Multimodal Interfaces

Remotroid

Siavash Bigdeli / Christian Lutz

University of Neuchatel and University of Fribourg

1. June 2012

Table of contents

1 Introduction

One tasks of the lecture “Multimodal Interfaces” was to design and implement a multimodal project. Multimodality means that the project uses different modalities like gesture, speech touch and others as input.

After having searched for a longer time for a useful project we decided to realize “Remotroid”. This application allows controlling the mouse pointer of a laptop with a smart phone.

This report presents the different development steps and the main points of it. In a second part a user evaluation was done and will be discussed.

2 Idea of the application

On most presentations at university and other places there are the same issues. The lecturer speaks from a certain place. If he wants to go to the next slide of his presentation or open a link, he has to do some manipulations on his laptop. He has to move to his laptop, change the slide and go back to his initial place. Sometimes the lecturer is just sitting behind the laptop during the whole presentation. For listener this is the worst case.

There are some devices for going to next/previous slide. Also with them it's not possible to open a file or a link during the presentation. Furthermore the lecturer has to take one more device with him. There is a better solution for this issues. It’s called “Remotroid” and was created for this project.

Remotroid is an android application which allows to control the mouse pointer of a laptop with tilting movements on phone side. For more precise movements the mouse pointer could also be controlled over the touchscreen of the smart phone. Supplementary several commands like copy/past, save etc. could be executed with speech commands. To execute clicks volume button and touchscreen on phone side are used. Remotroid has also an integrated speech recognizer. Its possible to write spoken words and sentences on a text area on computer-side. Remotroid can fully replace the mouse. Also keyboard can be replaced to a certain point even it’s not possible to write longer texts with this application.

3 Device and modalities

3.1 Device

Since the application is written to control a computer, a computer is required. The connection between computer and smart phone is realized over Bluetooth. On computer-side either an internal Bluetooth-card or an external Bluetooth-adapter can be used. Remotroid runs under Windows, Apple and Linux operation systems.

To prevent lecturers to bring one moor device to their presentations “Remotroid” was created for any android device, in special for smart phones. For our project we used the Google phone “Nexus one”. All of the newer smart phones are able to detect tilting movements with the integrated gyroscope and accelerometers. Whit those sensors tilting movements to left/right, up/down and also rotations could be detected. For this application only up/down and left/right movements are used.

The touchscreen can detect movements on the x and y axes.

All collected date is send over Bluetooth to the connected laptop. Due to the fact that since now it's not possible to connect an unrooted android device as a HID (Human Interface Device) to the laptop, a little server on computer side listens for commands. The revived commands are then treated by this server and executed.

3.2 Speech Recognition

Executing every command with clicking can be very painful. That's the reason we introduced speech commands to execute some common commands like “open”, “close”, “copy/paste” etc. For this part of Remotroid we used Google Speech recognition API. The speech recognition is done on phone side.

It is also possible to deactivate the speech commands. In this case speech recognition works as a keyboard replacement. The recognized word or sentence is written if any text area on computer-side is selected. This gives the possibility to the lecturer to enter an address on a browser or some words in a text document.

Even it’s possible to recognize different languages, the best results are obtained with English spoken text. This is the reason why all spoken commands, words and sentences must be in English.

4 CASE and CARE

Because this application has been written for the Multimodal course it must be classified on the CASE and CARE model seen during the lectures

4.1 CARE classification

To classify our System from the user-side we need the CARE model. Remotroid uses three of the four properties of the CARE model.

•Equivalence: Moving the mouse pointer can be done by tilting either the phone or over the touchscreen of the smart phone.

The user can run commands by clicking them in the menus. He has also the possibility the run the commands with speech input.

The user can do all commands with normal mouse and keyboard. But this is of course not the coal of the application.

•Complementarity: To move items on the desktop, the user must press the volume-up button and move the mouse pointer with the two approaches described before.

The same is true for some commands which are executed on specific items like copy a file etc. The user has to select the item and execute a command either with speech or right-click.

•Assignment: Opening the context menu can only be done with the volume-down button from the phone.

4.2 CASE classification

To classify our System from the machine-side we need the CASE model. Remotroid uses three of the four properties of the CASE model.

•Concurrent: The user can execute 2 different commands in parallel. He can move the mouse pointer and at same time execute a global command over speech input like “play music”, “next song” etc.

•Synergistic: To move an icon on the desktop the volume-up button has to be pressed during the whole action.

•Exclusive: Right-click can only be done with the volume-down button of the phone.

4.3 Fusion

The fusion of the different inputs is done on server-side. Since the smart phone writes all captured commands in a Buffer the commands could just read one by one on server-side. It is not necessary to decide which command has to be executed first because there are still in a sequential order.

4.4 Fission

Every mouse movement can be observed directly on screen. No other feedback is needed for this kind of inputs. Only for speech commands an additional feedback is given. The executed command is written to a text box on the top of the smart phone screen.

5 System Architecture/ Realization

Since the application is written for android devices it's written in java. The little server on computer side is also written in Java. To develop the application we used Eclipse as development toolset. For the Android development we used the Android SDK plug-in for Eclipse. It comes with a virtual Android device for test purpose.

5.1 Initialization

During the initialization part the server on computer-side has to be started. A little window appears on the desktop. After that a Bluetooth connection has to be established between the smart phone and the computer. Now the connected computer has to be selected in the smart phone menu. Once this is done the initialization is completed. No calibration needs to be done on phone-side.

5.2 "Runtime"

Once the initialization is complete the smart phone begins to send the collected date into the server-side buffer. Different listeners on phone-side are collecting the inputs from accelerometers, touchscreen and speech engine. Every command is then send to the server as a package of 4 Bytes. The first two Bytes are used to specify the command. The other two Bytes are used to send the X and Y coordinates of movements. Even there are commands which do not need coordinates they also contain 4 Bytes. This makes it easier to read the commands out of the Buffer on server-side.

```
//Function handles the volume button
public boolean onKeyDown(int keyCode, KeyEvent event) {
      if (keyCode == KeyEvent.KEYCODE_VOLUME_UP) {
            //Send command for press left-lick
            mCommandService.sendData(4, 0, 0);
            return true
      } else if (keyCode == KeyEvent.KEYCODE_VOLUME_DOWN) {
            //send command for right-clic
            mCommandService.sendData(3, 0, 0);

      return super.onKeyDown(keyCode, event);
}
```

If the command was done over speech it will also be written in the textbox on top of the smart phone screen. That gives the possibility to the lecturer to check if a command was recognized or not.

The little server on computer-side reads 4 Bytes at once from the buffer. This is done in an endless loop. As soon as a command (4 Bytes) is read by the server it will be divided into the command part and x/y coordinates. After that the commands are executed using the Keyboard.java class. This class translate the commands in mouse movements and keyboard shortcuts. Those commands and movements are then executed by the Robot.java class.

```
while (true) {

                        /* First byte is the type of the command */
                        type = inputStream.read();

                        X_FINAL = 0;
                        Y_FINAL = 0;

                        /* First Byte of Y-Coordinate */
                        int command = inputStream.read();

                        Y_FINAL += command;

                        /* Second Byte of Y-Coordinate */
                        command = inputStream.read();
                        Y_FINAL += (command << 8);

                        /* First Byte of X-Coordinate */
                        command = inputStream.read();
                        X_FINAL += command;

                        /* Second Byte of X-Coordinate */
                        command = inputStream.read();
                        X_FINAL += (command << 8);

                        if (command == EXIT_CMD) {
                              System.out.println("finish process");
                              break;
                        }

                        processCommand(type, X_FINAL, Y_FINAL);
                  }
```

5.2.1 Movements

If a mouse movement command is read from the buffer the x/y coordinates declare how much the pointer has to move in the two directions. If it's a tilting movement the pointer is not moved directly. First the x/y coordinates are passed to a correction function which modifies them with an exponential scale. Big movements would be increased and smaller ones reduced. This makes it easier to control the mouse pointer with tilting movements.

```
private int convertMovement(int x) {
        int result;

        if (x < 0) {
            x *= -1;

            result = -1 * (int) (Math.pow(1.09, x));

        } else
            result = (int) (Math.pow(1.09, x));

        return result;
    }
```

6 User Evaluation

We generated a test to evaluate which combination of modalities perform better.

6.1 *Task*

We used Google earth software to simulate a search task. The participants were asked to find locations with similar difficulties (An airport in New York, Train station in Fribourg, Tokyo, Champs Elisées in Paris). Our application were used to perform zooming by replacing the mouse wheel and padding the map by replacing the keyboard arrow keys.

6.2 *Modalities*

First model: Device accelerometer for zooming and touch screen for padding.

Second model: Device accelerometer for padding and touch screen for zooming.

6.3 Participants

We asked 8 participants aged between 20 and 30 to test our application and they all used a smartphone and Google earth before.

6.4 Environment

Test took place in an empty classroom. A projection was used to make sure participants feel the concept of the application. And a Nexus 1 was used for all test during the evaluation.

6.5 Procedure

For each participants, we asked first to locate their home in the map to understand the concept of the model. After that they were asked to locate two of the predefined places. After we switched the application model, we again asked them to locate their home before two other places. And between the participants, the ordering of tasks was balanced. After the search test, the participants were asked to perform a predefined sequence of desktop manipulation tasks to evaluate the overall usability of the application. Finally they were asked to fill an evaluation form.

6.6 Error measurement

We defined the error by the amount of unnecessary user interaction with the device regarding to necessary ones. For example if user made three passes to the right and one pass to the left, we say there was a an error with the value of 1/2 for this task.

6.7 Results

As the result of the evaluation form, Figure 1 shows the average answerer of participants for these four questions:

Q1 : How much do you think the concept of this application makes sense?

Q2 : How much it was easy to interact with touch and gesture?

Q3 : How much are the vice commands useful?

Q4 : General impression of the application.

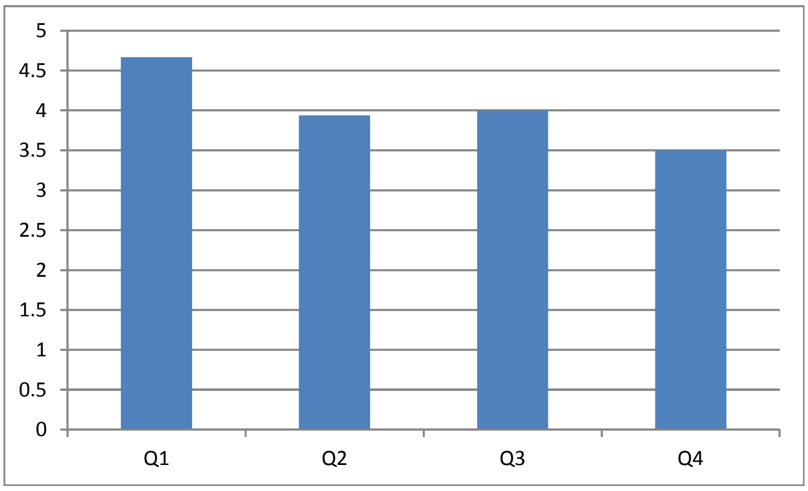

Figure 1 : Average of question responses

As shown in Figure 1, the concept of the application was acceptable by the participants (Q1), however, value of Q4 shows that this implementation could be improved (mostly lags in our system exaggerated the distraction). And they found the modalities useful and easy to interact.

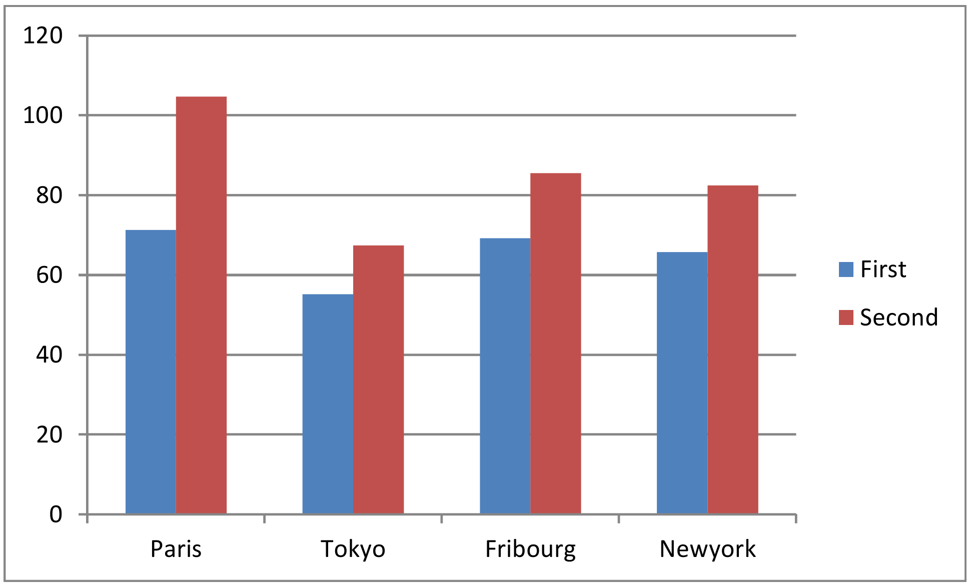

Figure 2 : Mean time for each city

In the case of statistical results, Figure 1 shows the mean average time that took participants to locate the target with each model. In all targets first model performed faster than the second model. However we could not find a significant difference between these two models in case of time.

As the number of interaction with the device, Table 2 shows the mean average of each city for each method. By the results of a t-test of one-tailed distribution and paired data over number of interaction for each city with 99% precision we could conclude that ***"First model needed much less of interaction than the second one for the proposed targets"***.

Table 1 : significant difference for each city

	Mean number of interactions	
	First	**Second**
Paris	468.25	3674.25
Tokyo	453.5	3107
Fribourg	490	3278.75
Newyork	424.25	2619.75
Avg.	**459**	**1471.3**

We also did a t-test with the same properties to check whether or not the above assumption stands for all test. The resulted value was 2.3E-04 (The significant difference between mean results of each model in table 1), which indicates this general fact that ***"First model needed much less of interaction than the second one"***.

We found the proposed error measurement system not useful. Figure 3 shows the computed error of each participant. In most of the cases, there is no significant difference between two model. The peak on this chart belong to a test for First model with Newyork. We should mention that the same participant did a similar result for the other tasks.

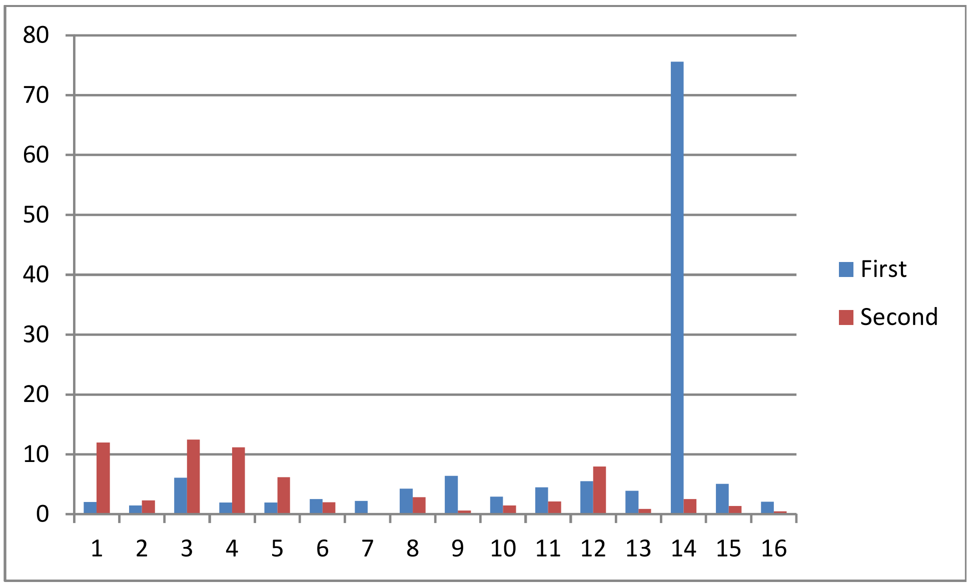

Figure 3 : Computed error of each test

7 Conclusion

In this project we aimed to replace traditional computer interfaces like mouse and keyboard with a portable, more interactive device. The goal was to provide a device that carries out the user-computer interaction and also to let users to have distance or move in the environment while using their computers. As the portability factor, we have chosen the smartphone as our interface which have a lot of input sensors and also is very popular between users.

We developed a multimodal interaction system for Android devices which can handle most of the user-computer interactions. Using device's accelerometer to control the mouse lets users to quickly navigate the cursor. Also the alternative touchscreen modality is used to add precision to the mouse navigation. We also used speech commands to replace keyboard shortcuts (e.g. copy, paste). The speech system can also be used for text entry like a URL address or an email.

A study has been done to check the usability of our application. All participants agreed that such a system is very useful and they rated our implementation to be very close to the desired system. A simple search task has been done to find the best combination of modalities for navigation and zooming. Most of the participants preferred to navigate by changing the device orientation and zoom by using the touchscreen. The same method also performed faster for the tasks. However at our final implementation we used both device orientation and touchscreen to navigate mouse and added a possibility to change these modalities.

We think that such a system is very useful in daily life. Lecturers can use Remotroid to avoid interacting with their computers. Remotroid can also be a replacement for a wireless-based projection like in DLNA compatible devices. With such systems users carry their computer and render some data in a bigger screen (mostly) while using Remotroid can remove the carrying part and handle the interactions.