Vim with Eye Tracker

Future User Interfaces 2017

Soumaya El Hariri Guillaume Pythoud Kevin Schibli

May 31, 2017

Contents

1 Introduction

The goal of the project in the *Future User Interfaces* course is to develop a multimodal interface. We decided to improve a text editor whose main modality is the keyboard by adding a foot pedal activated eye tracker as an additional modality. Then we compare that interface with the traditional keyboard and mouse combination.

2 Outline

2.1 Context

Our idea was to create a complementary and possibly faster way to use *Vim*[1], a command line text editor. The main philosophy of using *Vim* is to always keep your hands on the keyboard and in consequence be able to edit text faster. The cursor is moved with the keys `H J K L` which is great for small movements, however larger movements are more cumbersome, despite some more sophisticated keyboard shortcuts. This often leads to using the mouse for those and the hands are leaving the keyboard which causes a potential loss of efficiency. We wanted to avoid this loss by adding an eye tracker that moves the cursor *hands free*. With addition of a foot pedal to activate and deactivate the eye tracked cursor movement we find it a promising possibility of increasing the efficiency of using *Vim*.

2.2 CASE / CARE

The *CASE* model formalizes multimodal interfaces from the machine side. We classify our application as being alternate (See figure 1). We have one task, moving the cursor, that is executed using the pedal and the eye tracker in temporal alternation.

[1] `http://www.vim.org`

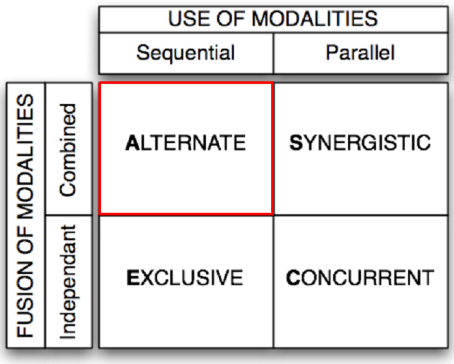

Figure 1: Classification according to the CASE model

The *CARE* model on the other hand assesses multimodal interaction from the human side. Our application uses *complementarity* for input fusion since both, the pedal and the eye tracker are to be used within a temporal window to move the cursor. The modalities are used sequentially.

2.3 Input Fusion

We use *data-level-fusion* between the pedal and the eye tracker. The features used to place the cursor in *Vim* e.g. the coordinates, come from the eye tracker. However this feature is only available as long as the pedal is pressed. The fusion of cursor and keyboard then takes place at the *decision-level*.

2.4 Hardware

Before settling for the hardware described below we evaluated several possibilities among others the *HTC Vive* VR headset from the departement.

2.4.1 Pedal

We built the pedal used in this project ourselves. The hardware consists of the following pieces:

- 3D printed frame
- Arcade pushbutton
- Arduino Uno[2]

[2] `https://www.arduino.cc/en/Main/ArduinoBoardUno`

2.4.2 Tobii Eye Tracker 4C

We use the *Tobii Eye Tracker 4C*[3]. It is a bar of ca. 30 cm that can be mounted below almost any monitor or laptop and provides eye tracking and head tracking using multiple cameras. An onboard chip that processes recorded data so that the CPU load on the host system should not be affected significantly.

3 Implementation

3.1 Pedal

We programmed the *Arduino* with a slightly modified version of the *State Change Detection* script from the documentation[4]. We changed it to send `On` or `Off` to the serial port once the pedal is pressed or released.

3.2 Eye Tracking

Tobii offers an SDK for their eye trackers which is available for C/C++, .NET, Unity and Unreal Engine. We used the .NET version because we are already familiar with C#. Since the *Eye Tracker 4C* is designed as an input method for video games, the SDK is accordingly feature rich. The only functionality we need for this project is the position of the eye gaze on the screen. This can be done with a few lines of code:

```
using (var eyeXHost = new EyeXHost())
{
  using (var stream = eyeXHost.CreateGazePointDataStream(
    GazePointDataMode.LightlyFiltered))
  {
    eyeXHost.Start();
    stream.Next += (o, args) => {
      Console.WriteLine("Gaze at " + args.X + "/" + args.Y);
    };
    // ...
  }
}
```

[3] `https://tobiigaming.com/eye-tracker-4c`
[4] `https://www.arduino.cc/en/Tutorial/StateChangeDetection`

The obtained gaze position now needs to move the cursor in *Vim*. To do so we use `USER32.dll`, a core Windows library. It allows us to emulate mouse clicks and choose the foreground window. We used the following functions:

```
IntPtr FindWindow(string lpClassName, string lpWindowName);
bool SetForegroundWindow(IntPtr hWnd);
bool SetCursorPos(int x, int y);
void mouse_event(int dwFlags, int dx, int dy, int cButtons, int dwExtraInfo);
```

Now everything is ready to connect all the parts. When the program starts, the window of *Vim* is brought to the foreground. As long as the pedal is kept pressed, the eye tracking is activated. For every gaze data received a mouse click is simulated at the gaze position.

3.3 Vim configuration

Below are our modifications to *Vim*'s settings followed by a brief description to ease the usage of the eye tracker.

```
set guifont=Monospace\ 36
```

Since the precision of our eye tracker is not good enough to use *Vim* with a normal font size equivalent to ca. 50 lines of text on the screen, we had to increase the font size to 36 resulting in 32 lines. The exact value depends on the monitor.

```
set virtualedit=all
```

Also with default settings the cursor cannot be moved to the middle of an empty line. This poses problems when the eye tracker tries to set the cursor just above the desired line on a line which happens to be empty. Consequently the cursor is set to the beginning of the empty line which is undesireable.

4 Evaluation

In our evaluation we wanted to verify if adding the modalities eye tracker and pedal to the base modality of *Vim*, the keyboard, improves the efficiency over adding the modality mouse.

4.1 Hypothesis

Our main hypothesis is:

> *H1:* It takes less time to edit source code in Vim with the eye tracker, pedal and keyboard than with mouse and keyboard.

Along with the null hypothesis:

> *H0:* It takes the same time to edit source code in Vim with the eye tracker, pedal and keyboard or with mouse and keyboard.

4.2 Experiment

This section describes the details of our experiment.

4.2.1 Subjects and Groups

All our subjects were required to have basic knowledge of *Vim* to minimize bias by the (quite steep) learning curve of *Vim*.

4.2.2 Setup

We prepared two Java files with 31 lines of code each along with a script of 6 changes to be made for each file. Below is the list of changes for one file. The number of lines between 2 changes within the files are kept rather large on purpose.

1. Change the author of the code with "Eddy Merckx"
2. Change the size of the new person to 175
3. Change toString() method to only return the firstName
4. Remove the commentary before the main()
5. Add a commentary before the main() that says "TODO: finish the project"
6. Add you name as a author

Every subject executes the two tasks on the respective Java file once with eye tracker, pedal and keyboard and once with mouse and keyboard. The order of the tasks and modalities were alternated for every other subject (within-group).

4.2.3 Variables

The dependent variable measured was the time to complete the task on the script.

The independent variables with their possible values are:

Eye tracking The modalities used. *Yes* - Eye tracker, pedal and keyboard or *no* - mouse and keyboard

Java file *1* or *2*, the chosen Java file to edit with a corresponding task list

Order *First* or *Last*, whether this measurement was taken from the first or the last task of a subject (see section 4.2.2)

4.3 Result

Figure 2 shows the measured time for the task with and without eye tracker. Along with a box plot visualizing quartiles and variability.

Time [s]	**Eye Tracking**
76	yes
49	no
76	no
97	yes
60	yes
53	no
50	no
71	yes
75	yes
58	no
40	no
45	yes

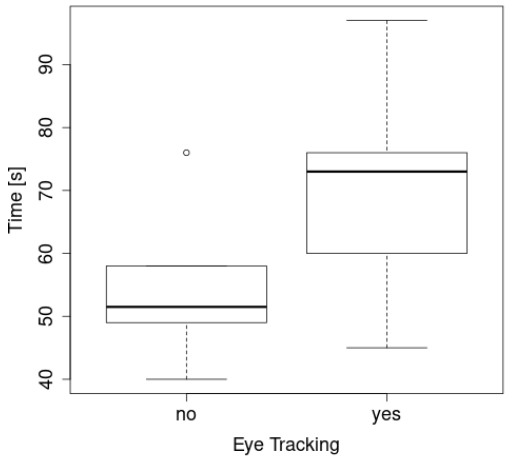

Figure 2: Measurements and box plot

We run the *Welch Two Sample t-test* on the measured time and used modalities. The mean time for the task with the eye tracker is 70.67s whereas the mean with the mouse is 54.33s. We could not reject the null hypothesis,

as the absolute value of $t = -1.8857$ is below two. The probability value $p = 0.09219$ is to high to have a 95 percent confidence interval.

5 Conclusion

The reason for not having significant results is probably due to the low number of only 12 measurements paired with different apprehension, typing speed and *Vim* skills. However we can still see a tendence towards the mouse being the faster input modality by comparing the means.
From the box plot we can see that the variance is bigger with eye tracking than without. This might result from the eye tracker being a new modality the subjects whereas using a mouse is already well trained.
We see the lack of precision of the eye tracker as the main handicap versus the mouse. However we felt that the general user friendliness of the eye tracker in combination with the pedal is quite good. Also it only takes little time to get used to the new modalities.