MICoT: A Tool for Multimodal Input Data Collection

Raymond H. Lee, Anurag Gupta
Motorola Labs
1301 E. Algonquin Road, Schaumburg, IL 60196, USA
{Raymond.H.Lee, Anurag.Gupta}@motorola.com

Abstract
In this paper, a multi-modal data collection tool called MICoT is described. We highlight the various design and implementation aspects that we consider to be important for MICoT. An example is given to illustrate the application of the tool to collect data for our research in multi-modal dialog system.

1. Introduction
Significant advances have been made in the area of spoken dialog system research and applications, e.g. [4] and [9]. This has been achieved with the availability of well-designed and collected speech corpus [5]. Such databases allow the evaluation and testing of new algorithms and their benchmarking in a standard and measurable manner. As we move towards devices and applications with multi-modal user interfaces [7], databases of multi-modal inputs and outputs are needed.

One of the challenges we faced when we began our research in multi-modal user interface technology is the lack of multi-modal input data for testing and evaluation of our work. This led to the development of MICoT, a software tool for multi-modal input data collection. This paper provides a detailed description of MICoT. By highlighting the various aspects of MICoT that we consider to be important and by providing a concrete use case, we hope to facilitate further discussion and effort in developing standard multi-modal databases and associated tools that will ultimately support the research and development of highly useable multi-modal systems.

The reminder of this paper is organized as follows: The next section describes MICoT including its design objectives and features. An example is also given to illustrate how it is used to collect multi-modal data for an application. Related works are described in Section 3 followed by discussions in Section 4.

2. Multi-modal input collection tool (MICoT)
The high level goal of MICoT is:

“To provide a generic tool to simulate an application and to collect user inputs from different modalities. Also the tool has to support both simple multi-modal command-and-control setting as well as multi-modal dialogs where a sequence of user turns needs to be collected.”

In order to achieve the above goal, the following design objectives and features have been developed:

1. The tool has to support multiple applications. In MICoT, applications are defined declaratively using eXtensible Markup Language (XML). A schema has been designed to allow developer to specify the task sequences of the application and for each task, the application’s outputs by different modalities such as speech, text and GUI.

2. The data collected has to be portable. MICoT captures user input for each task and can generate the information collected in XML or CSV (Comma Separated Value) format. The following user data is collected for each modality:
   - The time at which the user input starts.
   - The time at which the user input ends.
   - The modality used to provide the input.
   - The task visible on the application’s GUI.
   - Raw modality data such as speech waveforms.

3. The tool has to be extensible to different modality engines and new modalities. MICoT currently supports speech, text and stylus inputs. The standard Java Speech API (JSAPI) [3] has been chosen as the interface between the speech engines (speech recognition and text-to-speech) and MICoT. We intend to apply the same strategy when adding new modality support to MICoT in the future.

4. The tool should be able to run on different platforms. MICoT is developed in Java and can run on PC platforms (Win32 and Linux), as well as handheld devices such as iPAQ.

2.1. Implementation details
MICoT is implemented in Java and currently supports speech recognition, text and touch (mouse or stylus) inputs and text-to-speech, text and graphic outputs.

Figure 1 shows a screenshot of MICoT running a map based navigation application. The menu allows the loading of different application tasks declared in MICoT. Images of an application are displayed at MICoT’s window GUI to provide users with the look-and-feel of the application in operation. A text box is provided at the bottom for displaying text inputs and outputs.

In order to simulate an application in MICoT, its tasks (i.e., what a user can do with the application) have to be first declared in MICoT using the XML DTD schema shown in Figure 2. A multi-modal task sequence consists of one or more tasks. Each task is assigned a unique id and contains the instruction for the task in the instruction attribute. MICoT currently supports both SpeechPrompt and TextPrompt, for speech and text output using text-to-speech

Figure 2. A multi-modal task sequence consists of one or more tasks. Each task is assigned a unique id and contains the instruction for the task in the instruction attribute. MICoT currently supports both SpeechPrompt and TextPrompt, for speech and text output using text-to-speech
and text box GUI respectively, as defined in the Prompts element. The application’s GUI display is declared in the Image element, which defines the image file to display, the type of view to use (e.g., scrolling windows) and the coordinates of the image to show.

Figure 1: Screenshot of MICOt for a map based navigation application.

A JSAPI interface is provided by MICOt to connect speech recognition engines. A standard API is chosen here to improve the portability of MICOt to different speech recognition engines. This is useful to collect data and to compare performance of an application using different speech recognition engines, for example.

MICOt can also be connected to the voice input directly through JavaSound API (a standard Java API) without a speech recognition engine installed. A module is implemented in MICOt to emulate the behavior of a speech recognition engine. Currently, a time delay can be specified in the module to mimic the latency of the speech recognition process. This feature is useful in collecting data to study the effect of speech recognition latency, typically occurring on mobile devices with limited computational power, on the overall multi-modal processing.

<?Xml version="1.0"?>
<?ELEMENT MMTaskSequence (Task+)>
<?ELEMENT Task (Prompts, Image)>
<?ELEMENT Prompts EMPTY>
<?ELEMENT Image EMPTY>
<?ATTLIST Task id ID #REQUIRED>
<?ATTLIST Prompts TextPrompt CDATA #IMPLIED>
<?ATTLIST SpeechPrompt CDATA #IMPLIED>
<?ATTLIST Image file CDATA ViewType (scroll | no-scroll) OriginX CDATA OriginY CDATA #REQUIRED>

Figure 2: DTD schema for declaring tasks in MICOt.

2.2. Example

We apply MICOt to collect multi-modal user input data for two applications:

1. A drawing application where a user draws different geometric objects on the computer screen. For example, a user can say “Draw a square here” and drags over a marked area.

2. A map based navigation application where a user gets location specific information on a map. For example, a user can say “I want to go to Circular Quay from here” while pointing at a location on map.

In this section, we provide a detailed description of MICOt by illustrating how it is applied to collect user input data for the map based navigation application. The map based navigation application is implemented in Java and runs on handheld devices such as iPAQ PDAs. It is designed to take advantage of a variety of input and output modalities to create a natural user interface for handheld devices in mobile environments.

<table>
<thead>
<tr>
<th>Task name</th>
<th>Description</th>
<th>Example utterance</th>
</tr>
</thead>
<tbody>
<tr>
<td>FindTrainStation</td>
<td>User requests locations of train station(s).</td>
<td>“Where is the nearest train station from here[point to map]?”</td>
</tr>
<tr>
<td>TrainSchedule</td>
<td>User requests train schedule information</td>
<td>“What’s the next train from here [point to train station] to Sydney?”</td>
</tr>
<tr>
<td>FindRestaurant</td>
<td>User requests locations of restaurants matching search criteria</td>
<td>“Show me the Italian restaurants around here[circle area on map].”</td>
</tr>
<tr>
<td>FindAtm</td>
<td>User requests location of automatic teller machines (ATMs)</td>
<td>“Commonwealth bank ATMs”</td>
</tr>
<tr>
<td>FindRoute</td>
<td>User requests route to destination on map</td>
<td>“I want to go to this ATMs [point to map] and then there [point to map].”</td>
</tr>
</tbody>
</table>
Table 1 summarizes the main tasks implemented in the map based navigation application and Figure 3 shows a task sequence declaration in MICOt. The task sequence emulates a multi-modal dialog where the user finds the location of the nearest train station on map and the next train from that station to a desired destination. A possible instance of the dialog is as follows:

S1: “What would you like to do?”
[Text: What would you like to do]

U2: “I want to go to the nearest train station.”
[Text: What would you like to do]

S3: “The nearest train station is St. James St. What would you like to do?”
[Text: What would you like to do]

U4: “Next train to here [click on Circular Quay]”

S5: “The next train leaves in 10 minutes. Would you like to catch it?”
[Text: Catch next train.]

U6: “OK”

2.2.1. Data collection

The data collected by MICOt can be saved in either XML or CSV formats for further processing and analysis. Raw data files (i.e. waveform files for speech and coordinate points for touch) are also saved. A meta-data XML file is also created for each user session. It contains the information on the user’s identity, hardware platform used to collect the data and the modalities used to complete each turn and the task associated with the turn.

Figure 4 shows part of the data collected for a single turn of multimodal interaction using MICOt. The data shown covers the first task (id="Nav-1a") in the task sequence for the map based navigation application (Figure 3). MICOt’s data logging mechanism creates a MMTurn element, which describes the output provided to the user at the beginning of the turn and the subsequent multimodal inputs received within the turn, for each user turn. The timing of the spoken prompts provided by MICOt and the user’s visual context at the start of the turn is captured within the OutputState element. For example, the Visual element describes the image being displayed and the coordinates of the top left-hand corner of the scrolled image. Input in each modality is captured within the MMInput element, which describes the modality used, the times at which user input starts and ends, and name of the file containing the raw data received in the modality, e.g., each gesture input is recorded in a file (e.g., point0.xml stores gesture data in Ink Markup Language [2]) whose name is the value of the Data element.

```
<MMTaskSequence>
  <Task id="Nav-1a" Intention="Your task here, is to determine the nearest train station">
    <Prompts TextPrompt="What would you like to do?" SpeechPrompt="What would you like to do?" />
    <Image file="scenario_1a.gif" ViewType="Scroll" OriginX="45" OriginY="190" />
  </Task>
  <Task id="Nav-1b" Intention="Your task here is to find out when the next train to Circular Quay is leaving">
    <Prompts TextPrompt="The nearest train station is St. James St.. What would you like to do?" SpeechPrompt="The nearest train station is St. James St.. What would you like to do?" />
    <Image file="scenario_3a.gif" ViewType="Scroll" OriginX="0" OriginY="140" />
  </Task>
  <Task ID="Nav-1c" Intention="Your task here is to inform the system that you would like to catch this train">
    <Prompts TextPrompt="Catch next train?" SpeechPrompt="The next train leaves in 10 minutes. Would you like to catch it?" />
    <Image file="scenario_3b.gif" ViewType="Scrolling" OriginX="0" OriginY="140" />
  </Task>
</MMTaskSequence>
```

**Figure 3:** XML declaration of a map based navigation application task sequence

Figure 4: Snippet of data collected for navigation task

```
<MMTurn Start="0" End="1281" Duration="1281" TaskID="Nav-1a">
  <OutputState>
    <Visual file="scenario_1a.gif" X="45" Y="190" />
    <Speech Start="0" End="350" />
  </OutputState>
  <MMInput Modality="SPEECH" Start="130" End="190" Data="Data0.wav" />
  <MMInput Modality="GESTURE" Start="130" End="190" Data="point0.xml" />
</MMTurn>
```
3. Related Work

Clow and Oviatt [1] describe STAMP, a suite of tools for collecting and analyzing multi-modal data. MICoT performs the data collection functions similar to those provided by STAMP’s data logger and loader. Instead of an end-to-end tool, we focus on using standard modality APIs and XML to ensure MICoT’s portability to different applications and modalities. The XML data files collected, together with the DTD schemas defined, can be used by other data processing and analysis tools such as STAMP’s marking and analysis tools. MICoT can also generate its data in CSV format for processing without any programming, e.g., by a database program.

Life et. al. [5] describes a two stream strategy to collect data for the UI design and spoken language corpus required in the development of the Multimodal-Multimedia Automated Service Kiosk (MASK) project. High fidelity Wizard of Oz simulation was used for developing the UI while a working prototype is used for speech data collection. One drawback of this approach is the need to develop two data collection systems.

Our approach to multi-modal data collection is similar to that proposed by Rupp and Strube [8], where an iterative approach is used to collect multi-modal data for multi-modal dialog systems. The concept of ‘script’ experiment was introduced in which a subject is given a script with task description, with each task corresponding to a dialog step. The script does not constrain the modality to be used or the words to be said. MICoT is designed mainly for script experiment, as shown by the example in Section 2.2. The level of granularity in the script can be varied in MICoT to facilitate the iterative data collection approach. For example, a static image (in our case, a map) is declared in the task sequence XML file to emulate the display of an application in the early stage of its development. At a later stage, the “real” GUI can be displayed when that part of the application has been developed.

4. Discussions

This paper outlines the design and implementation aspects of MICoT that we consider to be important for a tool for multi-modal data collection. MICoT allows the declaration of task sequence of an application without the need to develop the application itself so that multi-modal interaction data can be collected prior to considerable effort in application development. Also, UI and interaction designers can rapidly prototype their ideas by writing configuration files in XML to simulate different scenarios. As the data collected is stored in XML files, they can be easily imported by other tools or databases for analysis.

We are currently using MICoT to collect multi-modal input data to study the characteristics of multi-modal interaction in the mobile environment. Preliminary analysis of the data collected is being conducted to statistically model input modality integration patterns.

5. Acknowledgements

The authors would like to acknowledge Peter Robbins from the Multimodal Language Technology Labs within Motorola Australian Research Centre, Sydney, for his work in the development of MICoT.

6. References

[2] Ink Markup Language (http://www.w3.org/TR/InkML/)