
An Effort at Simulating a Multi-touch Screen with Single-touch Hardware.

Girish Saini

College of Computing
Georgia Institute of Tech.
gsaini@cc.gatech.edu

Dhanik Shah

College of Computing
Georgia Institute of Tech.
dhanik@gmail.com

Onkar

College of Computing
Georgia Institute of Tech.
osnkars@gmail.com

Abstract

Multi-touch is a term that groups together a touch screen that recognizes multiple simultaneous touch points and software to interpret these simultaneous touches. Multi-touch values consist of the values of points that are touched and/or the relative pressure on these values. Several applications requiring multi-touch like two-finger zoom and rotation are more intuitive for the user. This project aims to determine if it is possible to simulate Multi-touch hardware with a single touch screen. Short-term cost benefits can be seen as the primary motivation for this work.

Keywords

Multi-touch, Single-touch, Hardware, Python, Jump Signature, Delta.

Introduction

A variety of rich gestures can be interpreted and used with multi-touch hardware, for instance two finger zooming and rotation is much more intuitive than its single stroke counterpart. Some large screen Multi-touch devices support more than one user on the same device at the same time, thus allowing even more novel ways of Interaction. A large portion of the project was devoted towards determining whether there can be

some software based distinction of single and multiple touches on a screen that reports just one value. We use the Motorola E-680i which features TFT single-touch screens with 65K colors on a 240 x 320 pixels with a screen size of 38mm x 50mm and the 02 XDA-II S Smartphone which features a 65K Colors, TOPOLY single-touch screen . The screen has a resolution of 240 x 320.

Part 1: The Motorola E-680i

Initially a demo python application was written to test the values obtained from the phones touch screen and it was found that if two points on the screen are pressed the phone reports a final point somewhere in between the two touches (close to the center). Also, according to initial observations "Jump Signatures" which we define as distance between consecutive points as reported by MOUSEMOTION events vary considerably for "normal" single strokes and dual touches. We provide graphs to illustrate a quantitative differentiation between these "Jump Signatures". From the graphs it will be evident that distance deltas are significantly greater for multi-touches than for single-touch strokes.

For a normal stroke geometric distance values in terms of pixel positions reported vary between 2 and 14. This is shown in Fig. 1. When compared with the "Jump Signature" of a multi-touch event we will see there exists a straightforward way of differentiation between these two events. Though the deltas between successive readings vary with speed of the stroke we use values that we consider would be "normal" in a usage scenario that consists or multi-touch zoom and rotation applications. Fig. 1 shows the deltas for 25

such points. All values reported were well below 14. Variation may be attributed to inconsistent drag speeds across the touch screen.

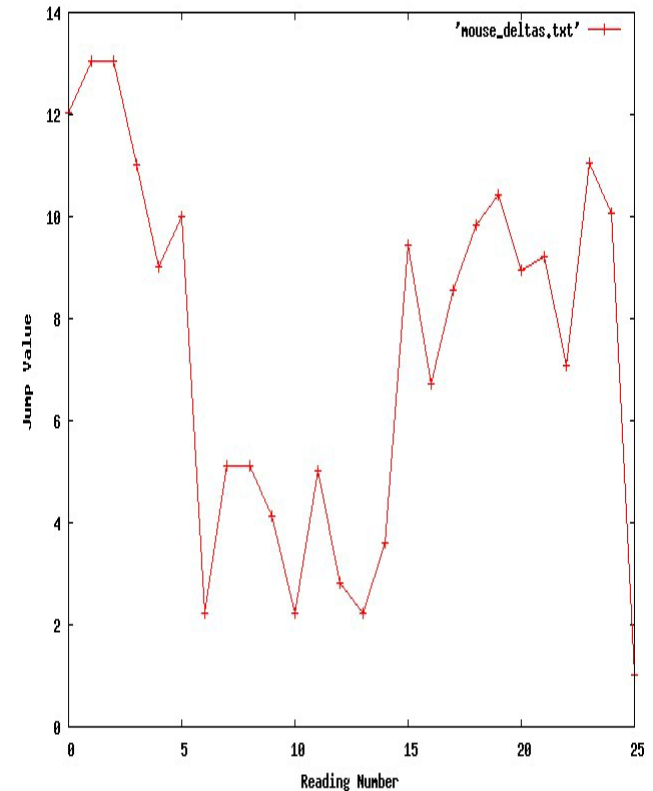


Figure 1 "Jump Signatures" for a single-touch stroke.

Fig. 2 shows the "Jump Signature" for a multi-touch event. The small values of around 3-4 show a slight movement of the stylus while placing it on the screen. Other values are consistently around the 25 mark. Note

distance is again measured as the geometric distance between mouse-position values.

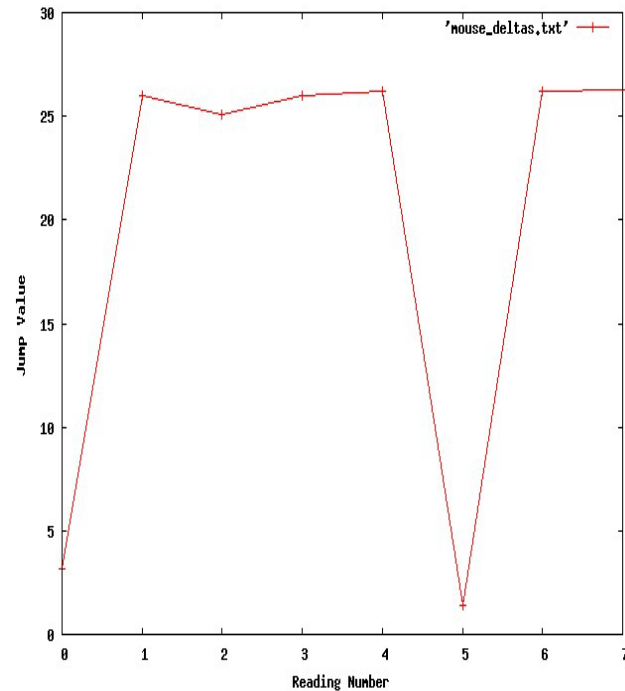


Figure 2 "Jump Signature of a multi-touch event.

Given a normal stroke (i.e. not too fast) and a multi-touch event this provides a straightforward way of distinction between them. Sample a few values and check if distance between successive MOUSEMOTION events is larger than a threshold.

Though the above procedure works well when styli are use when a person begins to use his thumbs or fingers then sampling the initial few values does not work as a strategy for differentiation between strokes and multi-

touch events. The touch screen reports a number of points around the thumb/finger press. Eliminating this 'noise' is essential to incorporate a degree of robustness into the application. Figure 3 shows the distance of points reported from the center of the thumb press.

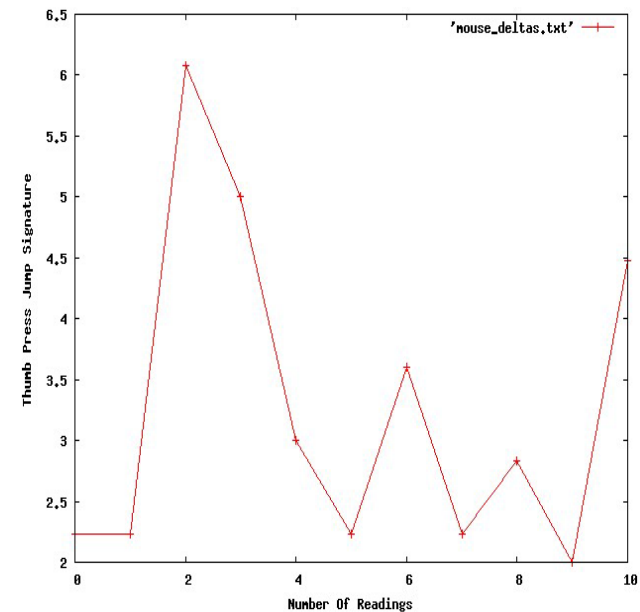


Figure 3 "Noise", Distance of points form center of thumb-press

Using the values noted in these steps it becomes straightforward to set up an application that identifies the event and performs actions based on the move that is detected. Our application is organized as a state machine and enters either the stroke or multi-touch modes based on the detected activity. Currently, we ignore all single touches but is trivial to incorporate

scrolling or some other activity that utilizes single-touches into the application.

Part 2: The O2 XDA2S

This was the second device we used for testing our algorithm to emulate the multi-touch effect.

The application was written in C# to test the values obtained from the XDA's touch screen and it was found that when two points are touched on the screen then the first point that is drawn on the screen lies between the two points touched on the screen.

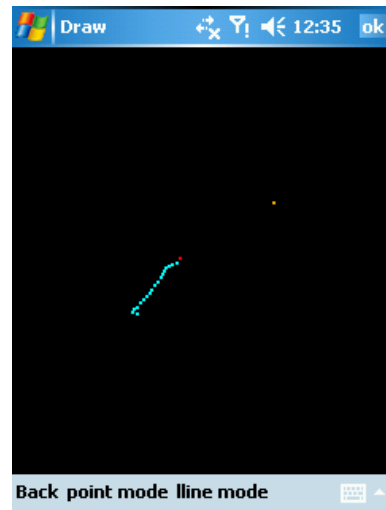


Figure 4 "Jump Signatures" for a two simultaneous points

It was found that if two points on the screen are pressed the phone reports a final point somewhere in

between the two touches (close to the center). Also, according to our observations "Jump Signatures" which we define as distance between consecutive points as reported by MOUSEMOVE events vary considerably for "normal" single strokes and dual touches. Also the points that are reported move slightly about the geometric center.

When two fingers are lifted from the screen the last point is reported as the finger which was lifted the fastest. Also in figure 4 the orange point is the first point where the finger was placed and the red point is the last reported point.

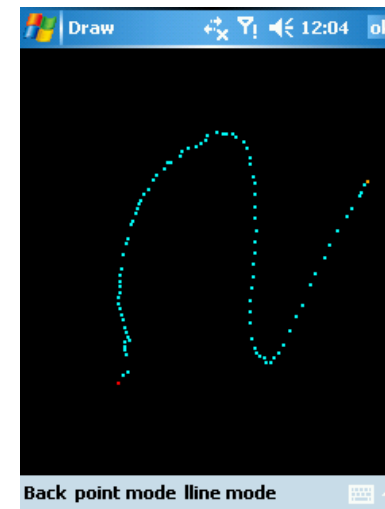


Figure 5 "Jump Signatures" for a single-touch stroke.

Fig. 5 shows the "Jump Signature" for a single-touch event. Each consecutive point in the single touch is at a distance less than 2px of each other. Also the distance between the points is evenly placed as is directly

proportional to movement of the finger. Fast movements were reported inaccurate and the distance between consecutive points was more than 10px in some experiments.

In Fig 6 we plot the co-ordinates reported during a multi touch motion. Here we place two fingers apart at a distance of 300px and then move them towards each other simultaneously. What we saw was that the points reported were in between the two starting points but what was unique was that the points were oscillating across the line joining the two points. The distance between each consecutive point was more than the single touch. This distance can be one deciding factor in differentiating the single touch from the multi-touch.

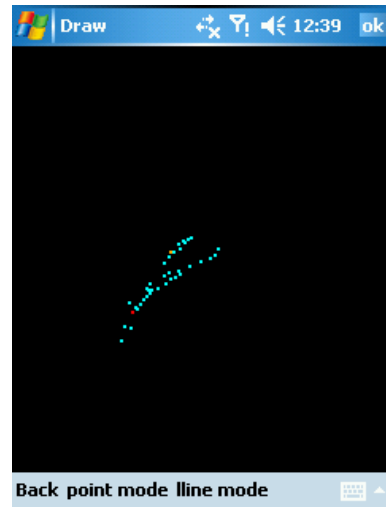


Figure 6 "Jump Signature of a multi-touch event.

Given a normal stroke (i.e. not too fast) and a multi-touch event this provides a straightforward way of

distinction between them. Sample a few values and check if distance between successive MOUSEMOTION events is larger than a threshold.

Though, as was the case for the E-680i the above procedure works well when styli are used, if thumbs or fingers are used sampling initial few touches does not work as a strategy for differentiation between strokes and multi-touch events. The touch screen reports a number of points around the thumb/finger press. Figure 7 shows the distance of points reported from the center of the thumb press on the XDA.

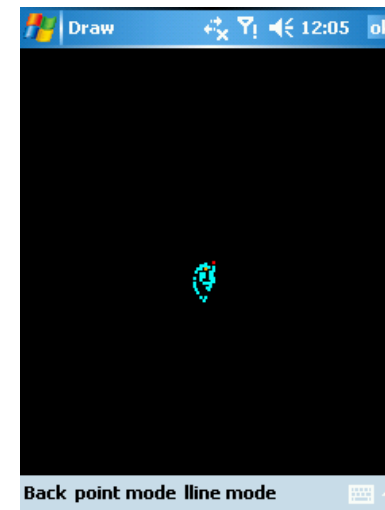


Figure 7 "Noise", Distance of points form center of thumb-press

Looking at the way in which the application reports the values of a single touch and the multi touch; we have been able to make our application only respond to multi

touch inputs. The single touch inputs are correctly identified and discarded.

Acknowledgements

We thank Thad Starner, Nirmal Patel and Chis Skeels for all their help and guidance along the way.

References

[1] Nobuyuki Matsushita, Yuji Ayatsuka, Jun Rekimoto. Dual Touch: A Two-Handed Interface for Pen-Based PDAs

[2] http://www.gsmarena.com/motorola_e680i-1136.php.

[3] http://en.wikipedia.org/wiki/Multi_touch_screen.

[4] <http://en.wikipedia.org/wiki/IPhone>.

[5] <http://www.billbuxton.com/multitouchOverview.html>.

[6] http://www.fingerworks.com/hand_gestures.html.