STAMP: A SUITE OF TOOLS FOR ANALYZING MULTIMODAL SYSTEM PROCESSING[*]

Joshua Clow and Sharon Oviatt[†]

Center For Human-Computer Communication, Department of Computer Science
Oregon Graduate Institute of Science and Technology

ABSTRACT

In this paper we describe a new automated suite of tools for capturing and analyzing data on multimodal systems called STAMP. STAMP is designed to support research and development efforts for advancing next-generation multimodal systems. STAMP permits researchers to analyze multimodal system performance by: (1) recording data on users' multimodal input and the system's responding, (2) supporting flexible replay of these multimodal commands, along with n-best recognition lists for the individual modalities and their combined multimodal interpretation, and (3) supporting automated analysis using different metrics of multimodal system performance. This collection of tools currently is being used to conduct basic research on the characteristics of multimodal systems, and also to iterate different aspects of the Quickset multimodal architecture.

1. INTRODUCTION

As a new generation of multimodal/media systems begins to define itself, researchers are attempting to learn how to combine different modes into strategically integrated whole systems. In theory, well designed multimodal systems should be able to integrate complementary modalities to yield a synergistic blend— one in which the strengths of each mode are capitalized upon and used to overcome weaknesses in the other. It has been hypothesized that multimodal systems potentially can function more robustly than unimodal ones that are based on a single error-prone recognition technology, such as speech, pen, or vision. However, actually achieving and demonstrating such a performance advantage depends on: (1) the development of effective multimodal systems, (2) the development of novel metrics and empirical evaluation of the performance of multimodal systems, and (3) the development of research infrastructure such as automated tools that support the development of multimodal systems.

The goal of this paper is to describe a new suite of multimodal data analysis tools, which are designed to permit researchers to analyze overall multimodal system performance. This collection of tools, called STAMP, supports the automated analysis of a multimodal system's individual components (e.g., speech recognition, gesture recognition), as well as their capacity to disambiguate one another's meaning when situated within a well optimized multimodal architecture.

2. BACKGROUND

Before describing the specifics of the STAMP analysis suite, some background about the framework in which it operates is necessary. STAMP was designed in conjunction with our multimodal Quickset system (Cohen et al., 1997).

2.1 Quickset

Quickset is a multimodal system that processes spoken and pen-based input. It supports map-based applications ranging from real estate and health-care selection to community rescue operations and military simulations. An example of the Quickset interface for a community fire management and rescue exercise is shown in Figure 1.

Through a combination of spoken and pen-based input that includes pointing, gestures and graphics, Quickset users can add entities to a map, edit or move displayed entities and ask questions about entities and related data. They also can issue commands to control the map's display, filter information in the database, and set up and activate map-based simulation scenarios. For example, a Quickset user could draw an arrow downwards on the map while saying "pan" (i.e., as shown in Figure 1). As further examples, the user could point to a map location and say "jeep" to add a jeep to the map, and then he or she could draw an irregular line in front of the jeep icon and say "jeep follow this route" to specify a route and initiate simulated activity.

In Quickset, the user communicates with a wireless hand-held PC, such as a Fujitsu Stylistic 1200. Quickset integrates spoken and pen-based input with natural language processing and multimodal integration subsystems via a distributed agent architecture (Cohen et al, 1994). To process multimodal input, Quickset uses a joint interpretation strategy based on parallel processing of the spoken and pen-based signals. During this processing, n-best lists are generated for the speech and gesture input signals before and after natural language parsing has been completed. In order to produce a final multimodal interpretation, natural language processing then involves a statistically ranked unification of these spoken and gestural semantic interpretations (Johnston et al., 1997). During this process, a total of five n-best

---

[*] This research was supported in part by Grant No. IRI-9530666 from the National Science Foundation, Grant No. DAT63-95-C-007 from DARPA, and a donation from the Intel Corporation.

[†] Authors: Center for Human-Computer Communication, Department of Computer Science, Oregon Graduate Institute of Science & Technology, P.O. Box 91000, Portland, OR, 97291 (joshc@cse.ogi.edu or oviatt@cse.ogi.edu; http:www.cse.ogi.edu/CHCC/)

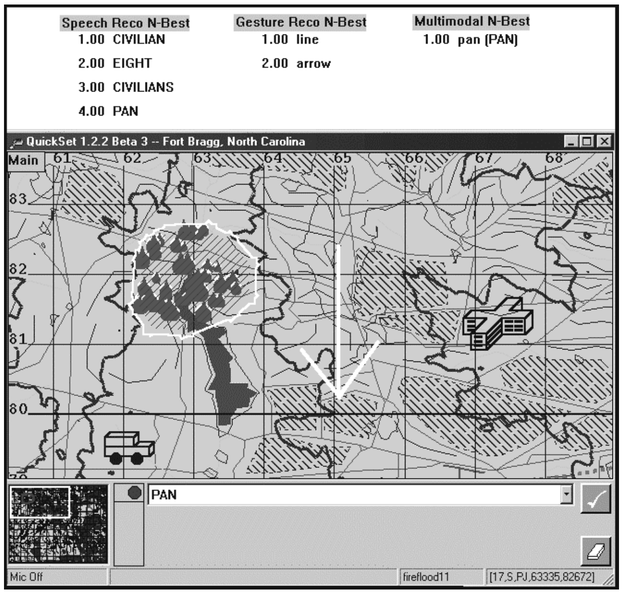
lists are produced for: (1) speech signal recognition, (2) gesture signal recognition, (3) interpretation of parsed spoken language, (4) interpretation of parsed gesture, and (5) final semantic interpretation of the multimodal language.

Quickset's basic functionality, interface design, and empirical specifications derive from research on the user-centered design of multimodal systems for spatial domains (Oviatt, 1997; Oviatt et al, 1997). Further details about Quickset, including its functionality, interface design, language processing, and architecture, have been outlined elsewhere (Cohen et al., 1997).

2.1 Mutual Disambiguation

The robustness of multimodal systems depends on designing an architecture that integrates modes synergistically. In a well designed and optimized multimodal architecture, there can be *mutual disambiguation* of two input signals (Oviatt, in press; Oviatt et al., in submission). For example, if a user says “ditches” but the speech recognizer confirms the singular “ditch” as its best guess, then parallel recognition of several graphic marks could result in recovery of the correct plural interpretation. This recovery can occur in a multimodal architecture even though the speech recognizer initially ranked the plural interpretation “ditches” as a less preferred choice on its n-best list.

Figure 1 illustrates an example of mutual disambiguation from a Quickset user's log (Oviatt et al., in submission). In this case, the user said “pan” and drew an arrow. Although the lexical item “pan” was only ranked fourth on the speech n-best list, and the arrow was ranked second on the gesture list, the correct semantic interpretation was recovered successfully (i.e., ranked first) on the final multimodal n-best list. This recovery was achievable because inappropriate signal pieces are discarded during the unification process, which imposes semantic, temporal, and other constraints on legal multimodal commands.



Speech Reco N-Best	Gesture Reco N-Best	Multimodal N-Best
1.00 CIVILIAN	1.00 line	1.00 pan (PAN)
2.00 EIGHT	2.00 arrow	
3.00 CIVILIANS		
4.00 PAN		

QuickSet 1.2.2 Beta 3 -- Fort Bragg, North Carolina

PAN

Mic Off fireflood11 [17,S,PJ,63335,82672]

Figure 1: Mutual disambiguation of speech and gesture signals during a user's command to “pan” the Quickset map during a community fire management exercise

Due to mutual disambiguation, the parallel recognition and semantic interpretation that occurs in a multimodal architecture can yield a higher likelihood of correct interpretation than recognition based on either single input mode (Oviatt et al., in submission). This improvement is a direct result of the disambiguation between signals that can occur in a well designed multimodal system, which exhibits greater performance stability and overall robustness as a result.

3. SYSTEM ARCHITECTURE

The STAMP suite consists of four separate pieces: a data logger, a loader, a marking/analysis tool and a video controller.

3.1 Data Logger

In a typical data collection session with Quickset, the user issues multimodal commands to the system during a task. The user's pen input is recorded directly from the screen in the context of the Quickset interface's map, and simultaneous speech input is recorded onto the same videotape. In addition to this videotape record, the results of system processing are logged at the levels of signal processing, parse interpretation, and final multimodal command interpretation.

The data logger itself has been created as an agent within the Quickset system. It is written in Prolog, and it effectively subscribes to and saves all messages containing data events of interest to a text file as the session runs. Initially, no attempt is made to extract specific pieces of data or to format the output collected, partly to expedite speed of data capture and also to ensure the availability of a complete backup record. However, this raw log file typically would total hundreds of pages for a one-hour session if printed, and it also would be extremely difficult to interpret in its original format. Even with reformatting and data extraction, this raw log file alone is not useful by itself for analyzing multimodal system performance, because it does not include any record of the user's actual language input and performance.

3.2 Loader

Once the data session is complete, the set of raw log files then needs to be processed for analysis. A Perl script turns the logged agent messages into a text-delimited database format, which then is loaded into a database created with Microsoft Access. This database contains user information, temporal information, the relevant n-best lists associated with each multimodal command, and the relations between these pieces of information.

3.3 Marking/Analysis Tool

The third component is a researcher's analysis tool, which is written in C++. The marking/analysis tool uses the database generated by the loader to present the data in a manageable way. In particular, it supports the researcher as he or she reviews and annotates the system's output for a given command (i.e., n-best lists) by synchronizing it with the corresponding videotaped record of the user's input to the system. Figure 2 illustrates the researcher's STAMP setup for analyzing a multimodal command, with side-by-side screens displaying the system's command interpretation as n-best lists (left side) and the corresponding videotape of a user's original input on the map interface (right side).

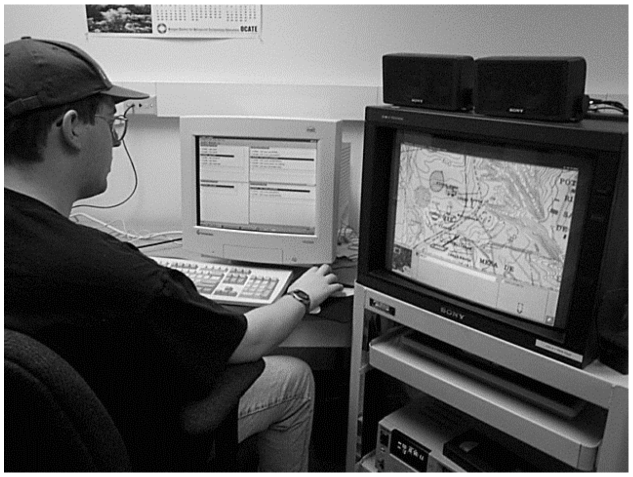
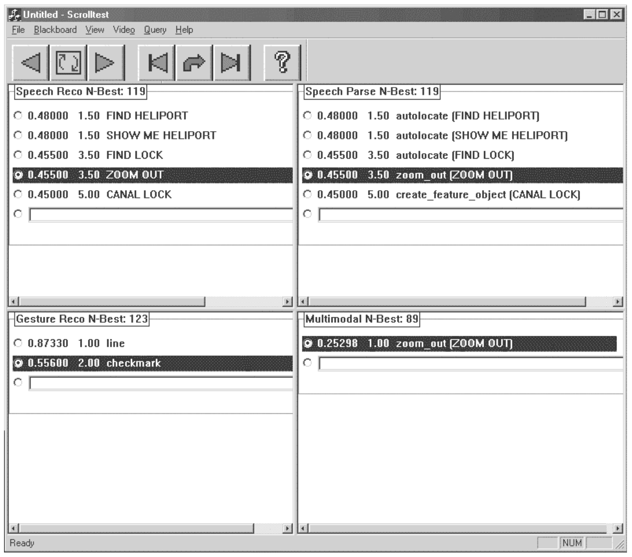


Figure 2: Researcher's setup for analyzing multimodal system processing

The analyzer's main screen is split into four or five panels, as illustrated in Figure 3. These panels correspond to the n-best lists for speech signal recognition, spoken language parse, pen gesture signal recognition, gesture parse, and the final multimodal interpretation after signal integration and natural language processing have taken place. Each panel displays the relevant n-best list of lexical interpretations in rank order according to probability estimates. As the researcher reviews the video record for any given command, he or she can mark which lexical content actually was uttered by the user and therefore is the "correct" one, which may not be the same as the first item on the system's n-best list, or even present on the list at all. In the example shown in Figure 3, the user actually spoke "zoom out" and drew a checkmark, so the researcher has marked these fourth and second ranked items on the speech and gesture lists. A separate text field also permits the entry of missing correct responses, and can be used to note any user or system performance irregularities that would influence scorability of the command.

This researcher's record of annotated data then is entered into the database and is available for analysis. By comparing the system's interpretation with the researcher's scored record, it is possible to evaluate the system's performance for the different input modalities as well as different levels of language processing (i.e., unimodal signal recognition, unimodal parse interpretation, multimodal interpretation).

In the course of analyzing multimodal commands from a session, STAMP also provides flexible navigation controls. It automatically determines what section of the videotape coordinates with the system's logged record of n-best lists, and advances the videotape to the correct location. This enables the researcher to avoid manual searching, and to verify very quickly and accurately (1) what the user actually gestured and said, (2) how these signals were formed, (3) whether any human performance errors were evident, and (4) what technical problems may have been present (e.g., ink skipping). STAMP also permits automatic location of the session's start, end, or next logged event— where the "next" event can be defined as all user input in sequence, only successful speech or gesture recognition events, only successful multimodal system integrations, or other researcher-defined types of outcomes. STAMP also generates automated summaries of recognition and mutual disambiguation rates averaged over subjects, conditions, or a whole corpus.



Speech Reco N-Best: 119

```
○ 0.48000  1.50  FIND HELIPORT
○ 0.48000  1.50  SHOW ME HELIPORT
○ 0.45500  3.50  FIND LOCK
● 0.45500  3.50  ZOOM OUT
○ 0.45000  5.00  CANAL LOCK
```

Speech Parse N-Best: 119

```
○ 0.48000  1.50  autolocate (FIND HELIPORT)
○ 0.48000  1.50  autolocate (SHOW ME HELIPORT)
○ 0.45500  3.50  autolocate (FIND LOCK)
● 0.45500  3.50  zoom_out (ZOOM OUT)
○ 0.45000  5.00  create_feature_object (CANAL LOCK)
```

Gesture Reco N-Best: 123

```
○ 0.87330  1.00  line
● 0.55600  2.00  checkmark
```

Multimodal N-Best: 89

```
● 0.25298  1.00  zoom_out (ZOOM OUT)
```

Figure 3: STAMP's summary of system processing for a multimodal command represented as a collection of n-best lists for speech signal recognition, spoken language parse, gesture signal recognition, gesture parse, and multimodal interpretation

Except for video control, which requires an additional piece of hardware as described below, all pieces of the above-described multimodal analysis tool run on a standard PC using Windows 95 or NT.

3.4 Video Control

The video controller component consists of a software agent written in C, as well as a piece of hardware (i.e., V-LAN Express unit) that is used to issue commands to the VCR. With an agent controlling the hardware, it is possible to use the same software for both data collection and analysis purposes. During data capture, the video agent resides on the subject's machine, and it periodically requests SMPTE timecode from the VCR, which it associates with the computer's clock time. These data then are logged. This technique is similar to that used in other computer-driven VCR systems described in usability labs (Weiler, 1993). During analysis, the video agent accepts commands from the analysis tool and issues its own commands to the hardware directly, which in turn plays the videotape.

In our current STAMP setup, researcher's annotations are performed with the analysis tool running on a computer next to the VCR's monitor in order to support high-resolution displays of all the critical information. However, depending on STAMP's application needs, a TV card could be used so the VCR's display is routed to the same screen as the analysis tool.

4. DISCUSSION

The described tools for analyzing multimodal system performance represent the kind of infrastructure that will be needed to support research and development on a new generation of multimodal systems. STAMP currently is being used to conduct research on the general characteristics of multimodal systems, and to optimize the Quickset multimodal system's architecture. It provides the thorough diagnostic information that is needed to iterate a multimodal system's design in an informed way.

One of the general benefits of using the described STAMP architecture is flexibility and extensibility. Adding data to the analyzer involves creating a new panel, associating the new panel to the appropriate table(s) in the database, and then ensuring that the table gets filled by the loader. The format and presentation of the other panels need not change. In addition, by using a proper database, many additions or modifications to the system can be done by non-programmers. Reports and statistics can be generated from within Access, and new browsing criteria can be defined. Although STAMP has been developed for use with Quickset and its different data sets, the techniques described above are not difficult to implement and could be applied to other types of multimodal systems and to the analysis of other metrics.

A second version of STAMP is being designed and built to extend its current repertoire of summary metrics, to streamline the layout of control panels, and to increase the ability to capture rich data in a flexible way through enhancement of annotation features, to track irregularities involving system performance and errors, and so forth. In the future, STAMP also will expand its data search capabilities by permitting the researcher to specify more complex browsing criteria. In addition, it will incorporate a bookmarking capability that would be useful for constructing highlights videos, and for tagging and organizing unusual or unexpected multimodal events.

5. REFERENCES

1. Cohen, P., Johnston, M., McGee, D., Oviatt, S., Pittman, J., Smith, I., Chen, L. and Clow, J. Quickset: Multimodal interaction for distributed applications, Proceedings of the Fifth ACM International Multimedia Conference, New York, NY: ACM Press, New York, 1997, 31-40.

2. Cohen, P. R., Cheyer, A., Wang, M. & Baeg, S. C. An open agent architecture, *Proceedings of the AAAI Spring Symposium Series on Software Agents*, Stanford University, Stanford, Ca., 1994, pp. 1-8.

3. Johnston, M., Cohen, P., McGee, D., Oviatt, S., Pittman, J., and Smith, I. Unification-based multimodal integration, Proceedings of the ACL'97 Conference, Association for Computational Linguistics, San Francisco, CA.: Morgan Kaufmann, 1997, 281-288.

4. Oviatt, S.L. Ten myths of multimodal interaction, Communications of the ACM, in press.

5. Oviatt, S.L. Multimodal interactive maps: Designing for human performance, Human-Computer Interaction 1997,12, 93-129 (special issue on ``Multimodal Interfaces").

6. Oviatt, S. L., DeAngeli, A. and Kuhn, K. Integration and synchronization of input modes during multimodal human computer interaction, in Proceedings of Conference on Human Factors in Computing Systems: CHI '97, New York, N.Y.: ACM Press, 415-422.

7. Oviatt, S.L., McGee, D., Clow, J., Cohen, P. and Johnston, M. Robust functioning through mutual disambiguation in a multimodal system architecture, in submission.

8. Weiler, P. Software for the Usability Lab: A Sampling of Current Tools (panel), in Proceedings of Conference on Human Factors in Computing Systems: CHI '93, New York, N.Y.: ACM Press, 57-60.