Java Front-end for Web-based Multimodal Human-Computer Interaction

Xing Jing, Jie Yang, Minh Tue Vo and Alex Waibel

Interactive Systems Laboratories
Carnegie Mellon University
Pittsburgh, PA 15213-3890, U.S.A.
Email: {xjing, yang+, tue, ahw}@cs.cmu.edu

ABSTRACT

As the Java programming environment offers immediate capabilities for creating cross platform interactive applications, the web-based universal multimodal interface becomes possible. In this paper we present a Java front-end for multimodal human computer interaction. We employ a client/server architecture to solve the problems with web-based multimodal interfaces. We address the problems in both system design and implementation. The feasibility of the proposed approach has been demonstrated by a web-based directory assistant system and a multimodal interface for medical applications.

1. INTRODUCTION

The World Wide Web (WWW) has provided a very effective means for accessing and disseminating multimedia information. As the WWW becomes more and more popular, we have to find a more efficient way of creating and manipulating multimedia information on the web. The effectiveness of multimodal human-computer interaction has been demonstrated by many researchers for many applications. Over the last few years, the Interactive Systems Laboratories (Carnegie Mellon University and University of Karlsruhe) have been developing multimodal human-computer interfaces by providing speech, handwriting and gesture in a integrated multimodal environment [1][2][3].

It will also be natural to employ multimodal interfaces on the web. However, the application of multimodal interfaces to the web has been limited by several factors. First, processing multimodal inputs needs a lot of computing power. Second, a multimodal interface takes a large space. Third, it requires a certain expertise to setup and maintain a multimodal interface. Finally, different platforms require different software supports. One way to remove these limitations is to use a client/server architecture, i.e., a multimodal server which performs speech, handwriting, gesture recognition, and multimodal interpretation provides services to a front-end residing in different web browsers.

There are two problems associated with the multimodal server approach: network bandwidth and machine independency of the client side programs. One sentence of normal speech data takes about 1 megabytes. If all the speech data are sent through the network, it will worsen the network traffic. In fact, we only need to handle speech and pen inputs for modalities of speech, handwriting and gesture. Fortunately, there is a lot of redundancy in the speech data. Our experiments indicate that feature data needed for speech recognition are less than 10% of raw speech data. Based on this fact, we can preprocess the speech data on the client side and send only feature data through the network. Furthermore, handwriting and gesture recognition needs only the coordinates of handwriting and gesture strokes, which take little bandwidth. In order to minimize the efforts of developing web front-end for cross-platform interaction, it is desirable to use a machine independent programming language. Java, a programming language developed by SUN Microsystems, perfectly matches our needs.

In this paper we present a Java front-end for multimodal human computer interaction. The basic idea is to use Java front-end to preprocess inputs from different modalities (speech, handwriting, and pen gesture), send feature data through the network, perform recognition and interpretation in the server. We address the problems in both system design and implementation. We describe the system structure and detailed interface for speech, handwriting, and gesture inputs from a web browser. The feasibility of the Java front-end has been demonstrated by a web-based directory assistant system and a multimodal interface for medical applications.

2. SYSTEM STRUCTURE

The system structure is shown in Figure 1. The system consists of a multimodal server and a Java front-end. The multimodal server performs speech, handwriting, and gesture recognition and multimodal joint interpretation. The multimodal server communicates with the Java front-end through two communication servers: i.e., HTTP server and TCP/IP socket server. Because a web page has to be supported by an HTTP server, we have developed a server monitor by using the CGI script. Whenever our web-based multimodal demo page is loaded by a user, the server monitor will check all related programs. If any program crashes, the server monitor will restart it automatically.

The Java front-end handles multimodal inputs, data preprocessing, and communication between the server and the

front-end. We will discuss the Java front-end design and implementation in next section.

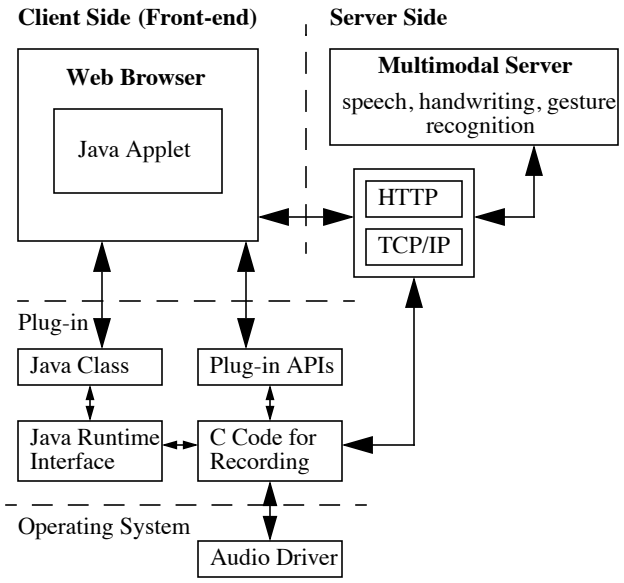

FIGURE 1: System Structure

3. JAVA FRONT-END

The Java front-end includes a Java applet and a web browser plug-in. They both are capable of running Java programs. When a web browser is equipped with the Java front-end, a user can easily use the browser to interact with the WWW not only by keyboard and mouse but also by speaking and drawing. Real-time requirement is a challenge for developing a web-based multimodal interface. Even without network delay, recognition itself takes time. In order to increase user acceptance, we use a run-on mode in the system. That is, the front-end starts sending data back to server as soon as the user inputs and ships partial recognition result back as soon as it is available. The recognizer is also in the run-on mode, i.e., it processes or recognizes what it has gotten so far while receiving more data.

3.1. Java Applet Interface

A Java applet is simply a Java program that executes in the context of a web browser capable of executing that program. Our Java applet was designed to capture pen inputs and control the speech recording plug-in at high level. The Java applet runs in a run-on mode, namely it can receive and display results while handling input events. The Java builtin support for threads provides a powerful tool to improve interactive performance. In order to enhance the interface performance, a mutilthread strategy has been used to handle the inputs, outputs, and interface management.

3.2. Speech Recording Plug-in

Plug-ins are dynamic code modules, native to a specific platform on which the browser runs. The primary goal of the plug-in is to allow existing platform dependent code to seamlessly integrate with and enhance browser's core functionality. Our plug-in is used for recording speech data and performing feature extraction. Because the system uses server structure, the Java front-end has to send various data to the server for processing. The amount of data transferred through the network will directly effect the efficiency of the system. As mentioned in the previous section, we can preprocess the speech data locally and only send the feature data to the server while it is doing recording. The feature data consumes much less bandwidth than original raw data; for example, 512 byte raw data can be reduced to 21 byte feature data (only 5 percent of raw data) in our system. The plug-in records speech data and extracts the features. It is controlled by the Java applet. In order to reduce the response time, the loop of the recording and data preprocessing was designed in such a way that feature data will be sent to server while a user is recording. When the user finishes recording, only data in the last buffer has to be sent to the server.

3.3. Network Communication Protocol

The Hypertext Transfer Protocol (HTTP) currently dominates internet traffic, but it is a simple one-directional request/response protocol. This protocol might work well for browsing a web page in normal fashion using keyboard and mouse; however, in a web-based multimodal interface, additional communication protocol is needed besides HTTP. We use two TCP/IP socket servers for each run-on modality, one for receiving data and another for sending results. Because the inputs and outputs of the Java applet are multithreaded it can receive the results and refresh the interface while handling input events, i.e., these two TCP/IP socket servers are concurrently working together.

4. INTERFACE DESIGN

A good human-computer interface is critical for an application software, especially a web-based application. Because web information is vast and time is valuable people do not want to learn how to operate a web application by manuals but spend their time accomplishing their goals with little or no frustration. Therefore, for a web application software, ease-of-use has become a prime factor in its success.

The interface of our system is simply an HTML web page which a Java applet is anchored in. When a Java-enabled web browser downloads such a document, the Java byte code representing the anchored applet will also be received and executed by the browser. The interface is designed by a user-centered model. We have incorporated several technical features in our interface to enhance its ease-of-use and robust.

4.1. Simplicity

Simplicity is very important to a user-centered multimodal interface, especially to a web-based interface. Its advantages are:

- Reduce the apparent complexity of the interface

- Reduce confusion for a user
- Reduce visual load on a user
- Reduce the interface set-up load on the web

Our interface consists of as few as possible GUI components. Every component has a clear sign to let a user know its function. If a component can be used for more than one function the information panel on the interface will give the user all the necessary navigation information.

4.2. Real-time

The delay of the result behind the input has a significant effect on user acceptance. However, delay cannot be avoided sometimes, especially on the web. Besides reducing time in processing and transferring data, the user's real-time feeling is very important.

First, run-on approach is one of the best solutions. The interface starts sending partial speech data immediately when they are available, the server interprets the data gradually and ships partial recognition result back as soon as it is available.

Second, we try our best to reduce the amount of data transferred through the network. As mentioned in the section 3.2, the interface only send the feature data (about 5% of raw data) to the server.

Third, the interface also needs to respond as soon as possible to any request from a user, such as giving the various navigation information, changing the button color and so on, so as to enhance the user's real-time feeling.

4.3. Navigation

Whenever a user makes a request to the interface he/she will get the navigation information or feedback result immediately. The navigation information shows detailed navigation instructions. The user can simply follow the information to implement his/her goal. For example, if a user uses the "Voice Pen Directory Assistant System" to find a CMU CS person's information, such as telephone number, email address, homepage address, etc. If a user chooses to use speech as the input, he/she can simply speak to the computer by holding down the button "Record/Search." After recording, if the voice input is appropriate, the user will get a feedback "Good recording. Please wait for the recognition result." Otherwise, the user will be asked to input again. If using pen, the user can simply write a name on the draw panel. After writing, he/she will see the information "After writing a name, please click the button "Pen Search" and then wait for the recognition results" so he/she can just follow what the information says.

4.4. Robustness

Robustness is a basic but stringent requirement to all software, especially to the software that works on the web and handles the perceptual inputs because its interface and server must run separately on the different web sites and all its inputs must be gracefully handled. We addressed robust issue as follows:

First, to avoid sending useless data to mess up or crash the server we defined a valid range for the voice input. If a user's voice input is in the range it will be preprocessed and sent to the server. If it is out of the range it will be abandoned and the interface will give some feedback information to the user. For example, if the voice input is not appropriate, the user will see the information "Bad recording. Please speek loudly when you try again."

Second, an interlocking and mutilthread strategy was used to handle the inputs, outputs, and interface event management. A user can interact with the interface while it is processing other events. However, if a process of interpretation, such as the voice recognition, has not been finished the user can't start another one, but the user will be informed on the information panel, such as "Previous task has not been finished yet. If you wants to start a new one, please click button STOP and then try again."

Third, as we mentioned in the previous section, we have developed a server monitor by using the CGI script. Whenever our web-based multimodal demo page is loaded by a user, the server monitor will check all related programs. If any program crashes, the server monitor will restart it automatically. Furthermore, if a server is crashed while a user is interacting with the interface, it will cause an error handling process in the interface. This process will trigger the monitor to restart the server and keep the user informed.

Fourth, a user is allowed to abort or stop any operation. The user might want to abort what he/she has done for some reason, such as waiting too long for all the results because the network or the server is very busy, or he/she has changed his/her mind, etc. If the interface has no such a function, the user has to wait for finishing even though the processing can not be finished for ever because something is wrong in the network or the server. The stop button can abort or stop any processing immediately whenever a user clicks it. If stopping occurs while the speech recognizer is processing the data the next immediate voice processing will have a little delay because the previous speech recognition is still not finished on the server site, but generally the user cannot feel any delay.

5. MULTIMODAL INTERPRETATION

5.1. Multimodal Components

Our labs have implemented recognizers and processing modules for various input modalities, most notably speech, pen input, and face tracking. These modality processors constitute the basic components of all our multimodal systems.

Speech. Our speech recognition subsystem is based on the recognition front-end of the JANUS speech translation system [4] which is capable of processing speaker-independent, spontaneous speech and working on the run-on mode. The recognizer can be adapted to any task domain by retraining the language models and possibly tuning the acoustic models if the domain involves special vocabulary. We also have a high-performance, real-time continuous spelling recognizer for large lists of 100,000 or more names.

Gesture. Our approach to pen-based gesture recognition is to decompose pen strokes into sequences of basic shapes such as line, arc, arrow, circle, cross... [5] The same gesture shape may mean different things depending on the surrounding context, hence each gesture component is augmented by gesture contexts indicating spatial relationships between the gesture and nearby objects in the user interface.

Handwriting. Our MS-TDNN-based handwriting recognizer [6] is capable of processing writer-independent, continuous (cursive) handwriting at a recognition rate of 94% on a 20,000-word vocabulary and working on the run-on mode. We employ simple heuristics to decide when to invoke handwriting recognition on pen input, e.g., when the gesture recognizer cannot identify the input strokes as basic shapes.

5.2. Joint Interpretation

In order to make sense of input from all available sources, we need a multimodal interpreter capable of producing an interpretation of user intent (e.g., a command to execute in the application interface) from the output of the modality processors.

In our joint interpretation scheme, the user intent is represented by a frame consisting of slots specifying pieces of information such as the action to carry out or the parameters for that action. Recognition output from the modality processors are parsed into partially filled frames that are merged together to produce the combined interpretation as described in [5]. This technique leads to uniform handling of high-level information from all input sources, which is very important for modularity and extensibility. To add another input modality we need only provide a module to convert low-level recognizer output to a partially filled frame to be merged with others. In addition, context information can be retained across input events by merging with previous interpretation frames.

6. APPLICATIONS

We have applied our Java front-end to many web-based multimodal interfaces. We describe two examples in this paper: Voice Pen Directory Assistant System and QuickDoc.

Voice Pen Directory Assistant System was designed to find a person's information, such as telephone number, email address, homepage address, using speech and handwriting as inputs. A user can use mouse or pen (if a touch sensitive screen is available) to write the name to look for. The Java front-end will send the query back to the server. The server will perform recognition and database query and send the results back to the user. Alternatively, the user can use speech as input if the computer has the audio input ability. This system has been used for locating people at School of Computer Science in Carnegie Mellon University.

QuickDoc demonstrates a multimodal interface for medical applications [3]. QuickDoc can be used to assist a user in performing a repetitive task with speed and convenience. For example, a doctor can go through a series of images such as X-rays or computer-aided iconography scans, quickly identify an anomalous area, label the area with the name of a disease or condition, and attach relevant comments. The end product is an HTML report that summarizes the doctor's findings in a compact table listing the annotated images, the corresponding preliminary diagnoses, and automatically generated hotlinks to relevant sites based on the diagnoses. QuickDoc is now fully functional in a web environment.

7. CONCLUSION

In this paper, we have presented the designs and applications of our Java front-end for web-based multimodal human-computer interaction. We have also addressed some of the problems that have arisen during its implemention and our solution. We demonstrated that a multimodal human-computer interface can efficiently access, create, manipulate and disseminate multimedia information on the WWW.

8. ACKNOWLEDGEMENTS

This research was sponsored by the DARPA under the Department of the Navy, Naval Research Office under grant number N00014-93-1-0806. Views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Navy or the U.S. Government.

9. REFERENCES

[1] Vo, M.T. and Waibel, A., "A Multimodal Human-Computer Interface: Combination of Speech and Gesture Recognition," *Adjunct Proc. InterCHI'93* (Amsterdam, The Netherlands).

[2] Waibel, A., Vo, M.T., Duchnowski, P., and Manke, S., "Multimodal Interfaces," *Artificial Intelligence Review, Special Volume on Integration of Natural Language and Vision Processing*, McKevitt, P. (Ed.), Vol. 10, Nos. 3-4, 1995.

[3] Waibel, A., Suhm, B., Vo, M.T. Yang, J., "Multimodal interfaces for multimedia information agents," Proc. of ICASSP 97.

[4] Suhm, B., Geutner, P., Kemp, T., Lavie, A., Mayfield, L., McNair, A., Rogina, I., Schultz, T., Sloboda, T., Ward, W., Woszczyna, M., and Waibel, A., "JANUS: Towards Multilingual Spoken Language Translation," Proc. ARPA SLT Workshop 95 (Austin, Texas).

[5] Vo, M.T. and Wood, C., "Building and application framework for speech and pen input integration in multimodal learning interfaces," Proc. ICASSP'96 (Atlanta, GA).

[6] Manke, S., Finke, M., and Waibel, A., "The Use of Dynamic Writing Information in a Connectionist On-Line Cursive Handwriting Recognition System," Advances in Neural Information Processing Systems 6, Morgan Kaufmann, 1994.