HIT3061 – Software Team Project - Semester 2, 2013

Leap Motion Development

Test Report

**Daniel Corsaletti**

SID: 6450458

E: 6450458@student.swin.edu.au

M: 0433 536 150

**Joshua Stopper**

SID: 5571391

E: 5571391@student.swin.edu.au

M: 0430 714 887

**Shengwei Li**

SID: 749999x

E: 749999x@student.swin.edu.au

M: 0420 478 750

**Minh Duc Nguyen**

SID: 171001x

E: 171001x@student.swin.edu.au

M: 0412 179 265

**Tran Xuong Tran**

SID: 6700691

E: 6700691@student.swin.edu.au

M: 0433 345 105

**Table 1. Document Change Control**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Changes |
| 1.1 | 28/10/2013 | Daniel Corsaletti | Create Document |

**Table 2. Document Sign Off**

|  |  |  |
| --- | --- | --- |
| Name | Signature | Date |
| Joshua Stopper |  |  |
| Minh Duc Nguyen |  |  |
| Tran Xuong Tran |  |  |
| Daniel Corsaletti |  |  |
| Shengwei Li |  |  |

Contents

[2 Introduction 4](#_Toc370813083)

[3 Test overview 4](#_Toc370813084)

[3.1 Testing phase 4](#_Toc370813085)

[3.2 Functional Testing 4](#_Toc370813086)

[3.3 Usability Testing 4](#_Toc370813087)

[3.4 Performance Testing 4](#_Toc370813088)

[4 Test Results 5](#_Toc370813089)

[4.1 Functional Testing 5](#_Toc370813090)

[4.2 Usability Testing 7](#_Toc370813091)

[4.3 Performance Testing 8](#_Toc370813092)

[5 Testing conclusions 9](#_Toc370813093)

[5.1 Defects 9](#_Toc370813094)

[6 Appendices 10](#_Toc370813095)

[6.1 Testing scripts 10](#_Toc370813096)

# Introduction

This document is used to display the results of testing the Tremor with Leap Motion application. It contains all of the results for tests executed during the testing phase of the project. Results for the functionality testing, the usability testing and the performance testing are all included, as well as scripts used to test the program.

# Test overview

## Testing phase

Throughout development of the project the application was tested based on performance issues and the application was refined to the final version that has been heavily tested in this document. The main test phase focussed mostly on functionality testing and performance testing as our usability testing was a very small and ongoing aspect. As stated in our test plan, we will not be testing the accuracy of the Leap Motion device, only our programs ability to use data captured by the device to generate results.

## Functional Testing

The functional testing was conducted by creating arrays that would be run through our application, checking whether our application was generating the same results that we intended it to. These arrays consisted of data that we believed would give us a good indication of how correct the results were, as well as some data that we intended to use to produce invalid results. The main functions that we tested were those that produced the values for the frequency, the amplitude, the velocity and the acceleration. We aimed to have functionality for noise cancellation developed in time to test with the final application; however we were not able to complete this functionality and thus it wasn’t tested for.

Another form of function testing that we completed was recording footage of each of our team members conducting a test and showing both the recording and the results to our client, in order to gauge whether the results being produced are somewhat similar to the results that he would expect from our hand movements.

## Usability Testing

To ensure our program was useable by our client, we mainly relied on feedback from him as to whether or not he was able to use the application we designed. Throughout development we were constantly checking how he wished the program would look so he was able to use it effectively.

## Performance Testing

The performance testing aimed to reproduce situations in which the application would and would not be able to be used effectively. We tested to try to identify any situation where the application could not be used due to the limitations of the Leap Motion device, or due to limitations of our application.

# Test Results

The tests conducted on the application are outlined below. Here we identify the specific criteria we tested for and the results that those tests produced.

## Functional Testing

To complete function testing it was necessary to create arrays of test data to run through the application. Each of these arrays was run through each of the main functions that we were testing. We identified what our expected results for each of these arrays would consist of then tested the program and listed the actual results that were produced. These arrays are:

|  |  |
| --- | --- |
| testArray1 | (1,2,3,4,5,6,7,8,9,10) |
| testArray2 | (-5,-4,-3,-2,-1,0,1,2,3,4,5) |
| testArray3 | (-110000,-105000,-1000,-100,-10,-1,0,1,10,100,1000,150000, 195000) |
| testArray4 | (-25,-15,-20,5,-5,20,0,1,5,10,9,10,9,5,-10,10) |
| testArray5 | (-5,5,-4,4,-3,3,-2,2,-1,1,0) |
| testArray6 | (5,10,4,6,8,1,0,4,9,5) |
| testArray7 | (-5,-10,-4,-6,-8,-1,0,-4,-9,-5) |
| testArray8 | (-10,10,-5,“A”,5,“B”,-3,3,-2,”C”) |

The first 7 arrays had data that we expected would work correctly when run through the functions, with testArray3 being one that we used to try identify if the program has any boundaries using large numbers. The last array was used to test if passing data other than the type expected caused any errors. We decided on running arrays of data like this through the functions because the application will extract numbers from the output of the Leap Motion device and send them through the functionality in arrays just like this.

The main functions being tested were the ones that produced the results for the velocity, frequency, acceleration and amplitude, however we also tested our application to make sure it was calculating the Euclidian distance of a user’s hand correctly. This functionality was to be used in the incomplete noise filter functionality. The results for these tests are:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Array** | **getVelocity**  **Expected** | **getVelocity**  **Actual** | **getFrequency**  **Expected** | **getFrequency**  **Actual** |
| testArray1 | 5.5 | 5.5 | 0 | 0 |
| testArray2 | 0 | 0 | 0 | 0 |
| testArray3 | 10000 | 10000 | 0 | 0 |
| testArray4 | .5625 | .5625 | 37.45 | 37.5 |
| testArray5 | 0 | 0 | 49.10 | 49.09 |
| testArray6 | 5.2 | 5.2 | 29.94 | 30 |
| testArray7 | -5.2 | -5.2 | 29.94 | 30 |
| testArray8 | Not valid | Not valid | Not valid | 18 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Array** | **getAcceleration Expected** | **getAcceleration Actual** | **getAmplitude Expected** | **getAmplitude Actual** |
| testArray1 | 1 | 1 | 0 | Not valid |
| testArray2 | 1 | 1 | 0 | Not valid |
| testArray3 | 25416.666 | 25416.666 | 0 | Not valid |
| testArray4 | 2.333 | 2.333 | 10.4 | 10.4 |
| testArray5 | .5 | .5 | 2.777 | 2.777 |
| testArray6 | 0 | 0 | 3.56 | 3.56 |
| testArray7 | 0 | 0 | 3.56 | 3.56 |
| testArray8 | Not valid | Not valid | Not valid | Not valid |

|  |  |  |
| --- | --- | --- |
| **Values**  **(x1,x2,y1,y2,z1,z2)** | **getEuclidian Expected** | **getEuclidian Actual** |
| 1,2,1,2,1,2 | 1.73 | 1.73205 |
| 2,5,3,9,5,1 | 7.81 | 7.81024 |
| -4,-2,-8,6,-10,2 | 18.55 | 18.54724 |

Note: getEuclidian function tested with arrays specified in the values column, they did not get tested with the above arrays.

These testing results show that our functions mostly work as intended; however there seems to be an issue running testArray8 through the getFrequency function. Our expected result was for the application to return an error message as we didn’t expect the function to be able to correctly calculate the average of an array with characters in it. Another strange discovery was the getAmplitude function returning not valid when there has been no directional changes in the values. These are both issues that we will address and rectify in the final build of the application. Other discrepancies occur from rounding of decimal places; however we are counting those values as correct.

These tests have helped us to confirm that our functions are working as intended. Another test we had completed to help identify if our application was working correctly was the video of recording of tests being completed which our client looked over. He was happy with these results but felt that a lot of the values seem very inconsistent. He noted that a user takes a test more than once and the values returned each time sometimes differ. As we feel we have testing our functions adequately and are happy with the results, we believe the variation in results is contributed to the Leap Motion device not always recording data correctly.

The belief that the data recorded from the Leap Motion device is not recorded consistently was backed up by some tests we conducted by running the application on 2 separate computers at once, with 2 Leap Motion devices analysing the data. Each test had the same hand movements recorded, however the results output by the applications were different. It seems that despite our best efforts to accurately detect a tremor the accuracy of the device can be somewhat questionable.

Testing was also completed on a model plastic hand in order to check the reliability of the application when it is testing a stationary object. These results however weren’t very conclusive as the Leap Motion device had trouble identifying the fingers and palm of the hand. This could be due to the fact that the hand was made of shiny black plastic and light may have been reflecting of the hand, causing issues for the Leap Motion device. A glove was placed on the hand; however issues still persisted with valid data capture so this testing was abandoned.

## Usability Testing

The usability testing for this application didn’t occur mainly during the testing phase for this project, it was an ongoing process during the development stage. As this application is being designed for our client to use and not as a final product we decided to design the application based on his feedback, so we were in constant contact with him and asked for what he would like the application to look like. Although it was designed mainly for the client to use, we also wanted the application to be easily useable by anybody so we aimed to design something that anybody could use whilst fulfilling our clients desires. Throughout development our client was always happy with the direction we had taken with the user interface design and happy with all the changes we suggested and implemented.

As we felt a proper test of the application in a usability lab was unnecessary we decided to instead focus on some situations that our client may be in that will affect how useable our application is. This seemed to be mainly because of the limitations of the Leap Motion device but we felt it was necessary to discover any issues that may persist in using the device and application in different environments.

|  |  |
| --- | --- |
| **Testing scenario** | **Result** |
| Application useable with either hand | Test completed with no issues. Either hand is able to be used when conducting a tremor test. The Leap Motion device has no issues detecting what hand is used, but the application still remains only useable with 1 hand |
| Application useable with gloves | Test completed with no issues. Both white and green latex gloves where used when testing and produced no issues |
| Application useable with jewellery | Test completed with some issues. Using the application while wearing regular sized rings, bracelets that do not dangle from the wrist and watches works fine. Using the application while wearing large rings and dangling bracelets causes problems for the Leap Motion sensor as it struggles to accurately identify all hands and fingers |
| Application useable under different lights | Test completed with almost no issues. The application was tested under fluorescent lighting varying between 20 – 50 watts and incandescent lighting varying between 75 – 100 watts. The application worked perfectly as indented in these situations; however it would rarely have issues working in an environment with low natural light or filtered light |
| Application useable with different clothing | Test completed with some issues. The application works correctly with any type of clothing unless the user has a baggy sleeve that dangles from or covers the wrist. This could include a jumper or even an unbuttoned shirt sleeve |
| Application useable with freshly cleaned hands | Test completed with no issues. As our client is a surgeon we believed it could be possible he will use the application just before surgery when he has freshly cleaned his hands. Application works correctly even when the user has wet, soapy, hot or cold hands |

The tests we conducted to check the usability of our application are all tests that we believed our user may find himself in. We believe that any of the issues that we discovered while testing usability are highly unlikely and are easily solvable, whether by simply turning on a light or pulling up a sleeve, however it is important we tested for these situations to uncover any issues that existed in our application.

## Performance Testing

When testing the application we aimed to discover any situations where it would not perform as well as it was intended to. Tests were conducted to identify what would make the application suffer and we tried to intentionally make the program stop working.

|  |  |
| --- | --- |
| **Testing scenario** | **Result** |
| Disconnect the computer from the internet mid test | Although this application runs in a web browser, an internet connection is not used. There was no issue |
| Disconnect the Leap Motion device from the computer mid test | The application freezes until the device is connected again. A better solution will be to display error message |
| Run the application on a computer that isn’t plugged into a power source | Application suffers from this. The application is not able to record as many frames per second of a user’s hand, so tests take longer and have different results |
| Run the application in different web browsers | The only browser that the application runs effectively in is Google Chrome. This was not the intention of the application, however the client uses Google Chrome so it is not a big issue |
| Find a position where the Leap Motion device is not able to accurately identify a user’s hand | Device is not able to recognize fingers as easily when the hand is held too far in front of the device. Detection range was fixed in the application |
| Move hands around quickly to check if hands still tracked | Some erratic movements can’t be tracked very easily by application, some movements still picked up in the detection range |
| Change the settings for how many hands and fingers will be tracked | On screen instructions will tell the user they have too many/ too few hands or fingers being tracked |

Some of the issues we discovered whilst doing performance testing are out of our control and can’t be rectified, however some of the issues that exist like only working on Google Chrome are not a big issue as we had designed it specifically to work in our user’s environment. The issue of the power needing to be plugged into a computer for the application to work correctly is one issue that we aren’t really able to remedy. Our application will be using the data gathered from the Leap Motion device and then performing calculations on it, if the Leap Motion device works better plugged into a computer it will be necessary to do that to use our application.

# Testing conclusions

After testing it is apparent that our application relies heavily on the Leap Motion device. Our testing has shown that our functionality works as intended, however results seem to vary when using the device suggesting that the device isn’t always able to accurately track a user’s hand. There are still issues that we can address to make sure our application works as it was intended to work. Issues to be resolved include the functionality accepting invalid arrays and the application freezing when disconnecting a device. These are simple fixes that will be implemented and handed over to the client with the final application. Although testing has been difficult for this application as we have been relying on another device being correct, it has helped us come to our final conclusions for this project and identify any issues that plagued the application along the way.

## Defects

This is a list of defects that were discovered throughout the testing stage of the project

|  |  |
| --- | --- |
| Application crashes when clicking the ‘X’ on the results page | Corrected |
| Grammatical error on the options selection for application | Corrected |
| Tremor test is running longer than the time that the user enters | Corrected |

# Appendices

## Testing scripts

These scripts were used to generate arrays and run them through our main functionality

var testArray1=new Array(1,2,3,4,5,6,7,8,9,10);  
var testArray2=new Array(-5,-4,-3,-2,-1,0,1,2,3,4,5);  
var testArray3=new Array(-110000,-105000,-1000,-100,-10,-1,0,1,10,100,1000,150000, 195000);  
var testArray4 = new Array(-25,-15,-20,5,-5,20,0,1,5,10,9,10,9,5,-10,10);  
var testArray5 = new Array(-5,5,-4,4,-3,3,-2,2,-1,1,0);  
var testArray6 = new Array(5,10,4,6,8,1,0,4,9,5);  
var testArray7 = new Array(-5,-10,-4,-6,-8,-1,0,-4,-9,-5);  
var testArray8 = new Array(-10,10,-5,'A',5,'B',-3,3,-2,'C');

console.log(getAmplitude(testArray1));

console.log(getAmplitude(testArray2));

console.log(getAmplitude(testArray3));

console.log(getAmplitude(testArray4));

console.log(getAmplitude(testArray5));

console.log(getAmplitude(testArray6));

console.log(getAmplitude(testArray7));

console.log(getAmplitude(testArray8));

console.log(getFrequency(testArray1));

console.log(getFrequency(testArray2));

console.log(getFrequency(testArray3));

console.log(getFrequency(testArray4));

console.log(getFrequency(testArray5));

console.log(getFrequency(testArray6));

console.log(getFrequency(testArray7));

console.log(getFrequency(testArray8));

var timeStamp1=new Array(1,2,3,4,5,6,7,8,9,10,11);

var timeStamp2=new Array(1,2,3,4,5,6,7,8,9,10,11,12,13);

var timeStamp3=new Array(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16);

console.log(getAccelerationAverage(testArray1,testArray1));

console.log(getAccelerationAverage(testArray2,testArray1));

console.log(getAccelerationAverage(testArray1,timeStamp2));

console.log(getAccelerationAverage(testArray1,timeStamp3));

console.log(getAccelerationAverage(testArray1,timeStamp1));

console.log(getAccelerationAverage(testArray6,testArray1));

console.log(getAccelerationAverage(testArray7,testArray1));

console.log(getAccelerationAverage(testArray8,testArray1));

console.log(getVelocityAverage(testArray1,testArray1));

console.log(getVelocityAverage(testArray1,testArray1));

console.log(getVelocityAverage(testArray1,timeStamp2)); console.log(getVelocityAverage(testArray1,timeStamp3));

console.log(getVelocityAverage(testArray1,timeStamp1));

console.log(getVelocityAverage(testArray1,testArray1));

console.log(getVelocityAverage(testArray1,testArray1));

console.log(getVelocityAverage(testArray1,testArray1));