Proceedings of the 2002 IEEE
Int. Workshop on Robot and Human Interactive Communication
Berlin, Germany, Sept. 25-27, 2002

A Prototype Robot Speech Interface with Multimodal Feedback

Mathias Haage[+], Susanne Schötz[×], Pierre Nugues[+]

[+]Dept. of Computer Science, Lund Institute of Technology,
SE-221 00 Lund, Sweden;
E-mail: Mathias.Haage@cs.lth.se, Pierre.Nugues@cs.lth.se

[×]Dept. of Linguistics, Lund University,
SE-221 00 Lund, Sweden;
E-mail: Susanne.Schotz@ling.lu.se

Abstract

Speech recognition is available on ordinary personal computers and is starting to appear in standard software applications. A known problem with speech interfaces is their integration into current graphical user interfaces. This paper reports on a prototype developed for studying integration of speech into graphical interfaces aimed towards programming of industrial robot arms. The aim of the prototype is to develop a speech system for designing robot trajectories that would fit well with current CAD paradigms.

1 Introduction

Industrial robot programming interfaces provide a challenging experimental context for researching integration issues on speech and graphical interfaces. Most programming issues are inherently abstract and therefore difficult to visualize and discuss, but robot programming revolves around the task of making a robot move in a desired manner. It is easy to visualize and discuss task accomplishments in terms of robot movements. At the same time robot programming is quite complex, requiring large feature-rich user interfaces to design a program, implying a high learning threshold and specialist competence. This is the kind of interface that would probably benefit the most from a multimodal approach.

This paper reports on a prototype speech user interface developed for studying multimodal user interfaces in the context of industrial robot programming [5]. The prototype is restricted to manipulator-oriented robot programming. It tries to enhance a dialogue, or a design tool, in a larger programming tool. This approach has several advantages:

- The speech vocabulary can be quite limited because the interface is concerned with a specific task.
- A complete system decoupled from existing programming tools may be developed to allow precise experiment control.
- It is feasible to integrate the system into an existing tool in order to test it in a live environment.

The aim of the prototype is to develop a speech system for designing robot trajectories that would fit well with current CAD paradigms. The prototype could later be integrated into CAD software as a plug-in.

Further motivation lies in the fact that current available speech interfaces seem to be capable of handling small vocabularies efficiently, with performance gradually decreasing as the size of the vocabulary increases. This makes it interesting to examine the impact of small domain-specific speech interfaces on larger user interface designs, perhaps having several different domains and collecting them in user interface dialogues.

The purpose of the prototype is to provide an experimental platform for investigating the usefulness of speech in robot programming tools. The high learning threshold and complexity of available programming tools makes it important to find means to increase usability. Speech offers a promising approach.

The paper is organized as follows: speech, multimodal interfaces, and robot programming tools are briefly recapitulated. Then, the prototype is described giving the design rationale, the system architecture, the different system parts, and a description of an example dialogue design. The paper concludes with a discussion of ongoing experiments and future enhancements to the prototype.

Figure 1: SAPI 5.1 speech interface application front end with a list of available command words.

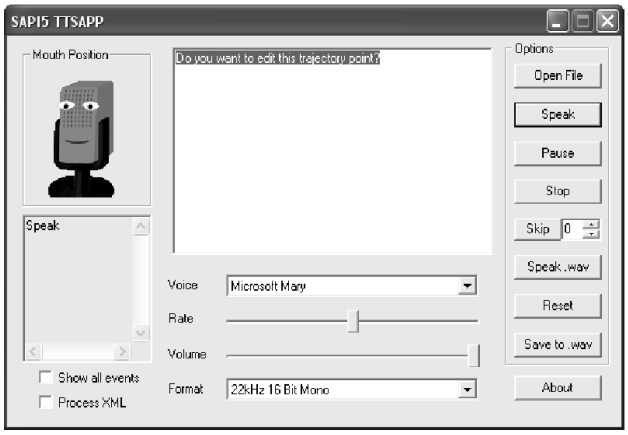

Figure 2: The SAPI 5.1 sample TTS application modified for use by the prototype system.

2 Speech, multimodal interfaces and robot programming tools

2.1 Speech recognition and synthesis

Speech software has two goals: trying to recognize words and sentences from voice or trying to synthesize voice from words and sentences. Most user interfaces involving speech need to both recognize spoken utterances and synthesize voice. Recognized words can be used directly for command & control, data entry, or document preparation. They can also serve as the input to natural language processing and dialogue systems. Voice synthesis provides feedback to the user. An example is the Swedish Automobile Registry service providing a telephone speech interface with recognition and synthesis allowing a user to query about a car owner knowing the car registration plate number.

A problem with speech interfaces is erroneous interpretations that must be dealt with [8]. One approach to deal with it is to use other modalities for fallback or early fault detection.

2.2 Multimodal user interfaces

A multimodal user interface makes use of several modalities in the same user interface. For instance, it is common to provide auditory feedback on operations in graphical user interfaces by playing small sounds marking important stages, such as the finish of a lenghty compilation in the Microsoft Visual C++ application. Rosenfeld gives an overview in [7].

Different modalities should complement each other in order to enhance the usability of the interface. Many graphical interfaces, including robot programming interfaces, are of the direct-manipulation type. Speech should therefore complement direct-manipulation interfaces [2]. Grasso [4] lists complementary strengths and weaknesses related to direct-manipulation and speech interfaces:

- Direct manipulation requires user interaction. It relies on direct engagement and simple actions.
- The graphical language used in direct manipulation interfaces demands consistent look and feel and no reference ambiguity to be usable. This makes it best suited for simple actions using visible and limited references.
- Speech interface is a passive form of communication. The medium allows for describing and executing complex actions using invisible and multiple references. It does not require use of eyes and hands making it suitable for hand-eye free operations.

Put in other words: speech might be used to avoid situations where you know exactly what you want to do but do not have a clue as where to find it in the graphical user interface. It may also help to avoid situations when you are able to describe an operation but do not know how it is expressed in the user interface.

2.3 Industrial robot programming interfaces

Essentially all robot programming boils down to the question of how to place a known point on the robot at a wanted position and orientation in space at a certain point in time.

For industrial robot arms, the known point is often referred to as the tool center point (TCP), which is the point where tools are attached to the robot. For instance, a robot arm might hold an arc-welding tool to join work pieces together through welding. Most robot programming tasks deal with the specification of paths for such trajectories [3].

Below is discussed how modeling of trajectories is performed in three different tool categories for programming industrial robots.

Teach pendant

A single robot operated by a person on the factory floor is normally programmed using a handheld terminal. The terminal is a quite versatile device. For instance, the ABB handheld terminal offers full programmability of the robot. The terminal has a joystick for manual control of the robot. Function buttons or pull-down menus in the terminal window give access to other features. Program editing is performed in a syntax-based editor using the same interface as for manual operation, i.e. all instructions and attributes are selected in menus. Special application support can be defined in a way uniform to the standard interface.

Trajectories are designed by either jogging the robot to desired positions and record them or by programming the robot in a programming language. For ABB robots the programming language used is called RAPID [1].

Off-line programming and simulation tools

In engineering environments, programming is typically performed using an off-line programming tool. An example is the Envision off-line programming and simulation tool available from Delmia. These tools usually contain: An integrated development environment. A simulation environment for running robot programs. A virtual world for visualizing running simulations and being used as a direct manipulation interface for specifying trajectories.

Trajectories are designed by programming them in a domain-specific language or by directly specifying points along the trajectory. The simulation environment provides extensive error checking capabilities.

CAD and task level programming tools

Task level programming tools typically auto-generate robot programs given CAD data and a specific task, for instance to weld ship sections. The software works by importing CAD data and automatically calculate necessary weld trajectories, assign weld tasks to robots and generate programs for these robots. These tools are typically used for programming large-scale manufacturing systems.

IDE	*Visualization*	*Programming*
Teach pendant	Real env.	Jogging & lang.
Off-line tool	Virtual env.	Lang. & sim.
Task-level tool	Virtual env.	CAD data

Table 1: Visualization and programming in different categories of robot programming tools.

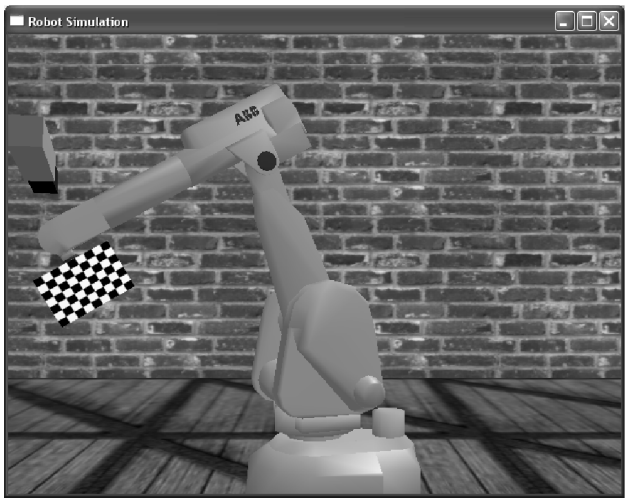

Figure 3: Virtual ABB IRB 2000 industrial robot arm with 6 degrees of freedom (developed in cooperation with Tomas Olsson, Dept. of Automatic Control, Lund University, email: tomas.olsson@control.lth.se).

3 Prototype

Two observations can be made concerning the user interfaces in the above programming environments: The typical task performed by all IDEs (Integrated Development Environment) is to model task specific robot trajectories, which is done with more or less automation, depending on tool category. The user interface consists of a visualization and a programming part, see Table 1.

The prototype presented here is a user interface where speech has been chosen to be the primary interaction modality but is used in the presence of several feedback modalities. Available feedback modalities are text, speech synthesis and 3D graphics.

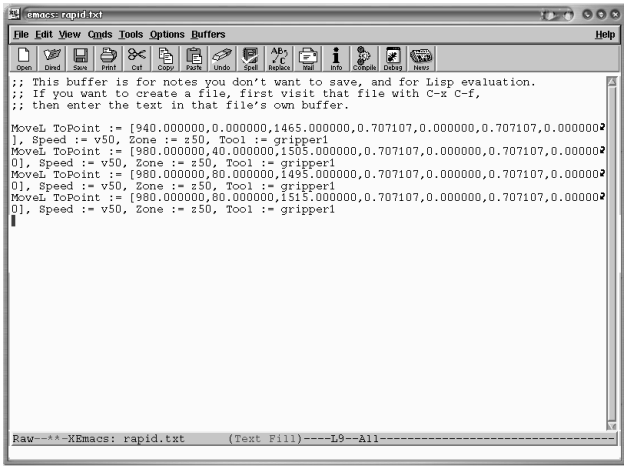

Figure 4: XEmacs is used as trajectory editor and database.

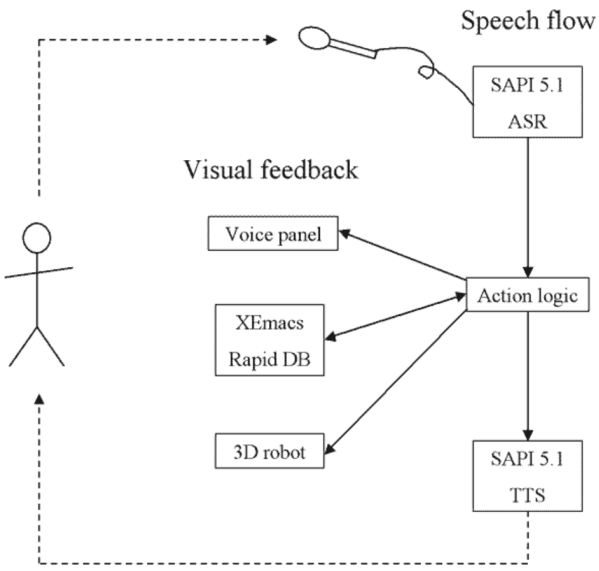

Figure 5: Prototype system dataflow.

The prototype system utilizes the speech recognition available in the Microsoft Speech API 5.1 software development kit. The SAPI can work in two modes: command mode recognizing limited vocabularies and dictation mode recognizing a large set of words and using statistical word phrase corrections. The prototype uses the command mode. It is thus able to recognize isolated words or short phrases [6].

The system architecture uses several applications (see Figures 1, 2, 3, 4): The *Automated Speech Recognition* application, which uses SAPI 5.1 to recognize a limited domain of spoken user commands. Visual feedback is provided in the Voice Panel window with available voice commands. The *Action Logic application*, which controls the user interface system dataflow and is the heart of the prototype. The *Text-To-Speech* application synthesizing user voice feedback. The *XEmacs* application acting as a database of RAPID commands and also allowing keyboard editing of RAPID programs. The *3D Robot* application providing a visualization of the robot equipment.

A decision was made to not use any existing CAD programming system in the prototype. The reasons were twofold: extending an existing system would limit the interaction into what the system allowed, making it difficult to easily adjust parameters like the appearance of the 3D world and the behavior of the editor. The second reason is that by not including a commercial programming system it is possible to release this prototype into the open source community as a complete system.

3.1 System architecture

The prototype system architecture follows a traditional client-server approach. The action logic application acts as a server with all other applications acting as clients. Interprocess communication is performed using Microsoft Win32 named pipes and sockets.

The system dataflow is centered around the speech applications since it is the primary modality of the system. Basically information flows from the speech TTS to speech synthesis application through the action logic application. The action logic application then interacts with the other applications (XEmacs, 3D robot) in order to update the state and different views supported in the interface (Figure 5).

3.2 Prototype applications

Action Logic

The action logic application is the heart of the system. All information goes through this application. The logic controlling the interface is hidden here.

The basic work flow of the application is:

1. Receive spoken commands from the speech recognition application.

2. Interpret the commands and act accordingly: *Send Lisp editing commands* to the XEmacs editor that is storing the trajectory as a sequence of RAPID MoveL (Move Linear) commands. *Read trajectory* stored in XEmacs and send it to the 3D

application for execution and visualization. *Send feedback* to be spoken to the speech synthesis application.

Microsoft SAPI 5.1 speech recognition and synthesis

The speech recognition and synthesis applications are based on the Microsoft Speech API version 5.1. Each application is built by utilizing an example application delivered together with the SDK and modifying it for our purposes. The example applications used for the prototype are CoffeeS0 and TTSApp.

The modifications necessary were quite small. They included: Adding communication capabilities to the applications so that they could send and receive information from the action logic application. This was done by adding a new communication thread to the application. Modifying the application window message handler to issue and receive speech messages from the new communication code. Changing the user interface to show our domain-specific vocabulary. And finally tune the speech recognition application to our vocabulary. This was done by rewriting the default XML grammar read into the speech recognition application upon initialization.

XEmacs RAPID trajectory editing and database

XEmacs is utilized as a combined database, editing and text visualization tool. The trajectory being edited is stored in an XEmacs buffer in the form of a sequence of RAPID MoveL commands:

```
MoveL ToPoint := [940,0,1465,0.707,0,0.707,0],
   Speed := v50, Zone := z50, Tool := gripper1
MoveL ToPoint := [980,80,1495,0.707,0,0.707,0],
   Speed := v50, Zone := z50, Tool := gripper1
```

The trajectory code is visualized in text form in the XEmacs buffer window. It may be edited using normal XEmacs commands. Thus the interface, even if developed with speech in focus, allows alternate interaction forms.

The interaction between XEmacs and the action logic application is done using LISP, see Table 2. The action logic application phrases database insertion/removal/modification commands of trajectory parts as buffer editing commands. These are executed as batch jobs on the XEmacs editor using the gnuserv and gnuclient package.

Spoken command	*Emacs LISP*
Add point	(kill-new "MoveL..."), (yank)
Remove point	(kill-entire-line)
Move forward	(forward-line 1)
Move backward	(forward-line -1)

Table 2: Sample LISP editing command sent to the Emacs RAPID database in response to spoken commands.

Virtual environment

The prototype needed a replacement for the 3D visualization usually shipped with robot programming applications to be realistic. A small 3D viewer previously developed was taken and enhanced with interpretation and simulation capabilities for a small subset of the RAPID language.

The tool is capable of acting as a player of trajectories stored in the XEmacs database. Player commands (play, reverse, stop, pause) is controlled from the action logic application.

3.3 Dialogue design

A preliminary experiment based on Wizard-of-Oz data obtained from the authors has been implemented.

The basic idea of this interface is to view trajectory modeling as editing a movie. It is possible to play the trajectory on the 3D visualizer, insert new trajectory segments at the current point on the trajectory, remove trajectory segments, and moving along the trajectory backward and forward using different speeds.

All editing is controlled using spoken commands, see Table 3. The user gets feedback in the form of a synthesized voice repeating the last issued command, seeing the trajectory in text form in the XEmacs buffer window and seeing the trajectory being executed in the 3D window. The command is always repeated by a synthesized voice in order to detect erroneous interpretations immediately. At some points (for critical operations like removal of trajectory segments), the interface asks the user if he/she wants to complete the operation.

4 Ongoing experiments and future work

The prototype will be used to explore the design space of speech interfaces with multimodal feedback. Below follows a few issues that would be interesting to gather data on:

- Varying the degree of voice feedback, as well as the type of information conveyed.

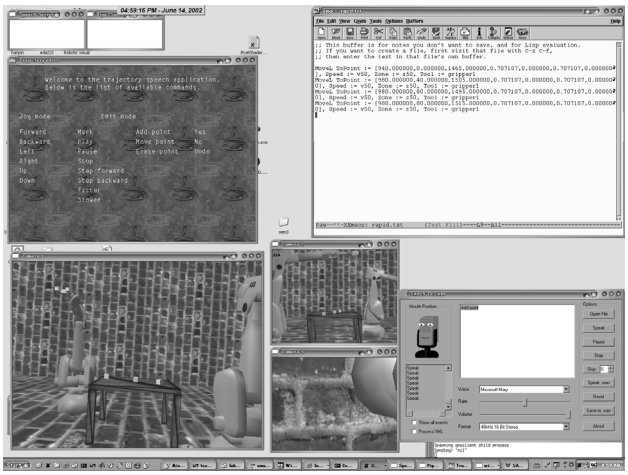

Figure 6: The prototype system user interface consists of four windows; 1. The voice panel containing lists of available voice commands. 2. The XEmacs editor containing the RAPID program statements. 3. The 3D visualization showing the current state of the hardware. 4. The TTS application showing the spoken text.

- Varying between different kinds of visual feedback.
- Varying the command vocabulary and interface functionality. For instance by allowing some task level abstractions in movement specifications, i.e. move to object, grab object.

For the future, there is a list of wanted extensions:

- Allow multiple domains in the speech recognition application, with the option of choosing which one to be applied from the action logic application. This feature could be used to test speech interfaces with state.
- Allow the entire experiment interface configuration to be specified in XML. Remove all hacking necessary to tune the interface. This would also speed up development since it would be easy to switch between different configurations.

5 Conclusion

We have developed a speech interface to edit robot trajectories. An architecture based on reusable application modules was proposed and implemented.

The work is aimed at studying feasability and usefulness of adding a speech component to existing software for programming robots. Initial feedback from users of the interface are encouraging. The users, including the authors, almost immediately wanted to raise the abstraction level of the interface by referring to objects in the surrounding virtual environment. This suggests that a future interface enhancement in such direction could be fruitful.

Spoken commands	*Purpose*
Forward, backward, left, right, up, down	Jog robot
Play, stop, step forward, step backward, faster, slower	Play trajectory
Mark, add point, move point, erase point	Edit trajectory
Yes, no	User response
Undo	Undo

Table 3: Vocabulary used in the prototype.

References

[1] ABB Flexible Automation, S-72168 Västerås, Sweden. *RAPID Reference Manual.* Art. No. 3HAC 7783-1.

[2] Cohen, Philip R. *The Role of Natural Language in a Multimodal Interface.* UIST'92 Conference Proceedings. Monterey, California. p. 143-149. 1992.

[3] Craig, John J. *Introduction to Robotics.* Addison-Wesley Publishing Company. Reading, Massachusetts. 1989.

[4] Grasso, Michael A, Ebert, David S, Finin, Timothy W. *The Integrality of Speech in Multimodal Interfaces.* ACM Transactions on Computer-Human Intraction, Vol 5, No 4. p. 303-325. 1998.

[5] *Prototype homepage*, `http://www.cs.lth.se/~mathias/speech/`.

[6] *Microsoft Speech Technologies*, `http://www.microsoft.com/speech/`.

[7] Rosenfeld, Ronald, Olsen, Dan, Rudnicky, Alex. *Universal Speech Interfaces.* Interactions November + December. p. 34-44. 2001.

[8] Suhm, B., Myers, B., Waibel, A. *Multimodal Error Correction for Speech User Interfaces.* ACM Transactions on Computer-Human Interaction, Vol. 8, No. 1. p. 60-98. 2001.