human-Robot Interaction*. In: Proceedings of the Third IEEE International Conference on Automatic Face and Gesture Recognition (FG'98), April 14-16 1998 in Nara, Japan.

TRINDI Website 2002. http://www.ling.gu.se/projekt/trindi.

OAA Website. 2002. http://www.ai.sri.com/OAA.