**Measuring Stress Using Computer Mouse Movements**

**Dylan Drein**

**13344741**

Final Year Project – **2017**

B.Sc. Single Honours in

Computer Science

****

Department of Computer Science

Maynooth University

Maynooth, Co. Kildare

Ireland

A thesis submitted in partial fulfilment of the requirements for the B.Sc.

Single Honours in Computer Science

Supervisor: **Kevin Casey**

Contents

[Declaration i](#_Toc476594711)

[Acknowledgements ii](#_Toc476594712)

[Abstract iii](#_Toc476594713)

[List of Figures iv](#_Toc476594714)

[List of Tables iv](#_Toc476594715)

[Chapter one: Introduction 1](#_Toc476594716)

[Summary 1](#_Toc476594717)

[1.1 Topic addressed in this project 1](#_Toc476594718)

[1.2 Motivation 1](#_Toc476594719)

[1.3 Problem statement 1](#_Toc476594720)

[1.4 Approach 2](#_Toc476594721)

[1.5 Metrics 2](#_Toc476594722)

[1.6 Project 2](#_Toc476594723)

[Summary 2](#_Toc476594724)

[2.1 Topic material 2](#_Toc476594725)

[2.2 Technical material 2](#_Toc476594726)

[Chapter three: The Problem 3](#_Toc476594727)

[Summary 3](#_Toc476594728)

[3.1 Project UML documentation 3](#_Toc476594729)

[3.2 Problem analysis 3](#_Toc476594730)

[Chapter four: The Solution 4](#_Toc476594731)

[Summary 4](#_Toc476594732)

[4.1 Analytical Work 4](#_Toc476594733)

[4.2 Architectural Level 4](#_Toc476594734)

[4.2 High Level 4](#_Toc476594735)

[4.2 Low Level 4](#_Toc476594736)

[4.2 Implementation 4](#_Toc476594737)

[Chapter five: Evaluation 5](#_Toc476594738)

[Summary 5](#_Toc476594739)

[5.1 Solution Verification 5](#_Toc476594740)

[5.2 Software Design Verification 5](#_Toc476594741)

[5.3 Software Verification 5](#_Toc476594742)

[5.3.1 Your test approach (i.e. unit testing, sub-system testing, system testing) 5](#_Toc476594743)

[5.3.2 Your tests (e.g. scenarios, test cases, test data, etc.) 5](#_Toc476594744)

[5.3.3 Your test results 5](#_Toc476594745)

[5.3.4 An interpretation of the results 5](#_Toc476594746)

[5.4 Validation/Measurements 5](#_Toc476594747)

[5.4.1 Results 5](#_Toc476594748)

[5.4.2 Explanation of Results 5](#_Toc476594749)

[5.4.3 Analysis of Results 5](#_Toc476594750)

[5.4.4 Comparison with previous solutions (if relevant) 5](#_Toc476594751)

[Chapter six: Conclusion 6](#_Toc476594752)

[**Summary** 6](#_Toc476594753)

[**5.1** **Contribution to the state-of-the-art** 6](#_Toc476594754)

[**5.2** **Results discussion** 6](#_Toc476594755)

[**5.3** **Project Approach** 6](#_Toc476594756)

[**5.3** **Future Work** 6](#_Toc476594757)

[References 7](#_Toc476594758)

[Appendices 8](#_Toc476594759)

[Appendix 1 Schematic of the hardware associated with this project. 9](#_Toc476594760)

[Appendix 2 Code developed for this project. 10](#_Toc476594761)

[Appendix 3 UML Class, Use Case and sequence diagrams for this project. 11](#_Toc476594762)

[Appendix 4 Screen shots of the project implementation 12](#_Toc476594763)

## Declaration

I hereby certify that this material, which I now submit for assessment on the program of study as part of **(add your degree here)** qualification, is *entirely* my own work and has not been taken from the work of others - save and to the extent that such work has been cited and acknowledged within the text of my work.

I hereby acknowledge and accept that this thesis may be distributed to future final year students, as an example of the standard expected of final year projects.

Signed: Date:

## Acknowledgements

## Abstract

Style

The abstract should be a microcosm of the full report.

The abstract must be self-contained, **without** abbreviations, footnotes, or references.

The abstract must be between 150-250 words.

The abstract must be written as one paragraph, and **should not** contain displayed mathematical equations or tabular material.

The abstract should include three or four different keywords or phrases, as this will help readers to find it.

Ensure that your abstract reads well and is grammatically correct.

## List of Figures

## List of Tables

[Table 2‑1 Table of interest: Aspect of your implementation 2](#_Toc476595277)

[Table 2‑2 Data sources used in your implementation 2](#_Toc476595278)

# **Chapter one: Introduction**

## Summary

Chapter 1 describes an overview of the existing work in this field and the motivation for carrying out this project as well as an outline of the approach and methodology used, from planning, through the implementation to the evaluation of the results.

## 1.1 Topic addressed in this project

This project uses computer mouse movement data collected over the 2014/2015 college year in and out of a college lab exam environment from 128.

The aim of the project was to use statistical analysis to determine whether or not information obtained from the raw data could be used to infer a relationship between user behaviour and stress levels and the environment in which the user used the computer.

Just over 19.6GB of raw mouse data was collected using a JavaScript based mouse event handler which stored the data in JSON format. The relevant data includes a UNIX timestamp for every mouse event, the nature of the event (mouseUp, mouseDown, mouseMove etc.), the X and Y coordinates of the mouse cursor on the screen (relative and absolute) as well as pseudonym identifiers for each user.

## 1.2 Motivation

The motivation for this project was to provide a passive, non-invasive and inexpensive model of detecting stress in users by monitoring their behaviour through computer mouse movements.

Previous research has strongly indicated that stress in the workplace has a negative effect on employee efficiency and employee performance, affecting work ethic and contributing to employee ‘burnout’. [1] Furthermore, in college students stress has been shown to negatively affect physical and emotional health [2] and strongly correlates with high levels of depression and anxiety. [3]

Although stress indicators are well studied and understood [4], measuring and collecting data in an unobtrusive and effective way is not always straightforward. [5] Electroencephalography (EEG) and electrocardiography (ECG) are proven methods in indicating stress in subjects [6], though they require specialised and expensive equipment to be in place before, during and after the stressful situation. Custom sensing hardware encounters similar issues, while self-report tools can provide biased and unreliable feedback based on the environment in which testing takes place. [5]

## 1.3 Problem statement

The project involved the parsing, categorising and analysis of vast amounts of data collected from 129 students over a college year. The raw data required a lot of cleaning before it could be effectively analysed in any meaningful way. The data also had to be parsed into 129 separate user files so that individual analysis could be carried out across the controlled situations.

It was necessary to determine what information would be most useful to obtain from this data to achieve the aims of the project before it was cleaned, in order to know what fields to keep and what to remove. This involved identifying the relevant analytical methods that would be required to carry out the appropriate analyses and how best to apply these analyses to the data set.

## 

## 1.4 Approach

### 1.4.1 Initial Decisions

Python was the programming language used to carry out this project, chosen for its widely supported and well documented statistical analysis and numerical packages, namely NumPy, SciPy, Pandas, scikit-learn for machine learning and matplotlib for graph plotting. Initially Project Jupyter’s IPython[7] command shell was used which offers an interactive browser-based development notebook, though as time went on the Windows command prompt was used as the predominant Python interpreter.

### 1.4.2 Designed Solution

Each of the metrics in section 1.5

### 1.4.3 Evaluation

Data was initially analysed by graphing its spread with histograms, scatter plots and box plots using matplotlib plotting software. Doing this allowed for further relevant analysis to be identified based on how the data looked visually and also allowed for tweaks and changes in the previous code to be made to better suit the data once it had been visualised.

In addition, descriptive statistics such as the mean, median, standard and variation were used alongside the visual analysis. This information was used to compare data between samples, allowing us to identify and test for correlation between in lab and out of lab conditions for each of the metrics highlighted in section 1.5.

Hypothesis testing was used to determine if corresponding metrics from in lab and out of lab environments were statistically different from one another. A student’s t-test was used to carry this out, using log transformations on data sets which were not normally distributed to normalise them. Where statistical significance was found, results from previous research was used as a basis for inferring stress based on the observed behaviour.

## 1.5 Metrics

Based on the spread of the data [**LINK TO WHERE IN DOC THIS IS DISCUSSED]**, relevant mouse movement sequences (hereafter referred to as *Click Sequences)* are limited to those which last for 1450ms and end with a *mouseDown* event or where another Click Sequence begins.

The relevant metrics that were obtained from the data and used for analysis were:

1. **Distance:** Both actual distance and optimal distances were calculated for each Click Sequence. Actual distance measured the real mouse path distance of each Click Sequence. Actual paths were made up of a collection of mouse event *‘mouseMove’* points and so the distances between these points were summed to approximate the distance travelled. Optimal distance was calculated as the Euclidean distance between the start and end point of the Click Sequence. [**CHECK WHAT FORMULA NUMPY NORM USES]**
2. **Time:** The duration of each Click Sequence was calculated in milliseconds.
3. **Speed:** The speed of each Click Sequence was calculated in pixels per millisecond (px/ms) and pixels per second (px/s).
4. **Click hover time:** The duration for which each user ‘hovered’ the cursor over the click point before clicking the mouse was calculated for each Click Sequence.
5. **Efficiency:** Efficiency of each Click Sequence was calculated by calculating:
6. **Overshoot:** The distance for which the user ‘overshot’ the target with the mouse cursor was calculated for each Click Sequence.
7. **X/Y-axis error:** The error was calculated for optimal and actual distance for the individual *x-y* components of each Click Sequence. This used the same method as (**1.**) though it measured the change in distance along each axis separately for each Click Sequence.

## 1.6 Project

# **Chapter two: Technical Background**

## Summary

## 2.1 Topic material

## 2.2 Technical material

Table 2‑1 Table of interest: Aspect of your implementation

|  |  |
| --- | --- |
| **Column description 1** | **Column description 2** |
| A | Text 1 |
| B | Text 2 |
| C | Text 3 |

Table 2‑2 Data sources used in your implementation

|  |  |  |
| --- | --- | --- |
| **Column description 1** | **Column description 2** | **Column description 3** |
| X | 22 | 33 |
| Y | 33 | 456 |
| Z | 17 | 22 |

# 

# **Chapter three: The Problem**

## Summary

The aim of the project is to create a system which analyses collected mouse movement data and identifies correlations between the specific metrics outlined section 1.5 for lab and non-lab environments. The raw data needs to be cleaned and prepared appropriately so that this can be carried out and the results need to be obtained in such a way that they can be statistically analysed to produce useful information.

## 3.1 Technology to use

Python will be used to analyse the data with its various statistical and data analysis packages and modules. Python has modules for reading and writing both JSON and CSV formatted data, which will prove useful as discussed below.

As the data is read in using Python’s .load() and .reader() modules, NumPy and SciPy support data storage in NumPy arrays which allow for fast and efficient computation to be performed on each element stored in the array. NumPy supports 64-bit floating point number precision which is used for all values throughout this project.

## 3.2 Preparing the Data

The JavaScript mouse tracking software compiles all of the mouse data for each user into a single log file. There are over 88 million mouse events in this log file with each event entry holding 11 fields. It will be necessary to remove non-relevant fields and delete duplicate event entries to make the analysis as efficient as possible. **[initialcleaning.py]**

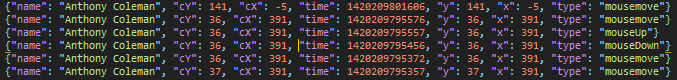
****

Figure 3-1: Cleaned, reversed JSON data showing a mouse click event.

One of the 11 fields recorded in each mouse event is the UNIX timestamp of when that event took place. In the log file, entries were logged from oldest to most recent, with the mouseDown event being the final event in a Click Sequence. During analysis the data will be read in line by line, starting with the mouseDown event and working backwards to the end of the Click Sequence or until another Click Sequence is encountered. Thus, the data needs to be loaded in the reverse order of which it is saved. The serialised nature of the data based on the timestamp means it is straightforward to reverse the log file ordering **[FileReverse.py]** and save the data from newest to oldest instead to make the backwards traversal of the Click Sequences possible.

After the data has been cleaned and reversed it will be necessary to create separate files for each user’s mouse movement data. **[InitialCleaning.py]** This will involve parsing the data based on the name field saved in each mouse event.

Due to the tabular nature of the data, with each column holding the same fields in each row entry, it may be appropriate and indeed more efficient to convert the JSON files to comma separated value files using the built-in reader and writer objects of Python’s CSV module.

**[INCLUDE SOMETHING ABOUT X VS CX AND Y VS XY COORD TYPES]**

## 3.3 Isolating Relevant Data

### 3.3.1 Isolate Click Sequences

As the project is concerned primarily with the analysis of user Click Sequences, it will need to be determined what does and what does not constitute a valid mouse Click Sequence. A generic definition of what we expect a valid Click Sequence to look like will need to be used as a general rule at first, with refinements and improvements made *a posteriori* based on the spread of the data and also based on previous research approaches that can be applied to this data.

**3.3.2 User Environment**

User mouse activity will need to be classified into two categories: Click Sequences made in a lab environment and those made outside of a lab environment.

The dates and times of the two labs that this project will focus on are known. The timestamps of both of these labs will need to be converted to UNIX so that, as each mouse event is loaded into the CSV module row by row, they can be compared to the timestamps of mouse events for each user and the mouse events can be grouped accordingly. Classifying the data in this way is what will allow us to compare metrics in stressful and non-stressful environments.

**Users**

In Lab Environment

Out of Lab Environment

Speed

Distance

Time

…

..

.

Speed

Distance

Time

…

..

.

Compare

Figure 3-2: Classification of User Click Sequences Between Environments [??????]

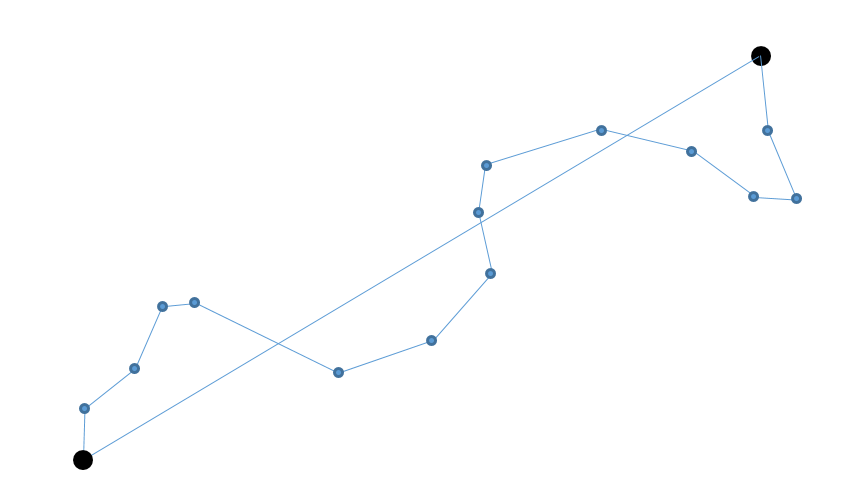
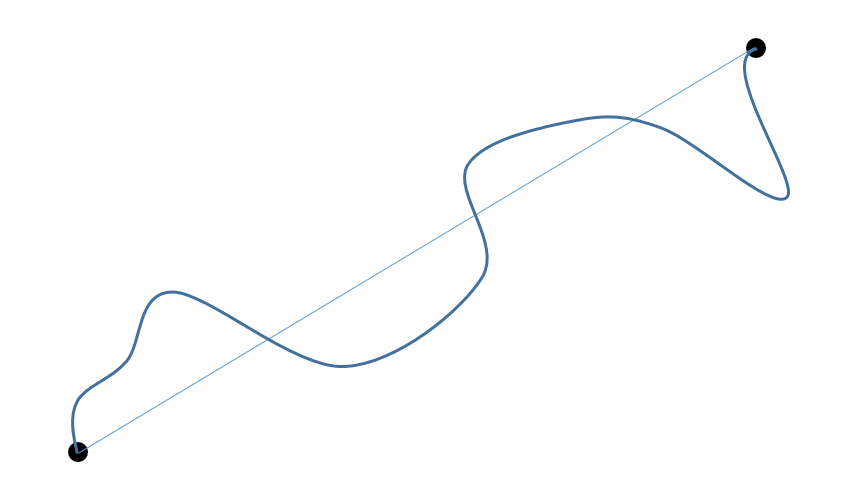
## 3.4 Calculating Metrics

The metrics that will be measured as part of this project will be calculated based on previous research in this field and what has been successful[8], as well as being based on the data itself and what can be computed from it.

**Distance**, both observed and optimal, will need to be calculated for each Click Sequence. Optimal distance is viewed as the Euclidean distance between the initial point of the Click Sequence to the final point, where a mouseDown event is observed (though, as stated, it is calculated in reverse). The actual distance will be approximated by calculating the Euclidean distance between each mouse move event along the observed mouse path from initial point to the mouseDown event and calculating the sum of these interval distances.

MouseDown Event

MouseDown Event



X

Y

(xn, yn)

(xn+1, yn+1)

Initial Point

Initial Point

Figure 3-3: Actual mouse path (left), Approximation of actual mouse path using intermediate Euclidean distances between points (right)

**Time** duration of each Click Sequence will be calculated using the UNIX timestamps associated with each mouse event. As the data will be analysed in reverse order, the initial mouse event the CSV module will read in will be the most recent, the mouseDown event. The parser will then work backwards to the end of the Click Sequence, saving the timestamp of the first and last point in the sequence and calculating the duration of the Click Sequence as follows:

**Efficiency** will be used as a measure of how close the actual path length of the Click Sequence was to the optimal path length. Using:

The efficiency of each Click Sequence can be parameterised as a numerical value such that with indicating greatest possible efficiency. Due to the fact that efficiency of a Click Sequence is calculated using distances, it may be required to apply some sort of weighting such as time or speed to the efficiency values when plotting their spread so that shorter Click Sequences do not result in a disproportionate level of skew towards high efficiency.

**Speed** of mouse Click Sequences will be measured in the traditional way as the rate of change in Actual distance with respect to time. The average speed of the entire Click Sequence will be measured, though it may prove useful to calculate differences in speed across specific intervals of the Click Sequence such as comparing the initial speed of the cursor movement to the speed of the cursor just before the mouseDown event.

**Overshoot** will measure how far the user overshoots the target click point before bringing the mouse back to that point. It should provide a useful measure of how accurate each Click Sequence is in getting from the initial point to the click point. It is expected that Click Sequences with large overshoot distances would have a corresponding drop in the measure of efficiency. Based on previous research [**CITE]** this measure can also be compared to metrics such as speed and time taken to get a better indication of the users stress level during that click sequence.

**[INSERT OVERSHOOT DIAGRAM HERE]**

**Click Hover Time** will be measured to indicate how long the user hovers the mouse cursor over the mouseDown event point before physically clicking the mouse button. This metric can be compared with speed, among other metrics, to investigate the relationship between hesitation and efficiency.

**Axis error** will be investigated to determine if there is any significant difference in the efficiency or the level of error between Actual distance and Optimal distance when the x and y axis movements are considered in isolation rather than together. This may prove a stronger indication of efficiency as it can better account for the variation in movement of the human arm vertically and horizontally when using a computer mouse.

**USE THIS TO MEASURE EFFICIENCY TOMORROW. OPTIMALX/ACTUALX ETC. GRAPH AS BEFORE!**

## 3.5 Problem Analysis

# **Chapter four: The Solution**

## Summary

## 4.1 Analytical Work

## 4.2 Architectural Level

## 4.2 High Level

## 4.2 Low Level

## 4.2 Implementation

# **Chapter five: Evaluation**

## Summary

## 5.1 Solution Verification

## 5.2 Software Design Verification

## 5.3 Software Verification

### 5.3.1 Your test approach (i.e. unit testing, sub-system testing, system testing)

### 5.3.2 Your tests (e.g. scenarios, test cases, test data, etc.)

### 5.3.3 Your test results

### 5.3.4 An interpretation of the results

## 5.4 Validation/Measurements

### 5.4.1 Results

### 5.4.2 Explanation of Results

### 5.4.3 Analysis of Results

### 5.4.4 Comparison with previous solutions (if relevant)

**Chapter six: Conclusion**

**Summary**

**5.1 Contribution to the state-of-the-art**

**5.2 Results discussion**

**5.3 Project Approach**

**5.3 Future Work**

# **References**

[1] W. U. Ali, A. R. Raheem, A. Nawaz, and K. Imamuddin, “Impact of Stress on Job Performance: An Empirical study of the Employees of Private Sector Universities of Karachi, Pakistan,” *Res. J. Manag. Sci. Res. J. Manag. Sci*, vol. 3, no. 7, pp. 2319–1171, 2014.

[2] T. Baghurst and B. C. Kelley, “An Examination of Stress in College Students Over the Course of a Semester,” *Health Promot. Pract.*, vol. 15, no. 3, pp. 438–447, May 2014.

[3] R. Beiter *et al.*, “The prevalence and correlates of depression, anxiety, and stress in a sample of college students,” *J. Affect. Disord.*, vol. 173, pp. 90–96, 2014.

[4] N. Mucci *et al.*, “Work-related stress assessment in a population of Italian workers. The Stress Questionnaire,” *Sci. Total Environ.*, vol. 502, pp. 673–679, Jan. 2015.

[5] D. Sun, J. Canny, and P. Paredes, “MouStress: Detecting Stress from Mouse Motion.”

[6] J. F. Alonso, S. Romero, M. R. Ballester, R. M. Antonijoan, and M. A. Mañanas, “Stress assessment based on EEG univariate features and functional connectivity measures.”

[7] “IPython: A System for Interactive Scientific Computing Python: An Open and General- Purpose Environment,” 2007.

[8] T. Yamauchi, “Mouse trajectories and state anxiety: Feature selection with random forest,” in *Proceedings - 2013 Humaine Association Conference on Affective Computing and Intelligent Interaction, ACII 2013*, 2013, pp. 399–404.

**Appendices**

## Appendix 1 Schematic of the hardware associated with this project.

## Appendix 2 Code developed for this project.

## Appendix 3 UML Class, Use Case and sequence diagrams for this project.

|  |
| --- |
|  |
| Appendix 4 Screen shots of the project implementation |
|  |