Login session using mouse biometrics

Christopher Johnsrud Fullu

Masteroppgave
Master i Teknologi - Medieteknikk
30 ECTS
Avdeling for informatikk og medieteknikk
Høgskolen i Gjøvik, 2008

Avdeling for
informatikk og medieteknikk
Høgskolen i Gjøvik
Postboks 191
2802 Gjøvik

Department of Computer Science
and Media Technology
Gjøvik University College
Box 191
N-2802 Gjøvik
Norway

Abstract

Biometric authentication is a field of strong research at the moment, some of the technologies are however more robust or well explored than others. Fairly robust technologies include retina scan and fingerprint authentication. The drawback with many of the more proven biometric authentication technologies is that they require some equipment not available to all persons. Equipment might be too large or not available for movement, or just to expensive. Fingerprint scanners are an exception that now is becoming more and more available to all persons, i.e. by integration on a laptop. A biometric that has not been a subject for that much research is the field of mouse authentication. The research that has been done in this has mainly concentrated on continuous authentication, that is detecting if a user changed. There has hardly been any research into the field of using the mouse as a login type way of authenticating people, that is available at least. The goal of this thesis is to design tasks for a user and collect data from this. What features can be extracted and how to compare them will be the main focus. Unlike some other biometric authentication methods like retina scan, it also does not seem threatening to people.

The experiment participants will be supplied with a program. This program collects positioning data and time. Two tasks are presented to the users; a stay-within-the-lines and a click-the-dots challenge. Users will perform these challenges over a few weeks.

The main topic of this thesis also makes it natural to touch on some other fields, such as human interaction. For instance if a user is told to use some equipment other than the one he or she is used to, is it plausible that this affects the way that this person is using the mouse. Thus submitting incorrect information on how he or she normally is using the mouse. Therefore the experiment participants will not be using any other hardware then their own. This results in that a study on how a different hardware and software preference affects the results can be conducted. Another topic is social exclusion, this applies to people that cannot use the mouse for some reason, either as the result of a handicap or some other reason.

Sammendrag

Biometrisk autentisering er et felt hvor det skjer mye forskning for tiden, noen av de foreslåtte teknologiene er derimot mer robust eller bedre forklart en andre. Eksempel på robuste teknologier kan være retina skanning eller fingeravtrykk autentisering. Bakdelen med mange av de mer etablerte biometri autentiseringsmetodene er at de krever forskjellig utstyr som ikke alltid er tilgjengelig for den vanlige person. Utstyret kan ta mye plass, kan ikke flyttes eller være veldig dyrt. Fingeravtrykklesere er ett unntak siden det nå har blitt mer og mer vanlig ved og for eksempel å være integrert på en bærbar pc. En felt innen biometri autentisering som er ganske nytt og ikke har sett så mye forskning er bruk av en datamus til autentisering. Forskningen som har blitt gjort har hovedsakelig konsentrert seg omkring kontinuering autentisering. Kontinuerlig autentisering betyr at man hele tiden følger for å se om en bruker har forandret seg. Det har nesten ikke vært noe forskning i det hele tatt som konsentrerer seg omkring bruken av mus som en login autentisering, eller statisk autentiseringsmetode. I alle fall ikke som er tilgjengelig. Målet for denne master oppgaven er å designe oppgaver for en bruker og samle data fra disse oppgavene. Hovedfokuset vil være å finne hvilke unike detaljer for brukerne vi kan dra ut fra disse dataene. Til forskjell fra noen andre biometriske autentiseringsmetoder som retina skanning, vil ikke bruk av en datamus virke truende på brukerne.

Eksperiment deltagerne vil bli gitt ett program. Dette programmet samler posisjons- og tids-data. To forskjellige oppgaver vil bli presentert for brukerne: en labyrint og en klikk-på-prikkene oppgave. Deltakerne vil gjennomføre disse testene flere ganger over noen få uker.

Hovedfokuset i denne oppgaven gjør det også naturlig å diskutere områder som menneskelig maskin interaksjon. For eksempel kan en bruker kanskje endre måten hun eller han bruker musen på hvis man må bruke annet utstyr enn det man er vandt til? Dette kan resultere i data som ikke direkte representerer denne brukeren. Derfor vil deltakerne av eksperimentet ikke bruke annet utstyr enn det de er vandt til, dvs. deres egne datamaskiner. Dermed kan man også gjøre en studie på hvordan ulikt hardware og software kan påvirke brukerne. Et annet felt som også kan forskes på er om datamus autentisering kan resultere i sosial utestenging. Noe som betyr at noen brukere ikke kan bruke autentiseringsmetoden og dermed ikke får brukt tjenesten som er assosiert med denne autentiseringsmetoden.

Acknowledgments

First and foremost I would like to direct a special thanks to my supervisor Patrick Bours of Gjøvik University Collage (GUC). His continued support, interest and inspiration is what made this master thesis possible. I would also like to thank him for introducing me to the interesting field that is biometrics. I would also like to thank the IMT department at GUC for allowing me to perform my master thesis in the field of authentication.

A thanks also goes out to fellow students Ketil Holien (for his implementation of DTW which is what I base my implementation of Levenshtein distance on), Kennet Flady (for his DET curve and EER calculation algorithm), as well as support and suggestions of all other students that have worked on their own master thesis. Much inspiration have come from the help and suggestions by fellow students.

I would also like to thank Dr. Ing. Rune Hjelsvold and Dr. Tech. Faouzi Alaya Cheikh of GUC for suggestions toward the end of the thesis period.

Last but not least I would like to thank the participants. Without their help this thesis would not have been possible.

Christopher Johnsrud Fullu, 30th June 2008

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

This chapter introduces the topics covered by this thesis, problem description, the justification, motivation and benefits and the research questions for the master thesis.

1.1 Topics covered by this thesis

Biometric authentication is a way of authenticating people by looking at their behavioral and physiological features. These features are being subjected to more and more research and are more widely taken into use. Using the mouse as an authentication tool has not been widely researched yet. The research that have been done on the subject focuses mainly on continuous authentication. Continuous authentication means the detection of user change over time.

This master thesis will focus on looking at the mouse as a static authentication tool. Research will consist of developing appropriate distances measurements to distinguish users and appropriate preprocessing to help this process.

1.2 Keywords

Biometrics, mouse dynamics, social exclusion, edit distance (Levenshtein distance)

1.3 Problem description

The focus of this master thesis is to look at the computer mouse as a authentication tool. The main idea is to look at the possibility to do statical authentication. What type of data and how to collect this data will have to be determined. This involves designing tasks for the user to complete in order to collect the data. Different tasks will produce data that will have to be analyzed differently. According to the type of task, different features can be extracted from the data. This means different preprocessing and different distance metrics will have to be researched to fit the task. This is needed to try to achieve the best possible performance of the authentication system. According to the results the best possible usages for the system or a specific task could be suggested. Whether it is in combination with an other system or a possible alternative to an existing system.

Social exclusion is a potential problem with biometric authentication systems. For a mouse authentication system this could also exist. This could be determined by looking if the data is stable. Unexperienced, old or disabled people might have different difficulties with using a mouse. These people might still use a computer for authentication in different ways.

1.4 Justification, motivation and benefits

The field of biometric authentication has become widely accepted as a trusted way of authentication. This field encompasses many authentication methods such as iris scan, fingerprint, gait and so on. These methods, apart from perhaps fingerprint, do require special equipment that is not necessarily available everywhere. There is a demand for safe and widely available biometric authentication methods. Biometrics are a description of authentication methods that is based on checking individual characteristics of a person. It is much harder to mimic something that a person is than something that a person knows. Lately pointing devices have gotten the attention for such a purpose. Some positive sides of using a mouse as a authentication tool:

- It is cheap
- Most people know how to use it
- Not a threatening device
- An alternative to passwords, which could be hard to remember

The research done in this field has mainly concentrated on continuous authentication. This thesis will focus on the mouse as static authentication tool. The amount of research in the field of mouse authentication is generally scarce.

1.5 Research questions

The research questions are divided into two groups, the first one regarding data collection and the second one regarding analysis. This is discussed further in Chapter 4.

First, can we use mouse authentication. To determine this data will have to be collected. What kind of data should be collected and how should it be done. How should the collected data be analyzed and preprocessed. What kind of distance measurements could be applied to this type of data. Is mouse authentication a technique good enough to distinguish people from each other.

Second, how can mouse authentication be used. Is this a good technique alone or can it best be used as an addition to an other technique like passwords.

1.6 Contributions

This thesis focuses on mouse authentication, which in it self is a field in an infant stage. The focus is to contribute with new ideas on what data should be collected for such a propose, how it should be analyzed and suggest different ways to further work with this subject. We use the Levenshtein algorithm to implement custom distance metrics to use for mouse authentication.

2 State of the art

Humans have a remarkable way of recognize other people. We look at other people and immediately recognize a person based on several features such as face, gender, height and so on. Some factors can reduce the probability that people accurately identify a person. Stress, attraction and inattentiveness are example of such factors. Automated systems for authentication and identification of people are being researched and used. Such systems falls into a category called biometrics [1, 2]. Biometrics aims at removing the subjectivity posed by humans.

Biometrics is a statistical method that can be applied to any living organism or population [3]. There are in principle three ways to identify someone:

- by something they know (i.e. password)
- by something they have (i.e. a key card)
- by something they are (i.e. a biometric feature such as the face or signature)

Biometrics means measuring a persons behavioral or physiological characteristics in order to distinguish people from each other. Biometrics should be much harder to cheat than a password, since mimicking such characteristics is much harder that stealing a password [3]. Making a biometric system solid inhabits many challenges. The data that is needed has to be collected. After the collection this data must be analyzed. At the end a decision will have to be made by the system whether a person is the one who they claim to be. This process is illustrated in Figure 1.

The aim of such systems is to be able to produce results with as few false positives (called False Match Rate or FMR) and as few false negatives (called False Non-Match Rate or FNMR) as possible. False positives means that a system wrongly gave someone access. False negatives means wrongly denying someone access. These errors will be produced because biometric systems are based on probability. The first data that is entered into the system (also called enrolled) will be used as a baseline for subsequent identification. If this data is improperly collected this may result in a user being wrongly denied access. To minimize this problem there is a quality checker in most biometric authentication systems. Figure 1 illustrates this. Systems with a low amount of FMR and FNMR have been taken into use, such as fingerprint systems. Many systems are under research, such as signature systems [4] or pointing device systems [5, 6, 7, 8].

2.1 Mouse movement authentication

Mouse movement authentication can be compared to pointing device authentication systems, such as signature systems [4]. The research done in the field of mouse authentication is not that extensive. What little is done focuses on continuous authentication [5, 6, 9, 7]. Continuous authentication means authentication of the user over time, collecting data from the user all the time. Very few [8] focus on static authentication using the mouse.

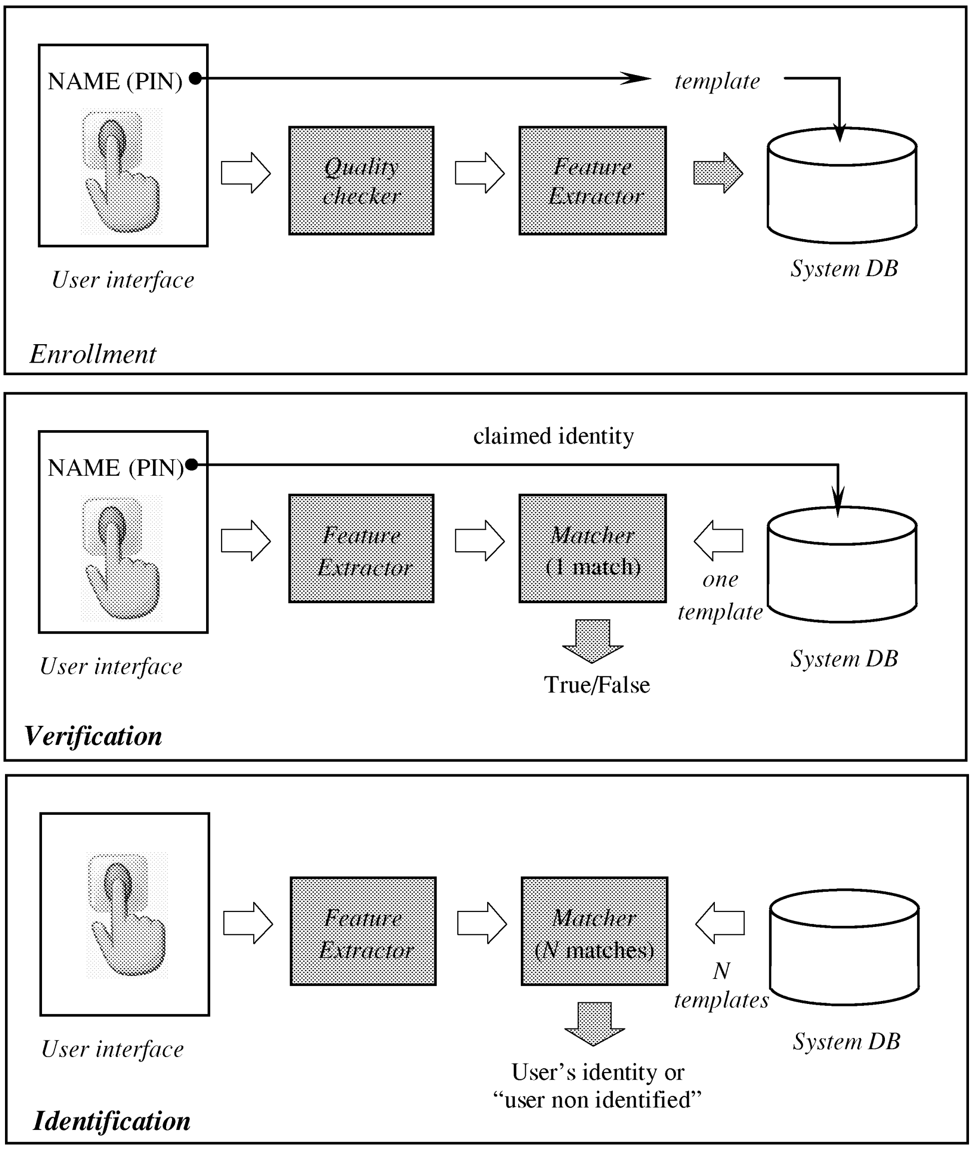


Figure 1: Block diagrams of enrollment, verification and identification. Figure taken from [2]

2.1.1 Data collection

The system presented in [8] uses a web-based environment to collect the mouse movements of the user while he or she inserts her user name and PIN code. The interface used to collect this data is presented in Figures 2 and 3.

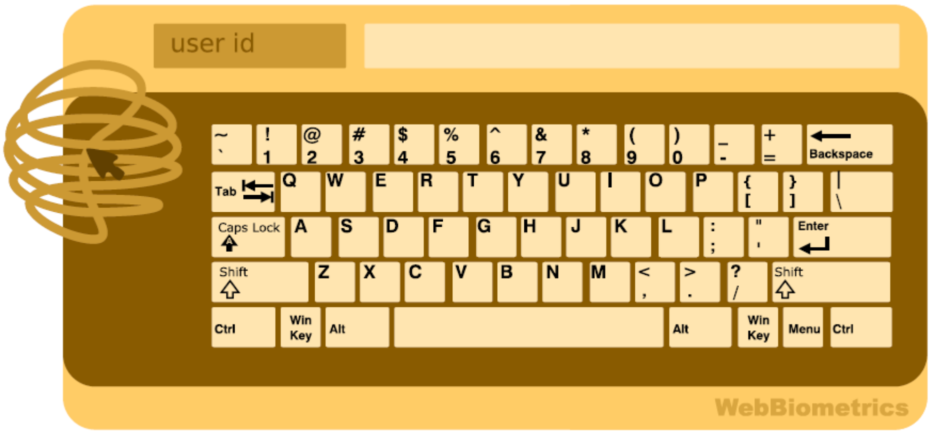


Figure 2: Virtual keyboard used to collect user name

The keyboard used for PIN input have randomly placed numbers. This approach creates robustness against automated attacks. Even if the user interaction is recorded, it cannot be replayed by using this technique. During the enrollment phase the user is asked to type his or hers PIN code three times.

The systems used to acquire this data is called the WIDAM (Web Interaction Display and Monitoring) [10]. This system allows the usage of an interaction recording system over a web page. WIDAM collects a list of events composed of: the id of the event (see [10] for details), the time instant of the event, the coordinates of the mouse (mouse movements) and the state of the modification keys (ALT, CTRL and SHIFT). To study the mouse biometric, a memory game was designed. This game has similarities to entering a PIN code. The memory game can be seen in Figure 4. The idea is that users are more cooperative in data collection if this is done in a game like environment as opposed to

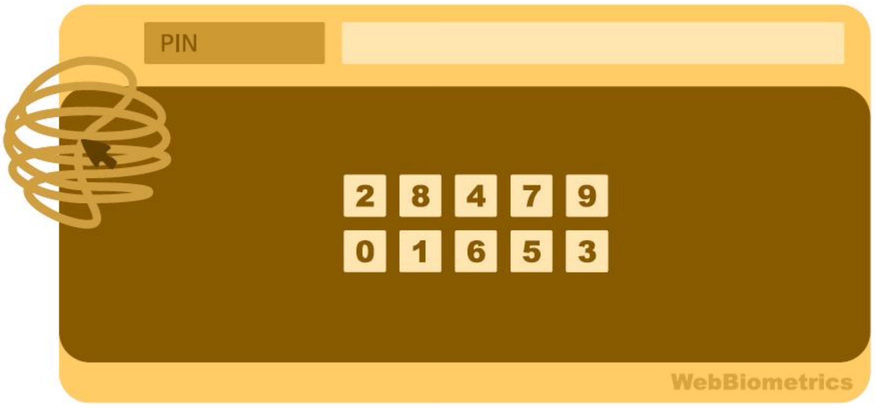


Figure 3: Virtual keyboard used to collect PIN code

just click on a number. The resulting graph of a user movement can be seen in Figure 5.

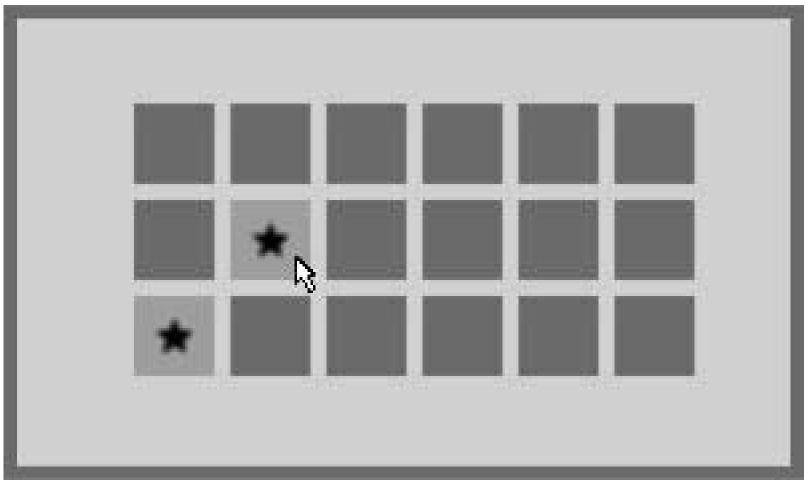


Figure 4: Memory game

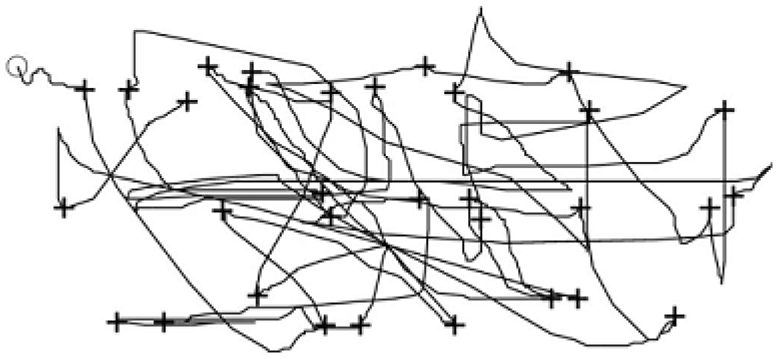


Figure 5: Data from memory game

The system defines mouse movement between successive clicks as *strokes*. These strokes are then associated with the length of the PIN code give by the user. A feature vector is then produced containing spatial (related to angle and curvature) and temporal (related to duration, position, velocity and acceleration) characteristics of the strokes. Results from this system shows that longer PIN code improves EER (Equal Error Rate). The EER for a 10 digit PIN code is 12.5% and 6.2% for a 15 digit PIN. In this study ([8]) there were 50 participants all using the same computer for the tasks. The system does not alter the normal login prcess of a normal online security system, only a software modification to the login protocol. This security layer is hard to circumvent since behavior is very difficult to mimic.

In [5] the term re-authentication is discussed. This means to continuously monitor a users actions. The system will raise an alarm if a users actions deviates from what is normal for that user. Continuous password authentication is compared to continuous mouse movement authentication. To have a continuous password authentication system, the system would have to ask the user for their password repeatedly. This is quite disruptive to the user, expensive and often unreliable when implemented. The system propose in [5] builds a model of a user's behavior from their mouse movements. It is discussed that this model should be updated over time so that unavoidable gradual changes are captured. This would avoid increasing the FNMR. The experiment done with this system uses eleven participants. Data is sampled (captured) of intervals of 100ms in this system. The basis for this sampling time is based on the statement that 100ms is a very short time for a human. A human might not move a mouse even a pixel in this period. The data that has been collected are mouse movements and mouse events. A classification of this system's mouse events can be found in Figure 6. The system presented in [6] has a

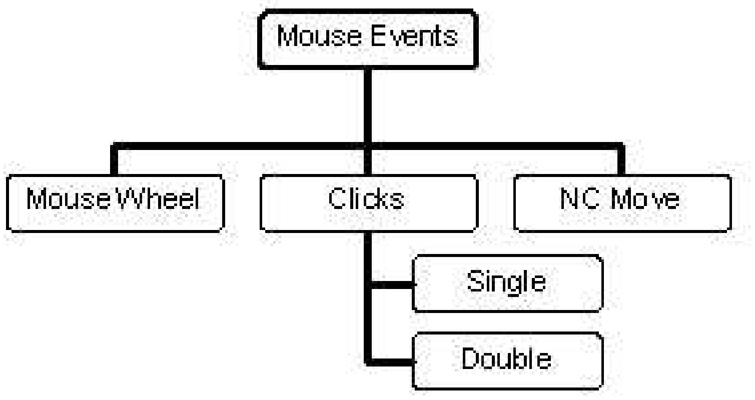


Figure 6: Types of mouse actions (NC means Non-client area movement)

similar approach. Grouping mouse events (or actions) into four different categories.

- Mouse-Move (MM): general mouse movement
- Drag-and-Drop (DD): the action starts with mouse button down, movement, and then mouse button up
- Point-and-Click (PC): mouse movement followed by a click or a double click
- Silence: no movement

In this system the classification system is taken further with the introduction of direction based grouping of events and movements. Figure 7 illustrates this approach. Total there are 8 groups, each responsible for mouse actions and movements performed within a 45-degree area.

Features extracted from the raw data in [5] are distance, angle and speed between pairs of data points. Then the mean and standard deviation are extracted from this data. The last parameters mentioned are calculated over a window of N data points. The mouse events are first grouped together and then split into categories according to Figure 6. A learning algorithm is then used to produce a template for a user base upon these features.

2.1.2 Noise reduction and environment variables

In [6] the matter of noise reduction is discussed. Incorrect readings can happen because of software or hardware error. The need for a filtration process to eliminate such problems is discussed. Such a operation should be done before submitting the data to the

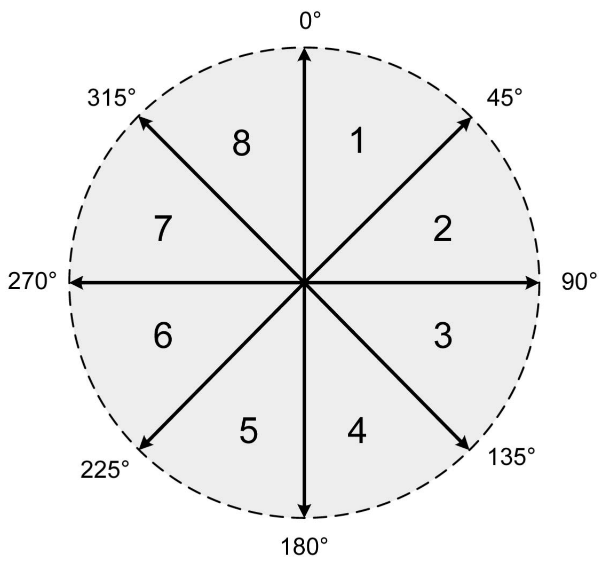


Figure 7: Direction based grouping of mouse actions and movements

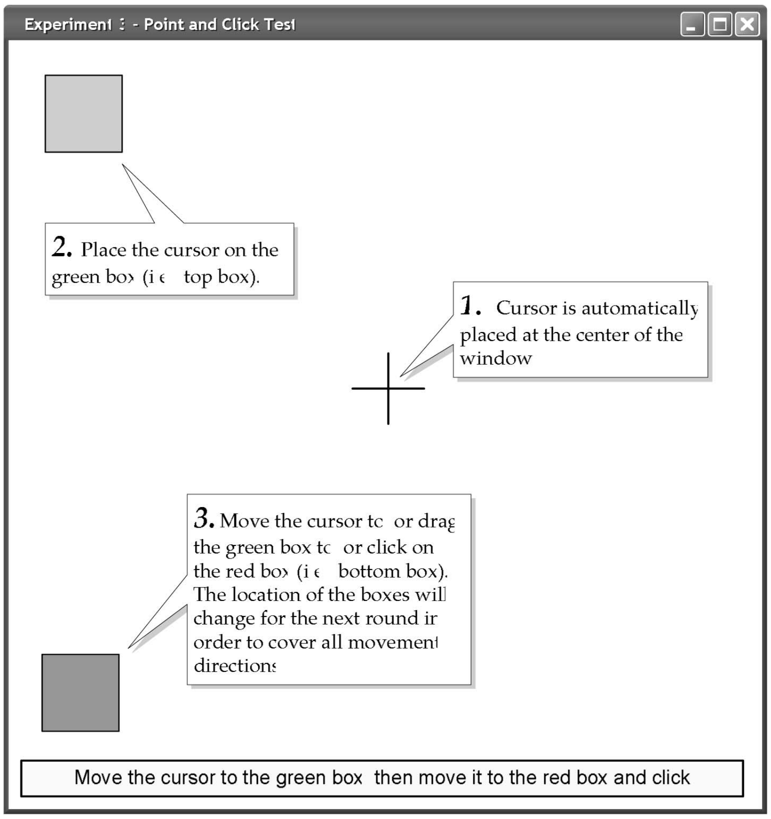
evaluation stage. This paper also sets a bound on the data to reduce effects of different screen resolutions. The maximum distances obtained by any user was set to 900 pixels. The reason for this is that a small mouse movement gets amplified on the screen. The amplification factor varies according to the screen resolution. The filter discussed here drops the distance data above 900 pixels. A different approach was also discussed. This involves normalizing the data to a specific screen resolution. The reason this was not used is that the amplification factor between the mouse movement and the screen size or resolution then will be altered. This makes the data for different users not comparable in the same measuring units (pixels). Limits for the filter that filtered speed was also used. Speeds over a certain threshold were considered as noise.

Two different filters are applied to the data. The first is a threshold process, where all data in the x-axis above a threshold are cut. The second filter is used to eliminate any y-axis (movement speed) data that highly deviates from the mean of the y-data.

The first experiment performed in [6] lets the participant be completely free. This experiment did not care what type of software or hardware was used. Two other experiments were designed to investigate whether different hardware or software could have an effect on the results. The first experiment used the same hardware and software for a smaller group of participants. The participants were asked to browse the web. Compared to the free experiment the results from this experiment showed that on a small scale (few participants), different hardware or software does not seem to have any effect. The second experiment enforced the same hardware and software, but also that participants should do a task in the same manner. Figure 8 illustrates this task. The results from this experiment suggested that fixed actions can be used to discriminate effectively among a small group.

2.1.3 Performance of mouse authentication systems

The performance of mouse biometric systems seems to wary depending on sample size. Table 1 taken from [7] shows this. The sample size in this table described the number



2. Place the cursor on the green bo› (i ‹ top box).

1. Cursor is automatically placed at the center of the window

3. Move the cursor tc or drag the green box tc or click on the red bo› (i ‹ bottom box). The location of the boxes will change for the next round in order to cover all movement directions

Move the cursor to the green box then move it to the red box and click

Figure 8: Fixed task for participants

of mouse movements collected, the number of coordinates stored. The same paper also

Sample size	EER
60	24.3%
120	20.6%
300	16.6%
600	13.8%
1200	11.8%
2400	10.9%
3600	11.1%

Table 1: Equal error rates for each sample size

states that the minimum and maximum EER's for individual users fluctuates highly. The reason for this was not clear and needed more analysis and research. See Table 2. In [6] a

Sample size	Min EER	Max EER
60	1.0%	37.0%
120	0.0005%	35.2%
300	0.0%	30.8%
600	0.0%	30.9%
1200	0.0%	38.0%
2400	0.0%	45.0%
3600	0.0%	44.0%

Table 2: Min and max EERs for individual users

comparison with their system and other biometric authentication systems are presented. Table 3 shows the results. Results show that mouse authentication using their system performs better that face or voice [11] authentication systems. In addition to Table 3,

Biometric	FNMR	FMR	Reference
Face	16%	16%	[12]
Voice	7%	7%	[12, 13]
Keystroke	4%	0.01%	[14]
Hand	3%	0.3%	[12]
Fingerprint	2%	0.02%	[12, 15]
Iris	0.25%	0.0001%	[12]
Mouse	2.46%	2.46%	[6]

Table 3: Mouse authentication systems compared to other systems

the system presented in [5] have produced a FMR of 0.43% and a FNMR of 1.75%.

2.2 Social exclusion

In all forms of biometrics the possibility that someone cannot use the authentication method has to be considered. Biometrics focuses on what a person is. These characteristics will have to be captured in some way. This means that every person that is going to enroll in the system will have to be able to perform the task asked. Examples of tasks could be writing a signature [4], voice recognition [11] or using a pointing device [5, 7, 10]. To enroll in a speech recognition system a person must be able to speak. To enroll in a signature system or pointing device system a person must be able to use the apparatus to record the data. For systems that are meant to cover large groups of people, capturing the biometric feature may present a major problem. Small rates translates into considerable figures. For instance in a system trail in the United Kingdom, 0.62% of the disabled sample group were unable to enroll any data at all [3]. This translates into approximately 62,000 of the UK population unable to enroll into the system. This means that for all these people the service connected with this system will be unavailalbe. This shows that it is very difficult to design a system that works well for all behavioral and physiological characteristics. The systems can only deal with people who are defined as ”normal” by the system. The enrollment problems will have to be fixed for many systems to avoid these kinds of problems.

Some groups of society are in higher risk than other to be disadvantaged by the use of biometric authentication systems. The list below is taken from [3]:

- People with physical and/or learning disability
- People with mental illness
- The elderly
- People of certain races (for facial recognition)
- People of certain religions
- The homeless

[3] also states that some work has been done for some of the groups, but that more is needed more most of them.

The purpose of a biometric system is to exclude people from having access to something that they should not. Social exclusion would however be seen as an unfair restriction. Since it is very difficult to make biometric systems that can handle all behavioral or physiological characteristic, it becomes an ethical question. Willingness to sacrifice some people in the interest of others. It would be wrong to argue that the rights of the many outweighs the rights of the few. Contrary should the public interest be balanced against other single individuals. This means that if a person is disadvantaged by the use of biometrics in a given situation it must be weighted against the disadvantages of not using biometrics for a person who can use it in that situation [16, 3].

2.3 Edit distance / Levenshtein distance

The analysis of data with different sample lengths offers a challenge. Data with different sample lengths cannot directly be compared without some form of adaptation or re-sampling. The Levenshtein distance algorithm is able to find the optimal alignment between two time series. This makes the algorithm ideal to use when comparing data from a mouse authentication experiment. Data from these types of experiments are rarely or never of the same lengths. The Levenshtein algorithm is also a common distance function used in string matching [17]. It was invented by Vladimir Iosifovich Levenshtein in 1965 [18], hence the name of the algorithm. The Levenshtein distance is also often called *edit distance*.

The distance between two series x and y is the minimal cost of a sequence of operations that transform x into y. The cost of a sequence of operations is the sum of the costs of the individual operations [18]. Table 4 outlines the operations that can be performed to turn one series of data into another when using Levenshtein distance [18]. There ex-

Operation	**Description**
Insertion	i.e. inserting the letter a
Deletion	i.e. deleting the letter a
Substitution	for $a \neq b$, i.e. substituting a by b

Table 4: Levenshtein distance operations

ist another operation; transposition, which involves swapping adjacent letters, however Levenshtein distance does not use this operation. The cost of all operations in Levenshtein distance is 1. This means the minimal numbers of operations needed to make two series of data equal.

To better explain how the Levenshtein algorithm works, an example will now be given. This example will describe how to transform the word *plaster* into *mystery* using the Levenshtein algorithm. There are several ways this can be done with the operations available. I will describe some ways below.

- Delete $p \rightarrow$ substitution of l by m and a by y $\rightarrow$ insert y to the end $\Rightarrow$ a total of four operations.
- Delete *pla* $\rightarrow$ insert *my* and $y \Rightarrow$ this would however give 6 operations.

The the transformation by 4 operations is not unique although the lowest numbers of needed operations is unique.

Appendix F contains the pseudo code for the Levenshtein algorithm.

3 Experiment setup

During the decisions on the experiment setup, ideas from a previous project performed fall 2007 [19] were looked at. In this experiment the users were asked to perform 3 tasks; follow the labyrinth, connect the dots and freely draw a star. After the data was analyzed the two most promising tasks showed to be the labyrinth, and the connect-the-dots. The labyrinth produced data that was very visual according to the layout of the labyrinth. Contrary the dots task allowed users freedom to more or less use their mouse as they normally do. Based upon the previous project, the two tasks labyrinth and connect-the-dots were chosen. Each test has however undergone a revision, which will be better described in Section 3.1.1 and Section 3.1.2. A program was developed to present the users with these tasks and collect the data. The program will will be described in more detail in Section 3.1.

The number of participants for the experiments was 47. This group included people that are skilled and people who are not used to use a mouse. Although the group of people skilled in using a mouse were significantly larger. The group of participants consisted of people from Norway and The United Kingdom. Participants were given the program and their personal user file, along with an explanation (in both Norwegian and English) on how to perform the tasks (these documents can be found in the Appendix C and D). These documents also informed the participants on how often they should perform the tasks and how the data should be delivered. All data the participants produced were emailed directly to the author of this thesis.

For the practical part of the experiment the participants were asked to run trough the two tasks 5 times each for every session they performed. They were asked to perform 6 sessions. This gives a total of 30 runs trough each of the tasks. The participants were asked not to perform all the sessions consecutively, but rather wait for a few days, for example 1-2 day(s) between each session. This spreads the experiment period over 1-2 weeks. The idea behind this approach is to minimize problems like learning curve artifacts. The learning curve will probably never disappear completely, specially for the labyrinth task this will introduce itself. The participants will most likely learn the labyrinth better and remember it better between each session. The case is a bit different for the dots test, since the dots are randomly placed each time. The dot to focus on is also randomly chosen every time. For a more in depth discussion, see Chapter 6.

3.1 Mouse logger program

The program were written in Java [20]. This was to make it compatible on the most common platforms and systems used today. The systems covered by Java includes Microsoft Windows [21], Linux [22], Sun Solaris [23] and Apple OS X [24]. This showed to introduce an unexpected problem on Apple OS X, were the program performed flawlessly on OS X Tiger [25], but did not work as it should on the newest OS X Leopard [24]. The result on OS X Leopard was that buttons and interaction parts of the program became invisible, making it hard to impossible to interact.

The program is supplied with a user file, which tracks the progress of the user (for more details on this file, refer to the Appendix A). This information makes it easy to identify both the data and the user, and what stage the participant is on. It also helps the user to confirm what stage of the experiment that they are on, displaying information on user number and current session. This is illustrated in Figure 9.

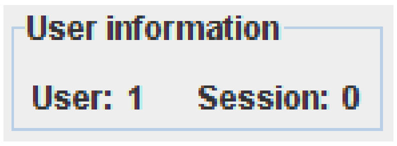


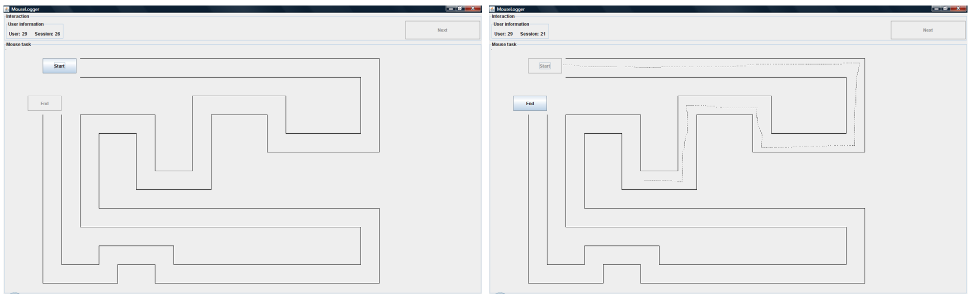
Figure 9: User information

The program stores the [X, Y] position of the mouse pointer, as well as the time that this position was recorded. The program samples this data at a rate of 100 times per second, and stores it in *.md files (format described in Appendix B). The files are named to easy inform of which user, what session and what type of task the data residing the the file is from. The naming is done like this: UserN_SessionM_TASK.md, where N is the user number, M is the current session and TASK are either labyrinth or dots depending on the task performed.

3.1.1 Labyrinth test

Figure 10 displays the labyrinth that is presented to the users, as well as the overall layout of the program. The idea with this test is for the participant to follow the labyrinth with the mouse pointer, from start to end. A dotted line will follow the mouse pointer to indicate that the participant is now running the test.

The data collected from this test have proved to be very visual, this means that it is easy to see from graphs where a user is at any time in the test. An example of the raw data produced by this task is illustrated in Figure 11.



(a) Clean labyrinth (b) Task under way

Figure 10: Labyrinth task presented to participants

3.1.2 Dots test

Figure 12 illustrates an example of the dots test. The dots are randomly placed each time, trying to minimize learning curve issues. The dots have colors to explain to the user what the participant should do:

- red means dots that have not been clicked yet.

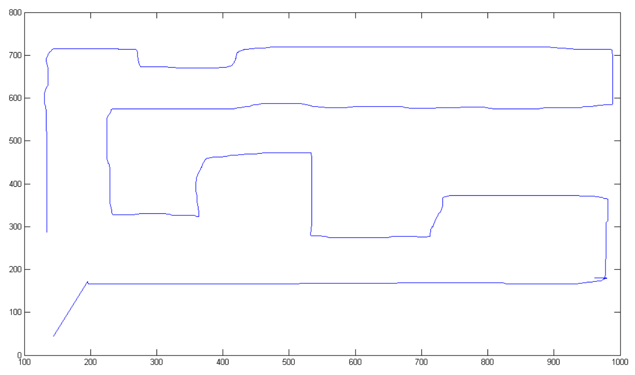
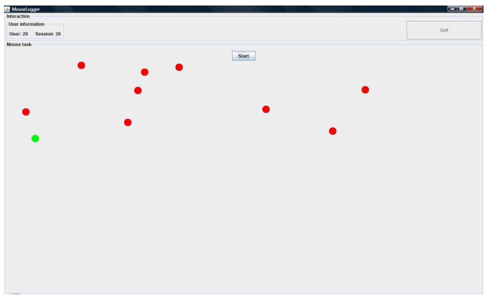


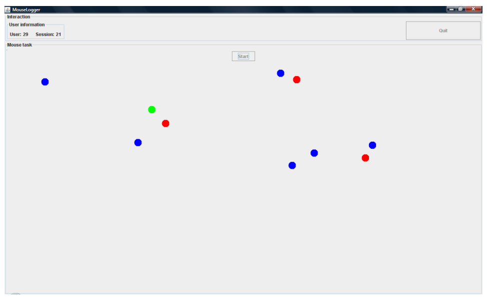
Figure 11: Raw data from the labyrinth task ([X, Y])

- green represents the dot that the user should move to and click next.
- blue represents dots that the user have clicked.

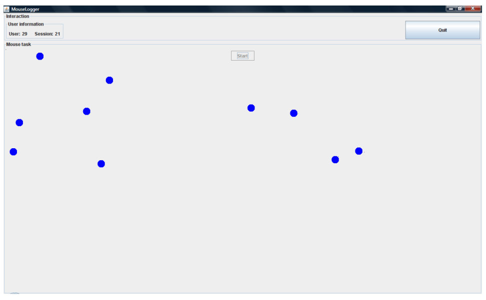
This task resembles the virtual keyboard used in [8] or the memory game used in [10]. The dots are however randomly placed and so is the next dot the user should click on. Making this test even more random than a memory game. An example of raw data from the dots task is illustrated in Figure 13.



(a) Clean dots task



(b) Task under way



(c) Task finished

Figure 12: Dots task presented to participants

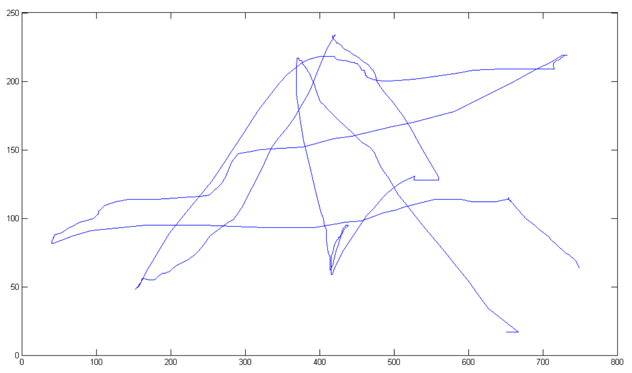


Figure 13: Data from dots task ([X, Y])

4 Algorithms and data preprocessing

This chapter will describe algorithms used to preprocess the data. This preprocessing is done to extract features for the distance metric to measure. Feature extraction is described further in Section 4.2, while the distance metrics used will be better described in Chapter 5. Before these features can be extracted, Moving Average (MA) or Weighted Moving Average (WMA) are used for noise reduction. These are described in Section 4.1. The data that is collected are position [X, Y] and time. Noise reduction is done on this data, and velocity is calculated. The formula used to calculate velocity data is described in Equation 4.1.

$$v_i = \frac{\Delta x}{\Delta t} = \frac{x_i - x_{i-1}}{t_i - t_{i-1}} \tag{4.1}$$

The transformation to velocity makes it possible to determine horizontal and vertical movement in the labyrinth. Section 4.2 describes this in detail.

4.1 Data filtering

When the speed data are calculated from the raw data supplied by participants, this generates very noisy signals. Figure 14(a) illustrates an example of an original speed signal. To reduce this noise two filters are applied to the data before velocity calculation.

Two different filters were applied to determine which one produced the best result in terms for later feature extraction algorithms. The first filter was the Moving Average (MA) filter. Formula for MA filter is described below.

$$x_i^{`} = \frac{x_{i-2} + x_{i-1} + x_i + x_{i+1} + x_{i+2}}{5}$$

The effect of this filter on an original signal can be seen in Figure 14(a) and Figure 14(b).

The second filter tried was the Weighted Moving Average (WMA) filter. This filter differs from the MA filter only in that it applies a weight on the values according to their distance from the value to be calculated. The value itself bear the most weight while the values on each side bear less weight. The formula for WMA filter is described below.

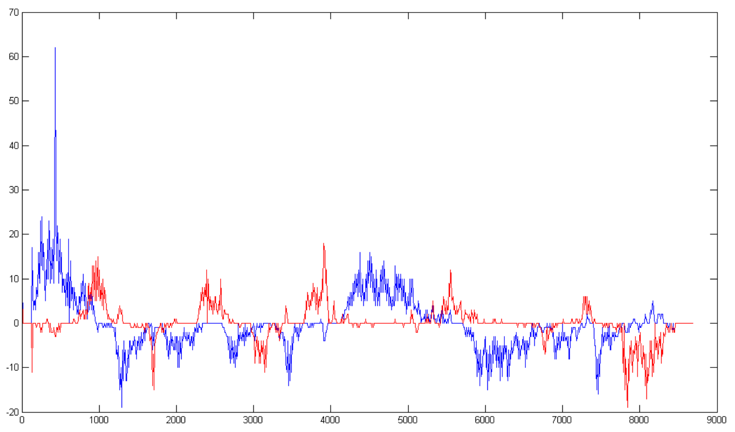
$$x_i^{`} = \frac{x_{i-2} + 2x_{i-1} + 3x_i + 2x_{i+1} + x_{i+2}}{9}$$

The divisor (9) in the previous expression is the sum of all the weights $(1 + 2 + 3 + 2 + 1 = 9)$. The effect of this filter on an original signal can seen by comparing Figure 14(a) and Figure 14(c). It is possible to use different filter lengths of both MA and WMA filters, but in this thesis the length of 5 are used.

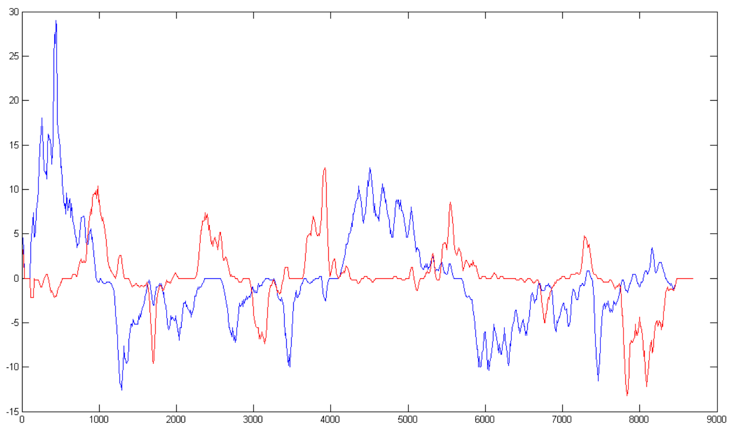
4.2 Feature extraction

This section will describe the features extracted from the data for the labyrinth. All feature extractions are performed on velocity data.

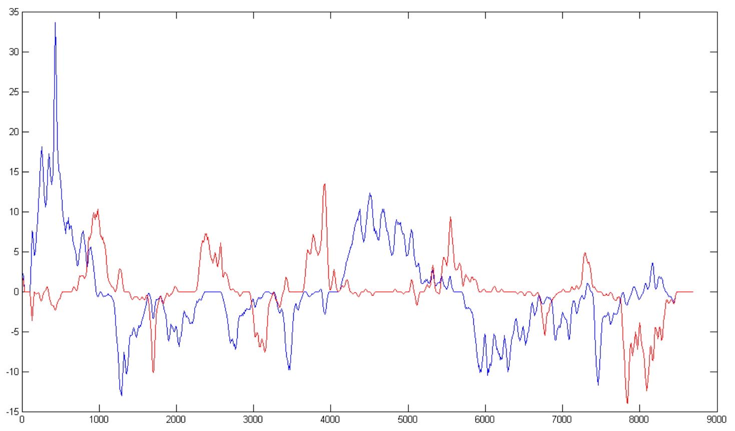
The layout of the labyrinth is illustrated in Figure 15. The features extracted from the speed signal for the labyrinth are:



(a) Original signal



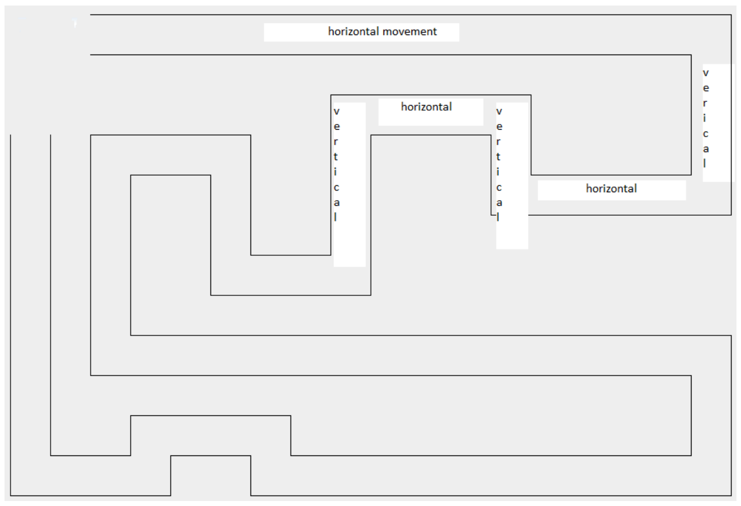
(b) Signal applied MA filter



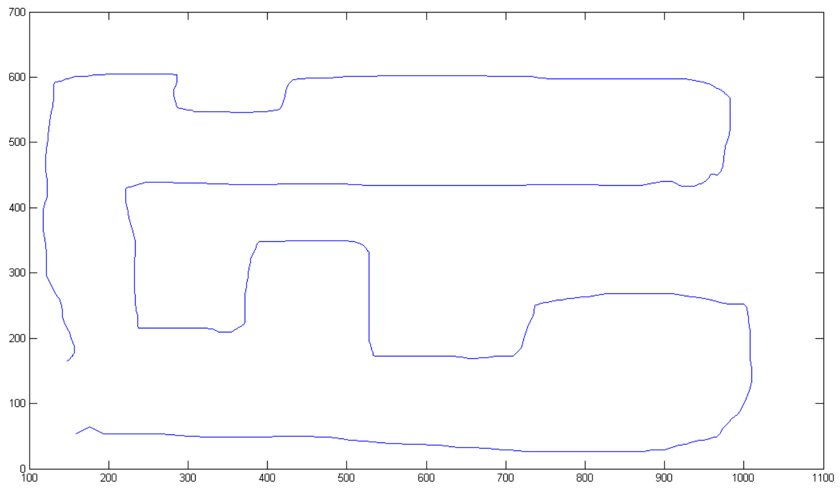
(c) Signal applied WMA filter

Figure 14: Original signal applied with MA or WMA filter

- extract tracks from the signal
- calculate the variance for the other direction of the track
- categorize horizontal and vertical moves



(a) Labyrinth layout



(b) Position data from performed labyrinth task (flipped because the smallest coordinates starts off in the beginning of the labyrinth)

Figure 15: Labyrinth layout and labyrinth performed by participant

The first task is to classify the data in either horizontal or vertical directions. These horizontal and vertical movement paths will be named *tracks*. This is clarified in Figure 15(a). From this figure we also see that there are 9 horizontal and 9 vertical tracks. Each horizontal track will have a high speed in the x-direction while the changes in the y-positions are only minor, i.e. the speed in the y-direction will be low. The opposite will be true for the vertical directions. This information is used in the ”Track separation algorithm”, described in Algorithm 1. If the speed in the x-direction has the highest absolute value, then the data point comes from a horizontal track, otherwise from a vertical track.

Algorithm 1 Track separation algorithm

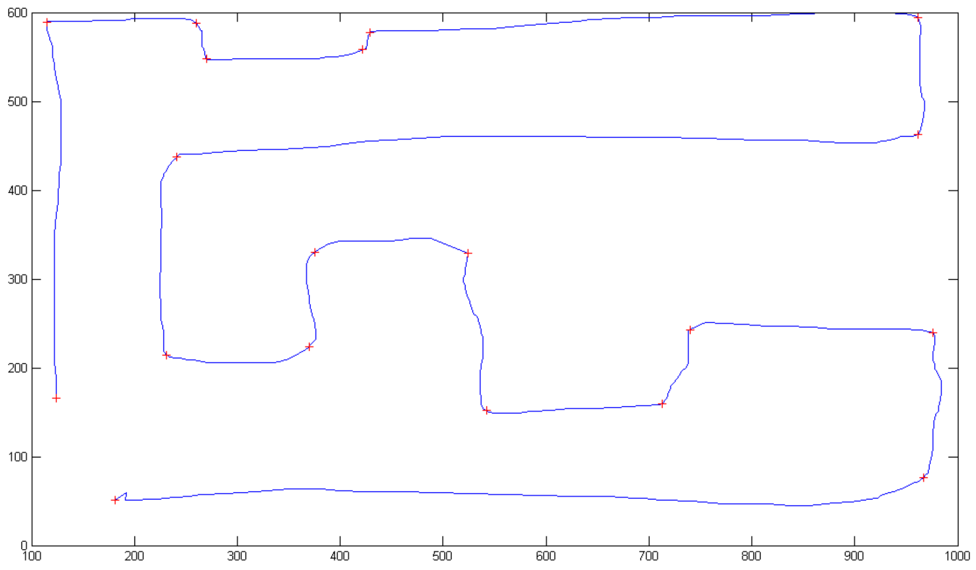
```
transition = true
for $i = 1$ to *number of samples in data* do
  if transition is true then
    if $|Y| > |X|$ then
      if length of track < minimum length then
        continue loop
      else
        add $i$ to array (transition found)
        transition = not transition
      end if
    end if
  else
    if $|X| > |Y|$ then
      if length of track < minimum length then
        continue loop
      else
        add $i$ to array (transition found)
        transition = not transition
      end if
    end if
  end if
end for
```

- If $|x_i| > |y_i|$ then $class_i = horizontal$
- If $|x_i| < |y_i|$ then $class_i = vertical$

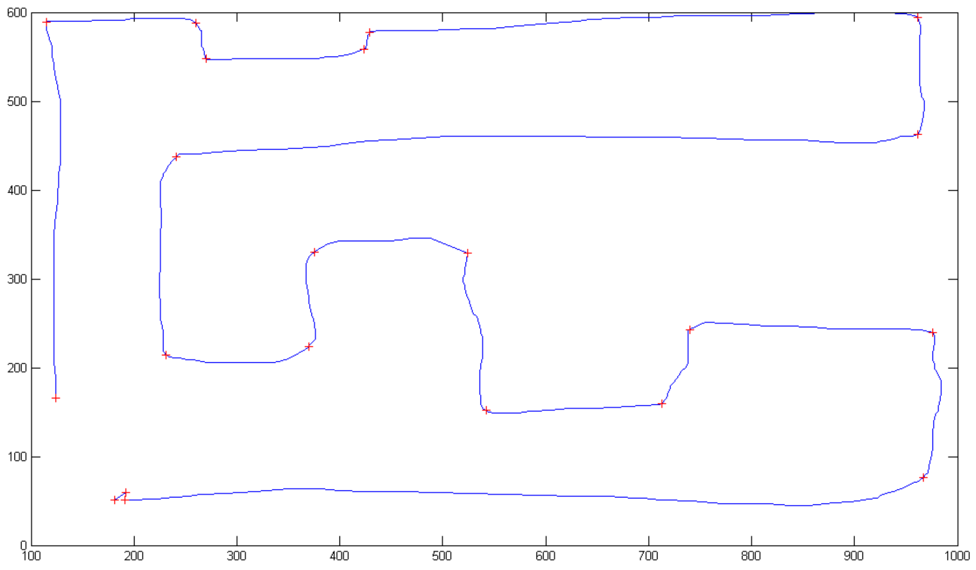
Obviously, when changing from a horizontal to a vertical track and vice versa, data points could be classified incorrect. For this reason a threshold will be used for the minimal length of a track. The ”Track separation algorithm” returns the locations where the transitions from horizontal to vertical movement and vice versa take place. The track itself now is the set of data points between two consecutive transition locations.

Figures 16 and 17 illustrate the results of the automatic track separation. In the figures the separation locations are marked with a red +. For each track in a specific direction, the variation of the speed in the other direction is also calculated and can be used as a feature for the distance metric used for authentication. This will be described more in Chapter 5. Figure 16(b) illustrates the effect of not using a threshold. This results in the algorithm detecting possible short and invalid transitions as actual transitions. We do not want this to happen, therefore a threshold of the minimum allowed length is used to exclude this from happening.

Both Figure 16(b) and Figure 17(b) illustrates occurences where the automatic track detection have failed. The first figure illustrates why threshold is important. The second figure illustrates an occurrence where the track separation fails on many occasions because the user does not alter his or hers movement very much when alternating between horizontal and vertical movement, resulting in the algorithm to detect false transitions or just not detecting them at all. Figure 17(a) illustrates an occurrence where the error described above might seem to happen, but does not. This is because the user does alter his or hers speed significantly enough during the transition parts. The way a user performs the transitions is critical this track separation algorithm.

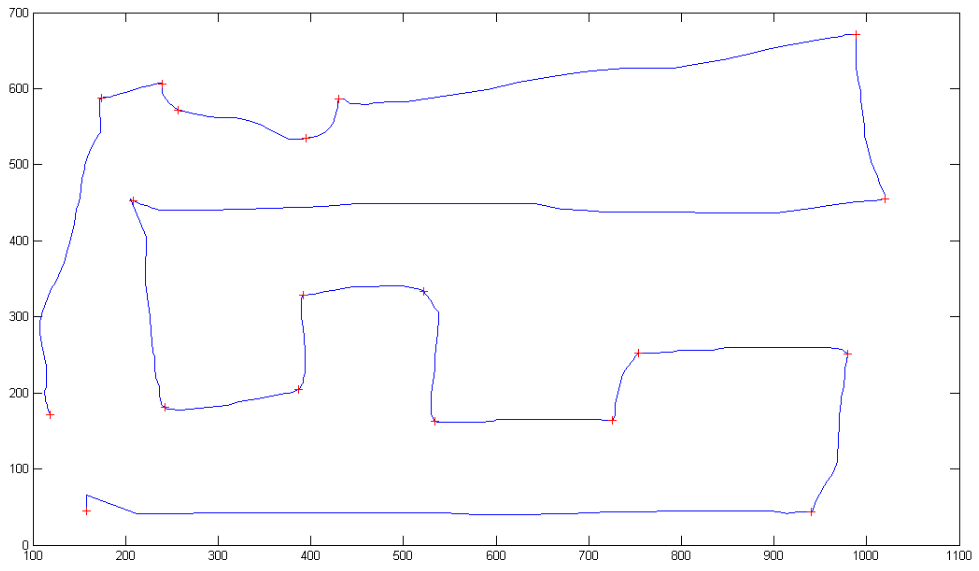


(a) Normal case

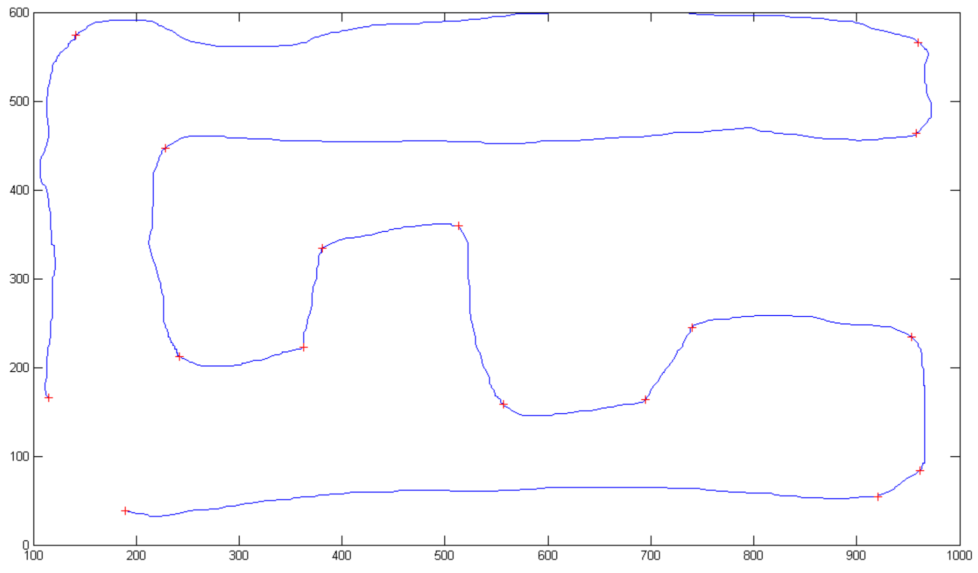


(b) No threshold

Figure 16: Horizontal and vertical detection



(a) Extreme case, correct detection



(b) Extreme case, not correct detection

Figure 17: Horizontal and vertical detection, extreme cases

5 Data analysis

In this chapter we will describe the algorithms and distance metrics used to evaluate an authentication attempt. Note that only the labyrinth task have been analyzed. Features and techniques suggested are based only for that task.

A *distance metric* determines the distance between two sequences. Suppose we have to series of points (P and Q).

$$P = (p_1, p_2, ..., p_n)$$

$$Q = (q_1, q_2, ..., q_n)$$

A distance metric will return the distance between these two series. If these two series are data from the *same user*, the distance should be *small*. If the two series are from two *different users*, the distance should be *high*. The challenge is to extract features and compare them in a way to best achieve this.

There are three established distance metrics used in this thesis:

- Edit distance.
- Euclidean distance.

The edit distance or Levenshtein distance has been discussed in Section 2.3. The Euclidean distance is best described as the distance that can be measured between two points with a ruler. Older literature may describe this as ”Pythagorean metric”. The formula for Euclidean distance is given in Equation 5.1.

$$d_{Euclidean} = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \tag{5.1}$$

An alternative to the Euclidean distance is the Manhattan distance. It is best described as walking along streets in a city, making turns at blocks until the target is reached. The total distance that was walked is the total distance between the two points. The formula for Manhattan distance is given in Equation 5.2.

$$d_{Manhattan} = \sum_{i=1}^{n} |p_i - q_i| \tag{5.2}$$

A graphical comparison of Euclidean and Manhattan distance is illustrated in Figure 18[1]. In this figure, the blue line illustrates Manhattan distance and the green line illustrates Euclidean distance. The Manhattan distance metric is not used directly in this thesis, but is described here to illustrate that this is an alternative to use instead of the Euclidean distance.

A distance metric does not have to restrict it self to use one of these. Several custom distance metrics have been designed for this thesis to try to utilize the features extracted

[1]http://en.wikipedia.org/wiki/Image:Manhattan_distance.svg (valid on 26.06.2008)

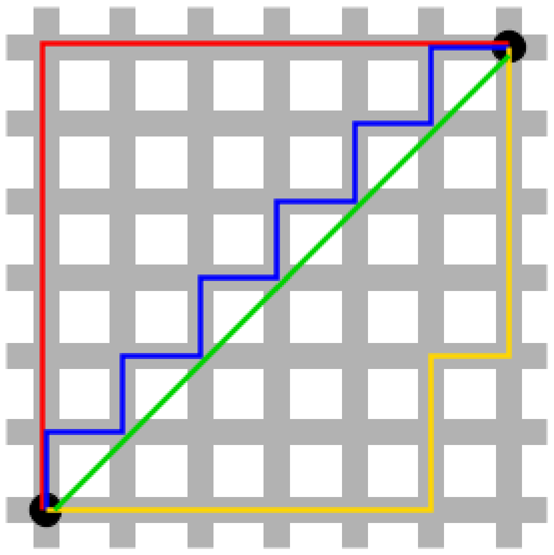


Figure 18: Comparison of Manhattan and Euclidean distance

from the labyrinth task. Section 5.1 will describe these further. The results and analysis of these simulations are presented in Chapter 6.

For a authentication system to be able to decide whether a user is who he or she claims to be, a *template* will have to be generated for each of the authentication systems users. The purpose of this template is to create a model of what is normal behavior for a particular user. The distance of this template against data from an authentication attempt is then calculated using a selected distance metric. To be able to decide if this authentication attempt is genuine or an impostor attempt, this distance is checked against a *threshold*. If the distance is lower than the threshold, it is considered a genuine attempt. Contrary if the distance is higher than the threshold, it is considered an impostor attempt.

To be able to run simulations of authentication attempts, an algorithm that chooses random templates and compares it to genuine and impostor attempts is used. This algorithm calculates the distance of genuine and impostor attempts against each randomly chosen templates and sorts them in two arrays. One containing genuine attempts and their distances to the template and one containing impostor attempts and their distances to the template. The algorithm is described in Algorithm 2.

The threshold is vital in such a system. If this threshold is set too *high*, many impostors may get access. The positive side of a high threshold is that all genuine users should get access. Contrary if the threshold is too low, some genuine users may have difficulties getting access. A low threshold should however ensure that no impostors get access to the system. Therefore the threshold will have to be simulated to find a threshold that ensure that balances these two cases. To simulate this two thresholds are chosen; one were all genuine users are accepted (high) and one were all genuine users are denied (low). Then a simulation is done with all thresholds within these two for both impostors and genuine authentication attempts. For each threshold the value of the fraction of genuine attempts that were denied access is calculated. This factor is called *False Non-Match Rate* (FNMR). The formula for FNMR is given in Equation 5.3. For the impostors value of the fraction of impostor attempts that got access to the system is calculated. This

Algorithm 2 Authentication simulation algorithm

```
for $i = 1$ to $100$ do
  choose random template for each user
  for each user session do
    if user session is not chosen as template then
      for each template do
        calculate distance of attempt against template
        if session and template are from same user then
          store distance as genuine attempt
        else
          store distance as impostor attempt
        end if
      end for
    end if
  end for
end for
```

factor is called *False Match Rate* (FMR). The formula for FMR is given in Equation 5.4.

$$FNMR_T = \frac{\text{number of denied genuine attempts}}{\text{total number of genuine attempts}} \tag{5.3}$$

$$FMR_T = \frac{\text{number of allowed impostor attempts}}{\text{total number of impostor attempts}} \tag{5.4}$$

Naturally we want as few imposers allowed access as possible; low FMR. A lower FMR may mean a higher FNMR however. A high FNMR is not wanted, so the threshold will have to be balanced. The FNMR and FMR can be plotted in a curve which shows FNMR against FMR. This curve is called the *DET curve*. In this curve we can illustrate where the FNMR is equal to the FMR, this value is called the *Equal Error Rate* (EER). A DET curve example with the line $y = x$ is illustrated in Figure 19. The point where the line $y = x$ intersects with the plotted FNMR and FMR curve is the EER. The EER factor is often used as a performance measure of a biometric authentication system. The lower the EER is, the better the performance.

5.1 Labyrinth

In this section we will outline the different distance metrics that have been used for the labyrinth task. The features extracted from the velocity data is the tracks and variance. These features have been used in distance metrics. Simple distance measurements have also been simulated. The different scenarios tested are:

- *edit distance* of noise reduced velocity data in *x* direction.
- *edit distance* of noise reduced velocity data in *y* direction.
- *edit distance* of noise reduced velocity data points [X, Y].

For all of the above the distance is calculated using this distance metric presented in Equation 5.5.

$$D = \text{edit distance}\,(\text{template}, \text{input}) \tag{5.5}$$

The custom distance metrics designed for the labyrinth works with the tracks extracted from the labyrinth data. Equation 5.6 describes the base for these custom distance metrics. In this equation the number 18 is used because there are a total of 18

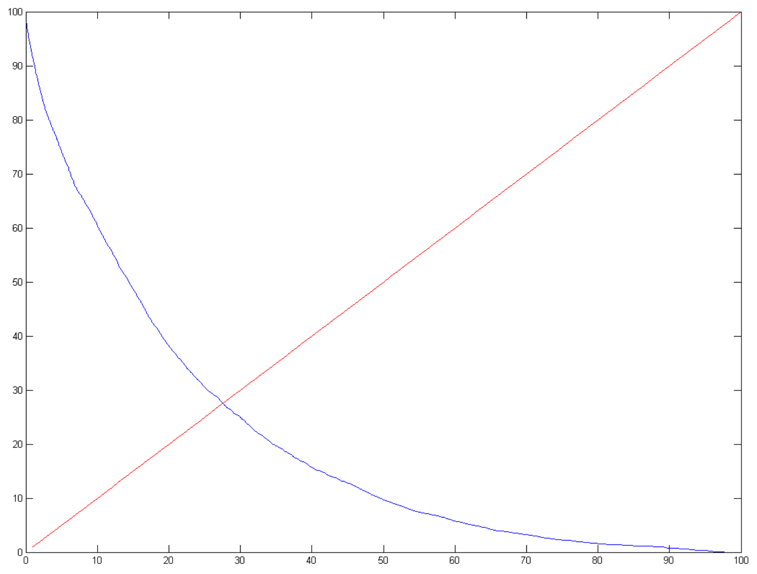


Figure 19: DET curve

tracks in the labyrinth, as illustrated by Figure 15(a).

$$D = \sum_{i=1}^{18} d_i \tag{5.6}$$

Where the parameter d_i is the distance between to single tracks. The same track is extracted from both the template and the sample and the distance is calculated. If track 1 is extracted from Figure 15(a), this is a horizontal track. This operation is continued for all tracks.

Table 5 outlines the combination of distance metrics used on a single track. The first

Edit distance using MA filtered data (whole dataset)
Edit distance using WMA filtered data (whole dataset)
Edit distance by looking at tracks
Edit distance by looking at tracks multiplied by distance of variance
Edit distance by looking at tracks multiplied by penalty

Table 5: Combinations of distance metrics on single tracks

distance metric that includes a penalty uses the distance of the variance that has been calculated of the opposing direction of the movement (we call this Variance Distance or VD). That means that if there is vertical movement, the variance is in the horizontal movement. Equation 5.7 describes the distance calculation done for each track using this distance metric.

$$d_{track} = d_{Editdistance} \cdot \sqrt{var_T^2 - var_I^2} \tag{5.7}$$

The distance metric that fist calculates the Edit distance of the tracks and then multiplies this with a penalty uses the Equation 5.8 to decide the penalty. This equation decides the penalty based upon the difference of the variance in the opposite direction of the movement. Same way as described for the previous distance metric.

$$p = f(n) = \begin{cases} 0.5 & \text{if } n < 0.1 \\ 1 & \text{if } n < 0.2 \\ 1.5 & \text{if } n < 0.3 \\ 2 & \text{if } n > 0.3 \end{cases} \tag{5.8}$$

Equation 5.9 describes how the difference that Equation 5.8 uses to decide the penalty is calculated.

$$n = \frac{|var_{template} - var_{input}|}{var_{template}} \tag{5.9}$$

For both of the two last described distance metrics the distances of the tracks are summed up to form the final distance. Equation 5.10 describes this.

$$D = \sum_{i=0}^{18} d^{i}_{track} \tag{5.10}$$

There are a total of 18 different tracks in the labyrinth, therefore Equation 5.10 uses this number.

6 Results

In this chapter we will present the results and a discussion of the results. Evaluation of proposed distance metrics will also be discussed.

The chapter is divided into three sections; full data set simulations, track feature extraction simulations and a comparison of the two techniques. The first section will describe the simulations done on the full data set of the labyrinth. The second section will describe the results from the custom distance metric using no penalty, track distance multiplied with variance distance (VD) as penalty and track distance multiplied with the result from the penalty function based on the variance.

The results presented are done with MA filtered data. Tests performed showed that using either MA filter or WMA filter seems to not have any significant impact on the performance. Figure 20 illustrates this, Table 6 describes the EER results. In both the

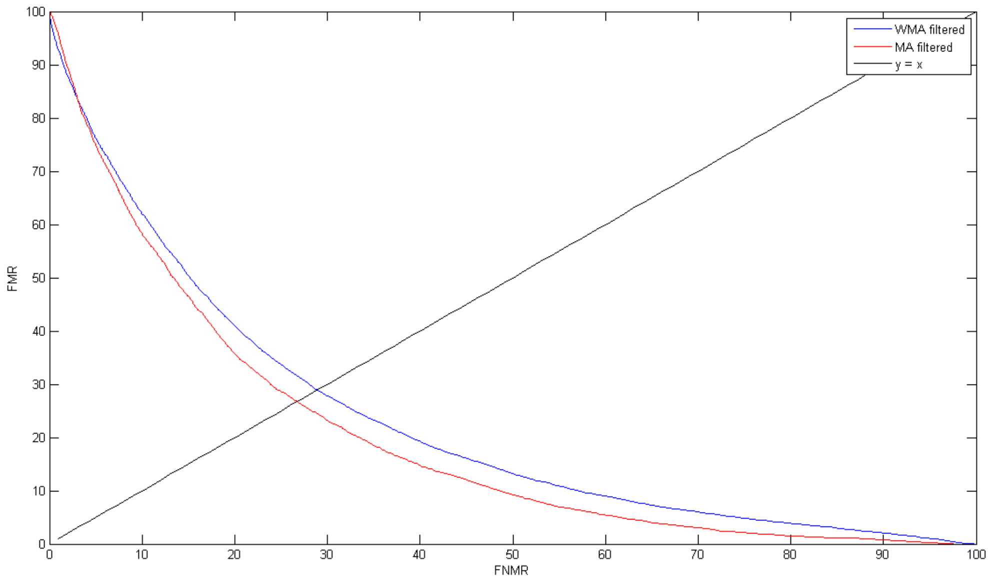


Figure 20: WMA versus MA filtering

Filter	EER
WMA	28.9%
MA	26.8%

Table 6: WMA versus MA - EER results

tests full horizontal direction data set was used. This was done to minimize any effect that feature extraction may have. We chose to use MA filter for the simulations presented in Section 6.1 and Section 6.2 because of its simple form.

6.1 Simulations on full data set

This section presents the results from simulating authentication using edit distance on velocity data in horizontal and vertical directions. The data have been filtered using MA filter. The whole dataset is used here. These simulations are slow because of the amount of data involved.

Figure 21 illustrates the DET curves for simulations using the full data set. Two simulations are using either horizontal or vertical data and one is using both horizontal and vertical data. When both directions are used Equation 6.1 is used to combine the directions. Notice that this equation is calculating the edit distance twice; first of the horizontal direction and then of the vertical direction. This calculation is slow, since the edit distance for horizontal direction is first calculated and then added with the edit distance for the vertical direction. This involves calculating edit distance twice, resulting in very long calculation time. This is not an ideal approach if speed is an important factor for a system. Table 7 describes the EER results from these simulations.

$$D = d_x + d_y \tag{6.1}$$

These results show that the EER is not heavily affected by which direction that is compared. To save computation time one could choose one of them. Perhaps investigate the best one. In this experiment the result of comparing horizontal data indicates a better EER. Although the results are to close to say that this actually is the case.

Distance metric	**EER**
Horizontal direction	26.8%
Vertical direction	27.0%
Both directions	27.5%

Table 7: EER results from simulations using the full dataset (All MA filtered)

The formula presented in Equation 6.1 presents just one way of combining the two directions. There are several ways this can be done. Some ways that this also could have been done could include:

- multiplying the two distances d_x and d_y
- the absolute value of the difference between d_x and d_y

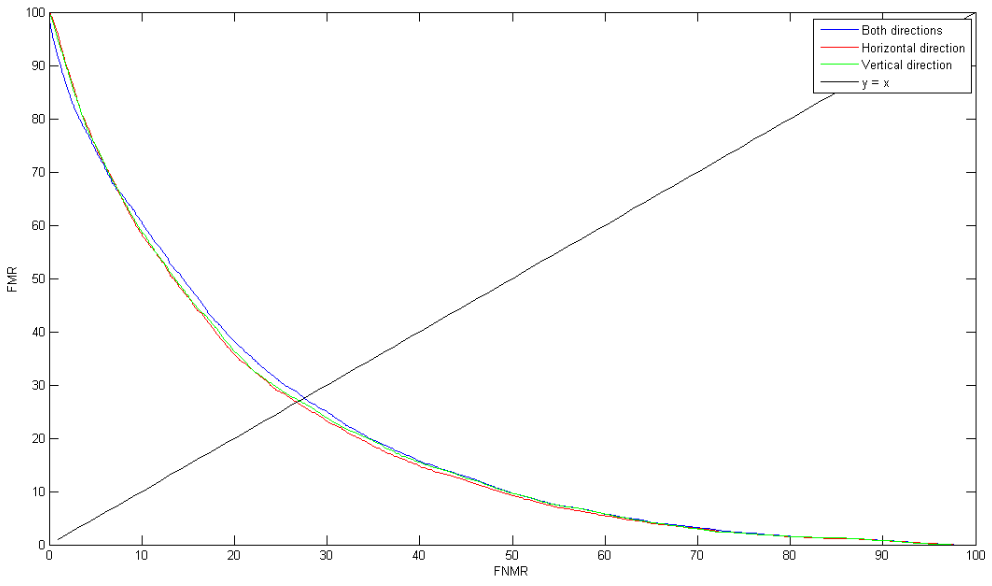
These are just other suggestions of other ways to perform this, and not actually performed in this experiment.

The EER of these simulations are general somewhat high. This could be explained that the whole dataset is used, any possible unique features are not looked at. The identification of features could improve the EER (performance). The simulations does show that there is potential in looking at velocity data for mouse authentication. Splitting the data into different directions does not seem to be important when doing the type of simulations presented here.

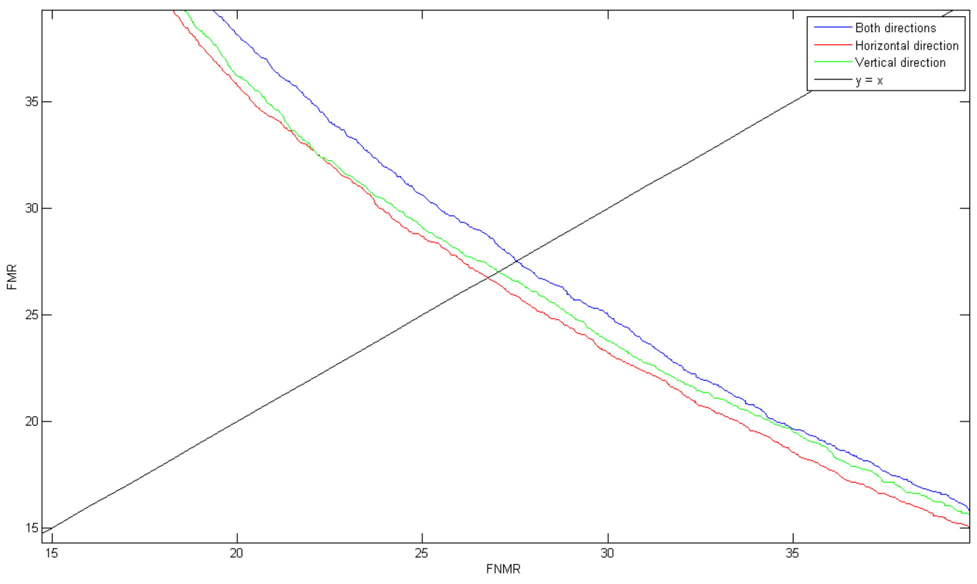
To involve both directions in this simulation without doing two comparisons, the Equation 6.2 could be used.

$$r = \sqrt{x^2 + y^2} \tag{6.2}$$

The result data series r would then be compared. The DET curve from this simulation is shown in Figure 22. This figure also contains the previous simulation presented in



(a) Full dataset



(b) Full dataset zoomed around EER

Figure 21: DET curve: Full dataset simulations

Equation 6.1 as a comparison. The EER for the simulations using Equation 6.2 was 29.0%

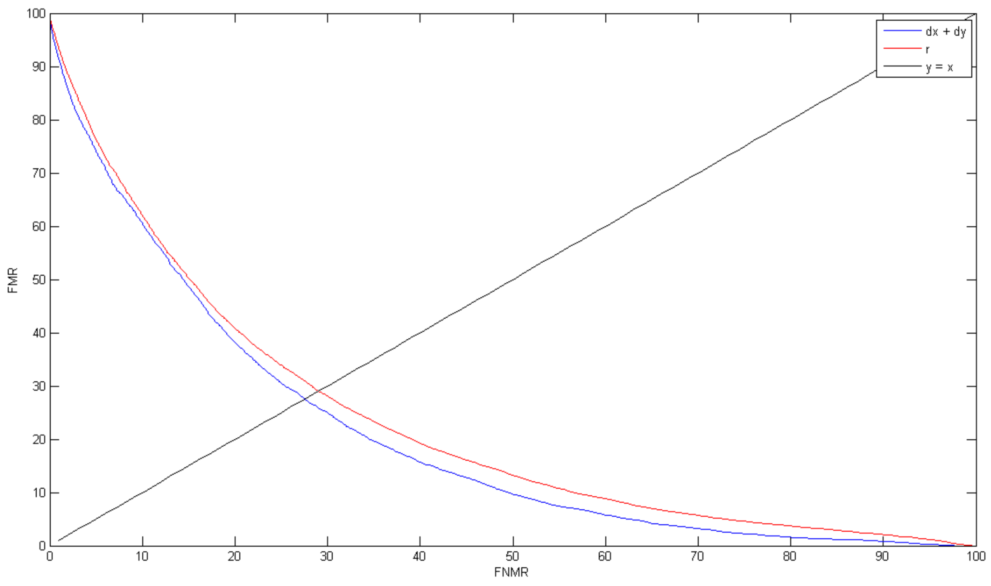


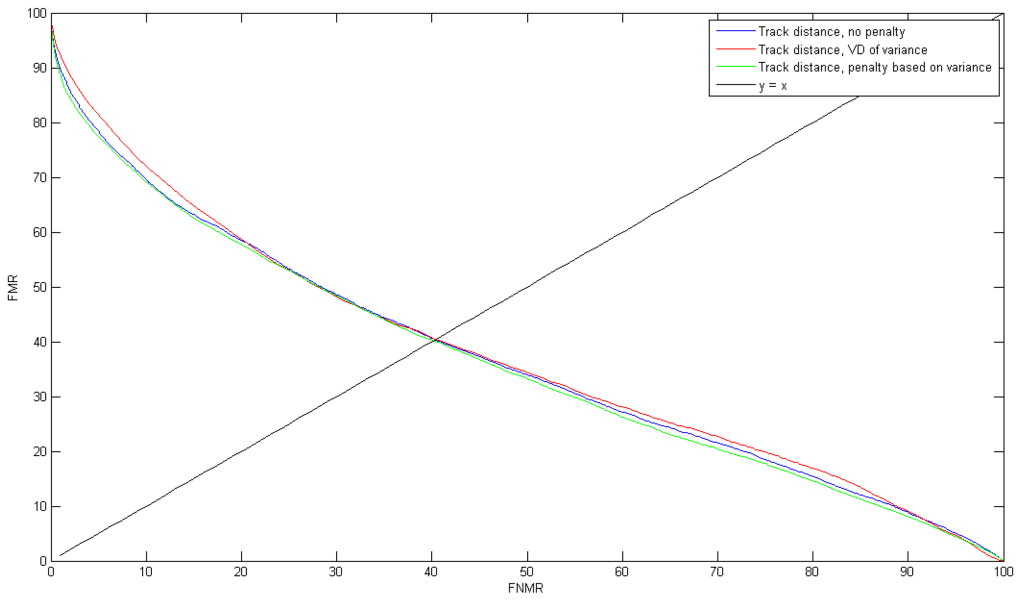
Figure 22: Both directional comparison

compared to an EER of 27.5% when using Equation 6.1. The EER presented in this figure shows a very small difference. The difference is really to small to conclude on which of the two techniques is the best. Using the Equation 6.2 does however allow for faster calculation time since the edit distance does only have to be calculated once.

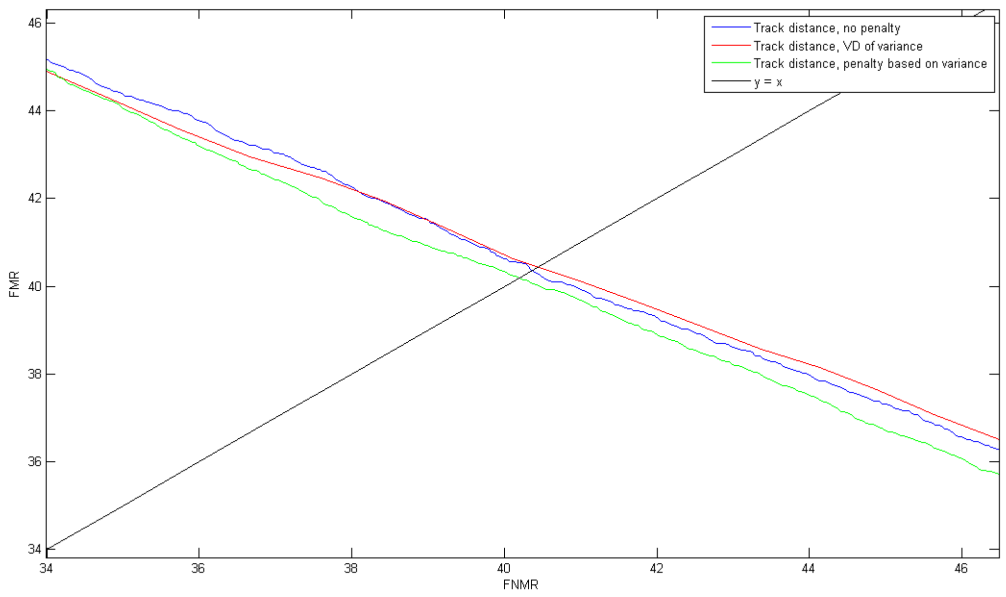
6.2 Track feature simulations

This section will present results from simulations using the extracted *track* feature. The track detection is done automatically in this thesis. Sometimes this automatic process (as shown in Chapter 4) detects tracks incorrectly. This could result in that tracks of different alignment being compared with each other. As an example of this we can present the case that a track transition is not detected in the input, but is detected in the template. The result of this is that for example a horizontal track is detected as being to long, and therefore also contains vertical data. When the correctly detected track in the template is compared to this incorrect track of the input it could result in a large distance. This problem could be increased in the distance metrics that use the variance as a penalty. If the variance is calculated it might actually be calculated for the direction the user is moving in and not for the opposite direction as it should. This would result in a high variance and hence a large penalty would be given. Ways of avoiding this problem are discussed in Chapter 8. This artifact has probably impacted the results when using this distance metric.

The blue DET curve in Figure 23 illustrates the result from the simulations where edit distance is used on the tracks. This is performed by calculating the edit distance of each track between the template and the input and then adding the distances for all tracks together forming the final distance between template and input. The EER for this simulations is higher than the results from the simulations presented in Section 6.1. This may indicate that extracting the tracks and comparing them might not be the best feature



(a) Not zoomed



(b) Zoomed around EER

Figure 23: DET curves, track distance

to look at. The other explanation of this result could be the problem with the automatic track detection. The track separation may prove to be the critical part for this distance metric to perform the best.

The red DET curve presented in Figure 23 presents the results from the simulation done where the edit distance of a track is multiplied with the variance distance of the variances as a penalty. The final distance between template and input is calculated in the same manner as the previous metric; by adding together the different distances of each track. The EER for this simulation is still higher then the results presented in Section 6.1. This distance metric is also affected by the feature extraction problem and by the resulting problem of the variance calculation, which may be a reason for the high EER.

The green DET curve presented in Figure 23 uses the function presented in Chapter 5.1, Equation 5.8 to calculate the penalty that should be multiplied with the edit distance of the tracks.

Table 8 compares the results from the three simulations done with tracks. Just looking

Distance metric	**EER**
No penalty	40.3%
Variance distance of variance as penalty	40.5%
Penalty based on the difference in variance	40.1%

Table 8: EER results from simulations using the full dataset (All MA filtered)

at this table would show that the final simulation with a penalty function produces the best result. The differences in EER is however too small to come to this conclusion. With the small differences shown here anything is possible.

The high EER may be an artifact of the track extraction problem. This problem will have to be solved before the true performance of these different distance metrics would emerge. The EER could be expected to drop if the track detection problem could be solved. The use of information of the opposite moving direction could prove to be a unique factor, this could show itself if the track detection problem could be solved. This tells us that using automatic extraction of tracks should be either greatly improved or be done manually. Doing the extraction of tracks manually is a time consuming task and could not be performed for this thesis. See Section 8.1 for possible solutions to this problem.

6.3 Comparison of the two techniques

In this section we will compare the two techniques presented in Section 6.1 and Section 6.2 as best as possible. Table 9 illustrates the EER achieved using the different distance metrics. It is obvious that the distance metrics using the whole dataset without any feature extraction gave the best performance. This comparison is not however entirely valid because of the feature extraction problem involved when working with tracks. The true performance of the distance metrics involving the tracks and variance will first be seen if the track extraction problem is solved. As mentioned this can be done by improving the algorithms that do the automatic extraction, or just performed manually. The manual extraction is a large task when dealing with many users. The improvement of the track extraction process has not been performed in this thesis, but is suggested as future work in Chapter 8.

The distance metrics involving tracks did however unearth the possible uniqueness of

Distance metric	**EER**
Edit distance horizontal (whole dataset, MA filter)	26.8%
Edit distance vertical (whole dataset, MA filter)	27.0%
Edit distance both (whole dataset, MA filter)	27.5%
Edit distance horizontal (whole dataset, WMA filter)	28.9%
Edit distance tracks	40.3%
Edit distance tracks - Variance distance of variance	40.5%
Edit distance tracks - Penalty based on variance	40.1%

Table 9: EER results from different distance metrics

the variance for each user. This measures how stable the user is in one direction, when he or she is actually moving the mouse in the other direction. The distance metric that used the difference of the variance and applied a penalty accordingly performed the best of all the track metrics. This shows the potential of uniqueness.

As discussed the overall comparison of the two techniques (no feature extraction and track extraction comparison) is not entirely comparable due to the problems of the track extraction. This make is hard to directly compare the two techniques performances. The two techniques show potential in their own way. The track metric shows that variance is a potential feature that is unique. When not extracting tracks it has been shown that with an EER of 26.8%, mouse authentication using a labyrinth is worth looking into deeper in the future.

7 Conclusion

The focus of this thesis was to look at using the mouse as an authentication tool. The research questions was split into two main parts which will be concluded in their individual sections.

7.1 Can we use mouse authentication?

To answer this question good data was needed to perform simulations. A program was written that presented the user with two tasks; labyrinth and connect-the-dots. The main idea was to perform simulations on both of these tests, however it turned out analyzing the data collected in the connect-the-dots program would become a too large task. The data produced by this task illustrates the most normal mouse usage, but this is largely random (see Figure 13 in Section 3.1.2). This makes it hard to extract features from this data, because the question of what data to extract requires much research to answer. Therefore the analysis of this task was left out. It also proved that the analysis of the labyrinth was very time consuming, even though this was very structured data (see Figure 11 in Section 3.1.1). The data from the connect-the-dots task was however not discarded, making it possible for future analysis of this task.

The data collected by the program was stored in its most raw form to make it possible to alter it in any way possible afterwards. The data was filtered to remove noise and velocity data was calculated. Two different main techniques were applied, one where unaltered velocity data was analyzed and one were tracks were extracted. The main problem was shown to be the track extraction which will need more work to show the full potential of the distance metrics utilizing the tracks.

7.2 Is mouse authentication a good stand alone technique?

At this point in time the simple answer is no. Given that the best EER from the simulations done in this thesis equals 26.8%, this is not good enough for a stand alone system. It does however show potential, indicating that there may be unique features within this data. The main problem in this thesis, as it turned out, was the track seperation. This resulted in that the proposed custom distance (track distance) metric could not show its full potential. The best EER from the simulations performed with this metric was 40.1%. It did however show that variance in the speed in the other direction might be a feature that would be usefull in the authentication process.

The main conclusion to this research question is that mouse biometrics will need more research into feature extraction before it can possibly be used as a stand alone system.

8 Future work

This chapter describes problems encountered during the work on the thesis. Possible solutions to these problems will be suggested as well as suggestions to new experiments that can avoid these problems. It will also suggest additions to algorithms and feature extractions that may have an impact on the measured distance. Social exclusion studies affiliated with mouse biometrics will also be suggested.

8.1 Problems encountered

The main problem encountered during simulations was the track separation problem. The automatic separations does fail on occasions where users do not alter their speed in a significant way. This results in the distance metrics and analysis of the data that relies on the tracks giving faulty performance rates. Improving the ”Track separation algorithm” described in Section 4.2 would be an important step toward seeing the potential of the successive distance metrics proposed in this thesis. Some possible solutions to this problem would be direct improvements to the algorithm, which works with speed data. One can also look at the positioning data and directly separate the tracks when the user passes a certain line. Figure 24 illustrates this approach. This approach would also make it easier to analyze the transition phases, detonated *e* in the figure. The red lines indicates possible transition lines, the boxes within the red lines indicates transition phases which could be further analyzed. The problem with this approach is that one would have

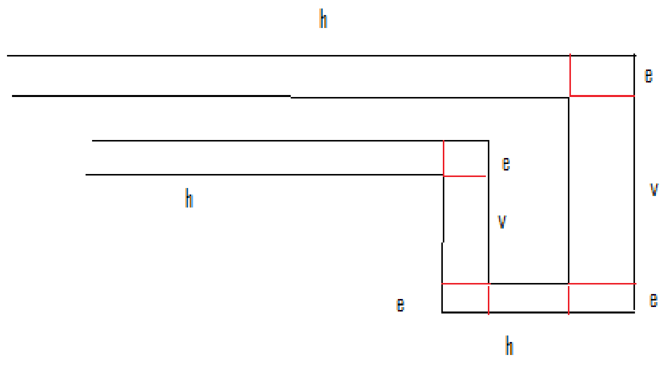


Figure 24: Transition lines

to know exactly where the lines of transition would fall. This could be done by letting the data collection program it self mark these places, or making a labyrinth of fixed size. In addition to the transition lines suggested in Figure 24 the following possibilities could also be considered:

- using diagonal lines in the corners where transitions happens.
- ignore the data in the *e* boxes completely.

- consider data in 3 boxes as separate, i.e. 3 types: h, v and e. The data in *e* might have a lot of information.
- analyze the *e* boxes further; there is exactly 1 transition from h to v (or from v to h) inside each *e* box.

The labyrinth in this thesis alters its size accordingly to the screen resolution of the user or the window size of the program, hence a fixed labyrinth approach is not possible for the data collection techniques presented in this thesis. Making a fixed labyrinth would in it self introduce problems due to the many different operating systems and screen resolutions. The other approach toward a solution to the track separation problem is to manually correct the mistakes made by the automatic process. This is however a time consuming process due to the amount of data.

There is also the problem of different behavior of the mouse logger application on MAC OS X, as described in Section 3.1. Due to this problem only happening on MAC OSX Leopard, the conclusion could be drawn that there is a problem with java on that platform. To make sure that the program will be compatible with different operating systems, the application could be written in the respective platforms API's (Application Programming Interface) in for example a language such as C++. Certain libraries as QT [26] could be used, which use the native API's. This problem was however not a serious one in this thesis due to the low amount of participants using MAC OSX Leopard during the experiment phase.

The analysis of the connect-the-dots task presented in Section 3.1.2 is something that was not further looked at in this thesis due to its complexity. The data looks random, and extracting features from this data is a hard task. On the other hand this data is what would have the most likely potential to represent a users normal mouse behavior, since there are no restrictions on how to move the mouse except for clicking the dots. Compared to the labyrinth which restricts the user all the way by instructing them to stay within two lines.

8.2 New work ideas

This section will suggest further work, both based on the ideas from this thesis and new ideas that appeared during the work on the thesis.

The use of mouse authentication in a web application could be a viable goal. Either as an extension to an existing password system or as a stand alone system. Tasks that could be implemented would include virtual keyboards, pin codes or point and click like games.

The Levenshtein distance algorithm used in this thesis (as described in Section 2.3) uses the operation cost of 1. These costs could be altered to try to find a variation of this algorithm. Below are some suggestions for alternations.

- Using different fixed costs for *deletion and insertion*, and *substitution*.
- Using fixed costs for *deletion and insertion*, and a cost dependent of the values involved in the data for *substitution*.
- Using costs dependent of the values involved in the data for all three operations (deletion, insertion and substitution).

The investigation of the impact of different hardware is not undertaken in this thesis.

This could prove to be an important factor and should be investigated. For example a user could switch mouse and not be able to access a system if this hardware change impacted the data. Other different types of mouse devices could also be investigated, such as mouse pads very common on a laptop. This could also include the study of learning curve problems associated with mouse authentication. Does the performance of the tasks alter over time? Does one need to adjust the templates in accordance? An interesting possible feature discovered at the end of the work on this thesis is related to learning curves. Users will learn how to move the mouse through the labyrinth at will get more fluent in this. The learning curve might be used as a feature to recognize users. Some figures are presented on this topic in Appendix E, however no analysis has been undertaken.

A social exclusion study could also be undertaken in regards with the mouse authentication. Could diseases (i.e. muscle control problems, nerve damage) exclude people from being able to use the system. What about the elderly, do they have other needs when using a pointing device. Such problems could be studied.

Bibliography

[1] J. Wayman, A. Jain, D. Maltoni, and D. Maio. *Biometric Systems: Technology, Design and Performance Evaluation*. Springer-Verlag, 2005.

[2] A.K. Jain, A. Ross, and S. Prabhakar. An introduction to biometric recognition. *Circuits and Systems for Video Technology, IEEE Transactions on*, 14(1):4–20, Jan. 2004.

[3] Jeremy Wickins. The ethics of biometrics: the risk of social exclusion from the widespread use of electronic identification. *Science and Engineering Ethics*, Volume 13, Number 1 / March, 2007:45–54, 2007.

[4] Friederike D. Griess. On-line signature verification (master's project report). Technical Report MSU-CSE-00-15, Department of Computer Science, Michigan State University, East Lansing, Michigan, July 2000.

[5] Maja Pusara and Carla E. Brodley. User re-authentication via mouse movements. In *VizSEC/DMSEC '04: Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, pages 1–8, New York, NY, USA, 2004. ACM.

[6] Ahmed Awad E. Ahmed and Issa Traore. A new biometric technology based on mouse dynamics. *IEEE Transactions on Dependable and Secure Computing*, 4(3):165–179, 2007.

[7] D.A. Schulz. Mouse curve biometrics. *Biometric Consortium Conference, 2006 Biometrics Symposium: Special Session on Research at the*, pages 1–6, Sept. 19 2006-Aug. 21 2006.

[8] H. Gamboa, A. L. N. Fred, and A. K. Jain. Webbiometrics: User verification via web interaction. *Biometrics Symposium, 2007*, pages 1–6, 11-13 Sept. 2007.

[9] I. Ahmed, A.A.E.; Traore. Anomaly intrusion detection based on biometrics. *Information Assurance Workshop, 2005. IAW '05. Proceedings from the Sixth Annual IEEE SMC*, pages 452–453, 15-17 June 2005.

[10] H. Gamboa and V. Ferreira. Widam - web interaction display and monitoring. In *5th International Conference on Enterprise Information Systems, ICEIS'2003*, Anger, France, 2003. INSTICC PRESS.

[11] Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of speech recognition*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

[12] L. O'Gorman. Comparing passwords, tokens, and biometrics for user authentication. *Proceedings of the IEEE*, 91(12):2019–2020, Dec 2003.

[13] A. Martin, M. Przybocki, G. Doddington, and D. Reynolds. The nist speaker recognition evaluation - overview, 2000.

[14] Francesco Bergadano, Daniele Gunetti, and Claudia Picardi. User authentication through keystroke dynamics. *ACM Trans. Inf. Syst. Secur.*, 5(4):367–397, 2002.

[15] D. Maio, D. Maltoni, R. Cappelli, J.L. Wayman, and A.K. Jain. Fvc2000: fingerprint verification competition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(3):402–412, Mar 2002.

[16] Townend D. Overriding data subjects' rights in the public interest. *The data protection directive and medical research across Europe*, 2004.

[17] E.S. Ristad and P.N. Yianilos. Learning string-edit distance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(5):522–532, May 1998.

[18] Gonzalo Navarro. A guided tour to approximate string matching. *ACM Comput. Surv.*, 33(1):31–88, 2001.

[19] Christopher Johnsrud Fullu, Kennet Fladby, and Lars Olav Gigstad. Authentication by mouse movement. *unpublished*, 2007.

[20] http://java.sun.com/, 2008.

[21] http://www.microsoft.com/windows/, 2008.

[22] http://kernel.org/, 2008.

[23] http://www.sun.com/software/solaris/index.jsp, 2008.

[24] http://www.apple.com/macosx/, 2008.

[25] http://www.apple.com/support/tiger/, 2008.

[26] http://trolltech.com/products/qt/, 2008.

A Userfile format (*.mu files)

The user file, which is supplied to every participant is written to file using java.io's writeObject function. This function writes java objects directly to file. The class written for the user file is as follows:

```java
// Handles user objects
import java.util.*;
import java.io.*;
import java.awt.Dimension;
import java.io.Serializable;

public class User extends Object implements Serializable
{
        private String username;
        private int session;
        private Dimension screenRes;

        public User( String username, int session )
        {
                this.username = username;
                this.session = session;
                screenRes = new Dimension( 0, 0 );
        }

        public String getUsername()
        {
                return username;
        }

        public int getSession()
        {
                return session;
        }

        // User has performed another session
        public void updateSession()
        {
                session = session + 1;
        }

        // User have changed full screenresolusion
        public void updateScreenRes( Dimension res )
        {
                screenRes = res;
        }

        public Dimension getScreenRes()
        {
                return screenRes;
        }
}
```

This class stores the users user name, session number and screen resolution.

B Mouse data file format (*.md files)

The *.md files stores the mouse data collected from the program when a user have performed the labyrinth and dot tasks. These files are labeled after what user, which session and which task was performed, for example *User1_Session3_labyrinth.md*.

This file is also written using java.io writeObject function, which means it also contains java objects. The format of the file is as follows:

- The file starts with the userfile content as a header (described in Appendix A)
- If this files contains data from the dots task, the position of each dot is stored after the header in the form of an array of *java.awt.Point* objects
- The file ends with a *java.util.Vector* object filled with MouseData objects (described below). This is where the actual data is stored

The format of the MouseData class is as follows:

```java
import java.io.Serializable;

/**
 * A class to keep all the collected data for a sample.
 **/
public class MouseData extends Object implements Serializable
{
        public double x;
        public double y;
        public int time_ms;

        public MouseData( double x, double y, int time_ms )
        {
                this.x = x;
                this.y = y;
                this.time_ms = time_ms;
        }

        public String toString()
        {
                return "[X:␣" + String.valueOf(x) + ",␣Y:␣" + String.valueOf(y)
                        + ",␣Time:␣" + String.valueOf(time_ms) +"]";
        }
}
```

C Mouse logger participant readme (English)

Christopher Johnsrud Fullu, 03.03.2008

MOUSE LOGGER

First and foremost, thank you for helping me with my master thesis by participating. Your help is greatly appreciated.

The following will describe what is required of your computer and a brief guideline on how to perform the tests using your mouse. What type of data that is collected will also be answered in this document.

The purpose of this master thesis, and thus the purpose of collecting this data, is to investigate if authentication of individual persons is possible by looking at mouse movements. For these two different tasks have been designed and the purpose is to see which one performs the best; labyrinth or connect-the-dots.

REQUIREMENTS

- Windows, Linux, Unix, Mac, Solaris operating system, or any operating system that supports java will do.
- Java runtime environment (JRE SE, version 1.5 or above)[1]
- A mouse (laser, optical – please do not use a ball mouse or a touch/mouse pad like on a laptop)

USAGE GUIDELINE

There are not many restrictions on how to perform the tests, that is because we want to capture your normal mouse usage. The goal is to perform the tests like you normally use your mouse. Please do not use a ball mouse or a touch/mouse pad (like on a laptop), but rather a laser or optical mouse if that is possible for you.

HOW TO RUN THE PROGRAM

There are several ways; the program is compressed into a jar file to make it easy for the user. There are several ways to execute this jar file:

- On Windows you can simply double click on the program if you have the jre installed and registered correctly.

- In general the jar files can be executed in a command prompt with the following command line:

[1] The jre se can be downloaded from http://java.sun.com/javase/downloads/index.jsp for Windows, Linux and Solaris operating systems, for Mac the jre can be downloaded from http://www.apple.com/support/downloads/javaformacosx104release6.html

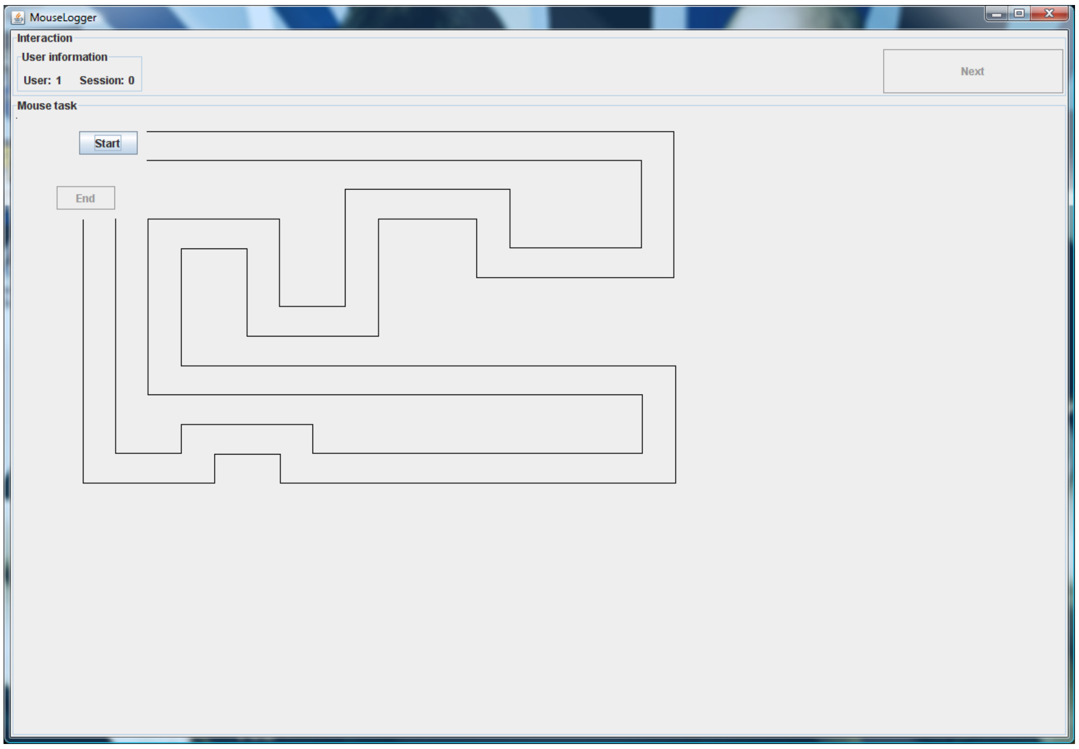
Christopher Johnsrud Fullu, 03.03.2008

java –jar MouseLogger.jar

The program is supplied to you with a user file (mouseuser.mu) and the main program (MouseLogger.jar). When you receive these files, please put them together in a folder, and the program is ready to go.

THE LABYRINTH

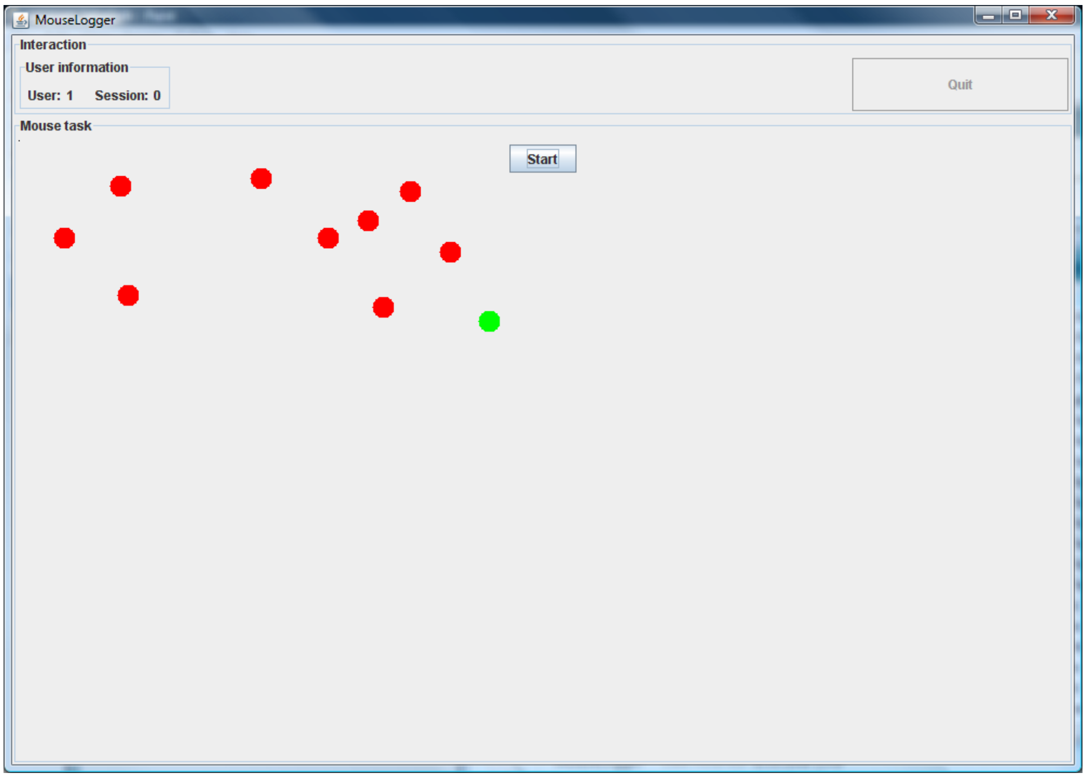
Below is a screenshot of the labyrinth. The idea with this test is to follow the labyrinth as well as possible with your mouse pointer. Click the start button and follow the labyrinth, the dotted line following the mouse pointer illustrates that the tests has begun, when you are done click the End button and continue to the dot task by clicking the next button.



THE DOTS TEST

Below is a screenshot of the dots test. You can see that there are several dots on the screen, most of them are red, but one is green. When you start the test by clicking the Start button your goal is to click on every green dot that appears. The dots are placed random each time the test begins and the green dot is randomly selected when you have clicked the previous green dot. Dots turn blue when they have been clicked on. When all dots have been clicked, the task is complete, please click the Quit button to close the program.

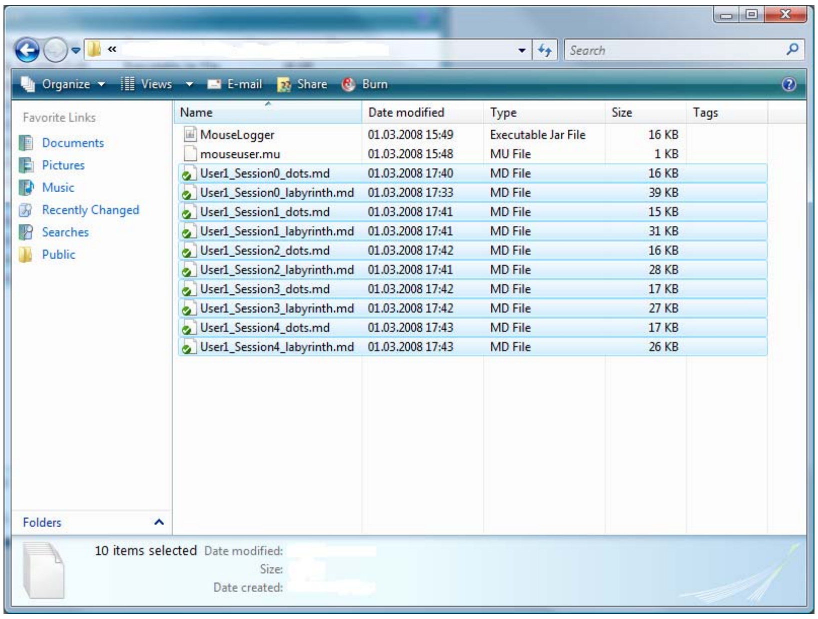
Christopher Johnsrud Fullu, 03.03.2008



HOW MANY TIMES AND WHAT TO SEND WHERE

We hope you have the time to complete the test 6 times over a period of approximately 2-3 weeks. For each time, please complete 5 sessions (repeating each test 5 times) after each other, but leave some days between each time. Making for a total of 30 runs through the tests, the program will display session number as 29 when you are done.

When you are done with a session some *.md files will appear in the directory from where the program resides. This is the data collected. Please send these files via mail to: christopher.fullu@hig.no or kjfull@online.no and set the email subject to "Mouse authentication", thank you.



Name	Date modified	Type	Size	Tags
MouseLogger	01.03.2008 15:49	Executable Jar File	16 KB	
mouseuser.mu	01.03.2008 15:48	MU File	1 KB	
User1_Session0_dots.md	01.03.2008 17:40	MD File	16 KB	
User1_Session0_labyrinth.md	01.03.2008 17:33	MD File	39 KB	
User1_Session1_dots.md	01.03.2008 17:41	MD File	15 KB	
User1_Session1_labyrinth.md	01.03.2008 17:41	MD File	31 KB	
User1_Session2_dots.md	01.03.2008 17:42	MD File	16 KB	
User1_Session2_labyrinth.md	01.03.2008 17:41	MD File	28 KB	
User1_Session3_dots.md	01.03.2008 17:42	MD File	17 KB	
User1_Session3_labyrinth.md	01.03.2008 17:42	MD File	27 KB	
User1_Session4_dots.md	01.03.2008 17:43	MD File	17 KB	
User1_Session4_labyrinth.md	01.03.2008 17:43	MD File	26 KB	

10 items selected Date modified:
Size:
Date created:

Christopher Johnsrud Fullu, 03.03.2008

WHAT DATA IS COLLECTED

First off, nothing is collected that can link the data back to you, this is impossible.

What does the user file (mouseuser.mu) store?

- Your user number (username), for example 1 (this is also shown in the program)
- Your session number (what session are you on at the moment). This is incremented with one each time you complete the labyrinth and dots test, by clicking the Quit button.
- Your screen resolution. This might be important in distance calculations.

What data is collected? (What is stored in the md files)

- The data stored in the user file (mouseuser.mu)
- The position of the mouse, this information is stored 100 times per second. In other words, [X, Y] and time data.
- For the dots test, the position of the dots is also stored, as the dots are randomly placed each time.

Thank you, if there are any questions, please contact me via email: christopher.fullu@hig.no or kjfull@online.no

D Mouse logger participant readme (Norwegian)

Christopher Johnsrud Fullu, 03.03.2008

MOUSE LOGGER

Først og fremst, tusen takk for din deltagelse i min master oppgave, din hjelp er sterkt verdsatt.

Dette dokumentet vil beskrive hva som er påkrevet av din pc, samt ne kort forklaring av hvordan testene skal gjennomføres og hvordan data skal sendes tilbake til oss. Hva som blir lagret vil også bli beskrevet.

Målet med denne master oppgaven, og dermed grunnen til at disse data skal samles, er å forske på om autentisering av individuelle personer er mulig ved å se på bevegelsene en person gjør med en data mus. For å undersøke dette har to tester blitt designet, for så å se hvem av dem som gir best mulig resultat; labyrint og connect-the-dots.

SYSTEM KRAV

- Windows, Linux, Unix, Mac, Solaris operativ system, eller et hvilket som helst system som støtter java.
- Java runtime environment (JRE SE, versjon 1.5 eller senere)[1]
- En datamus (laser, optisk – vennligst ikke bruk en ball mus eller en touch/mouse pad som for eksempel på en bærbar pc).

BRUKER GUIDE

Det er ikke mange restreksjoner på hvordan testene skal gjennomføres, det er fordi vi vil samle hvordan datamusen brukes til vanlig. Målet er å gjøre testene sånn som du vanligvis bruker din datamus. Vennligst ikke bruk en ball mus, eller en touch/mouse pad som for eksempel på en bærbar pc.

HVORDAN KJØRE PROGRAMMET

Det finnes forskjellige måter å kjøre programmet på; programmet er komprimert i en jar fil for å gjøre det enkelt for brukeren.

- På Windows kan jar filer kjøres direkte ved og dobbelt klikke på dem, så sant jre er installert riktig.
- Generelt kan jar filer kjøres i en kommando linje med kommandoen:

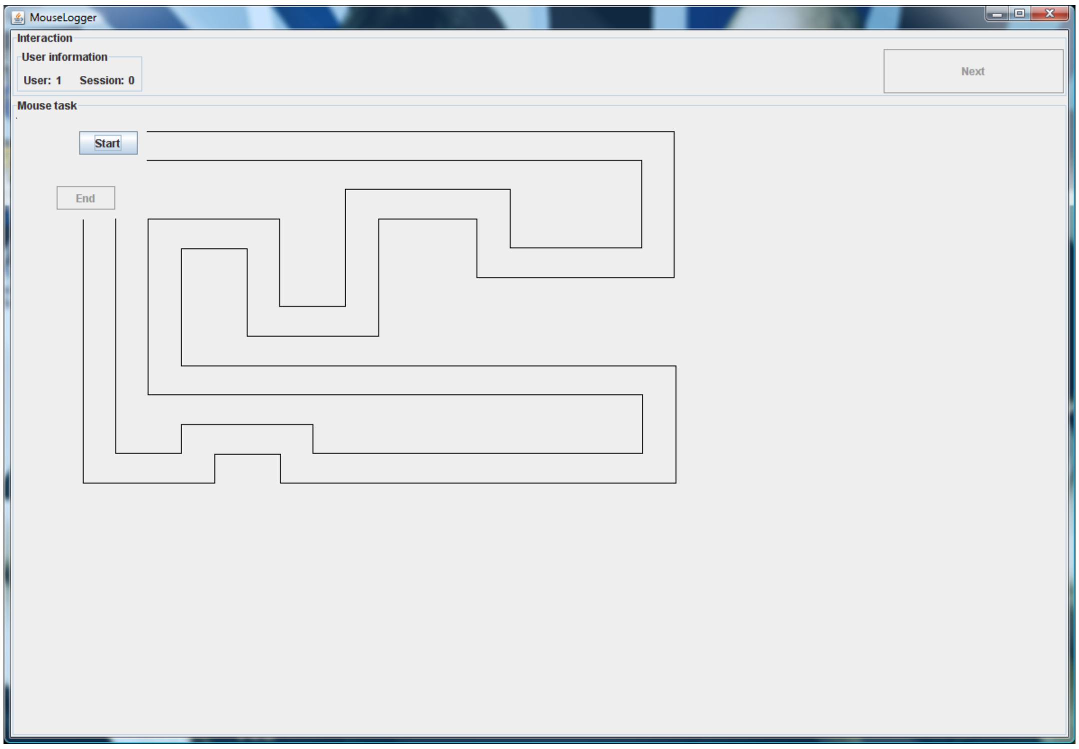
  java –jar MouseLogger.jar

Programmet har blitt sendt til deg med en brukerfil (mouseuser.mu) og selve programmet (MouseLogger.jar). Når du har mottatt disse filene, vennligst legg dem i samme mappe, og programmet skal være klart til å kjøre.

[1] Jre kan lasts ned fra http://java.sun.com/javase/downloads/index.jsp for Windows, Linux og Solaris, for Mac kan den lasts ned fra http://www.apple.com/support/downloads/javaformacosx104release6.html

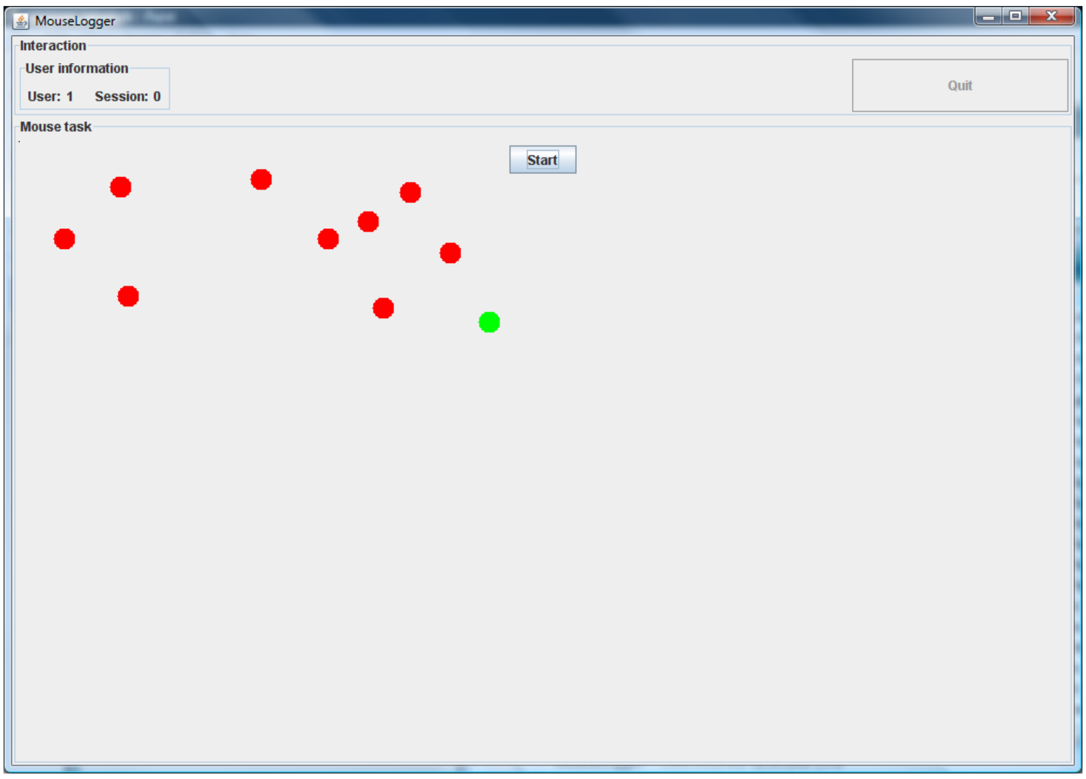
LABYRINTEN

Nedenfor ser man et bilde av labyrinten. Ideen med denne testen er å følge den så best som mulig med muse pekeren. Klikk på Start knappen for å starte, og når man er ferdig trykker man på End knappen. For å gå videre til neste test trykk på Next.



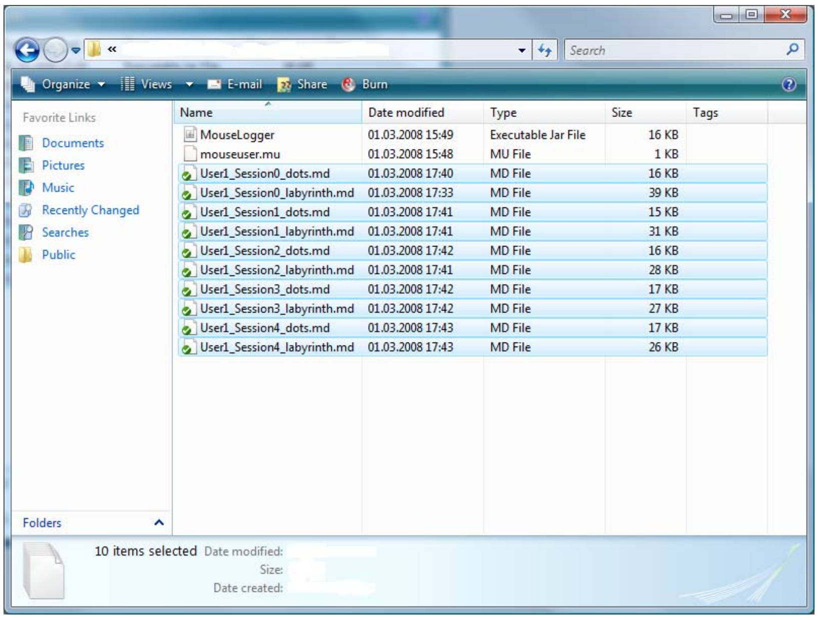
DOTS TESTEN

Under finner man ett bilde av dots testen. De røde dottene er dotter som ikke har blitt klikket på enda, den grønne er den man skal klikke på. Det kommer en ny grønn med en gang man har klikket på en. Dottene blir merket blå når de har blitt klikket på. Når alle dottene har blitt klikket på er man ferdig med testen, vennligst klikk på Quit knappen for å avslutte.



HVOR MANGE GANGER OG HVA SENDER JEG HVOR

Vi håper du har tid til å gjennomføre testene 6 ganger over en periode på 2-3 uker. For hver gang, vennligst gjør 5 sessions (starte programmet om igjen 5 ganger og gjøre testene hver gang), men la det gå noen dager mellom hver gang. Dette vil totalt bety at man har kjørt de to testene til sammen 30 ganger. Programmet vil vise session som 29 når man er ferdig. Når man er ferdig med en session, noen *.md filer vil finnes i den mappen som programmet ligger. Dette er dataene som ble lagret fra testene. Vennligst send disse filene via email til: christopher.fullu@hig.no or kjfull@online.no og vennligst sett emne til “Mouse authentication”, takk.



Name	Date modified	Type	Size	Tags
MouseLogger	01.03.2008 15:49	Executable Jar File	16 KB	
mouseuser.mu	01.03.2008 15:48	MU File	1 KB	
User1_Session0_dots.md	01.03.2008 17:40	MD File	16 KB	
User1_Session0_labyrinth.md	01.03.2008 17:33	MD File	39 KB	
User1_Session1_dots.md	01.03.2008 17:41	MD File	15 KB	
User1_Session1_labyrinth.md	01.03.2008 17:41	MD File	31 KB	
User1_Session2_dots.md	01.03.2008 17:42	MD File	16 KB	
User1_Session2_labyrinth.md	01.03.2008 17:41	MD File	28 KB	
User1_Session3_dots.md	01.03.2008 17:42	MD File	17 KB	
User1_Session3_labyrinth.md	01.03.2008 17:42	MD File	27 KB	
User1_Session4_dots.md	01.03.2008 17:43	MD File	17 KB	
User1_Session4_labyrinth.md	01.03.2008 17:43	MD File	26 KB	

10 items selected Date modified:
Size:
Date created:

Christopher Johnsrud Fullu, 03.03.2008

HVA BLIR LAGRET

Først og fremst, det blir ikke lagret noe som kan identifisere deg som person. Det er umulig å finne tilbake til deg i ettertid.

Hva lagrer brukerfila (mouseuser.mu)?

- Ditt bruker navn (nummer), for eksempel 1 (dette vises også i programmet)
- Ditt session nummer (hvilken session man er på i øyeblikket). Denne plusses på med 1 hver gang man har fullført en session og trykket på Quit knappen.
- Din skjermoppløsning. Dette kan være viktig med tanke på distanse beregninger.

Hva slags data blir lagret (hva er lagret i *md filer)?

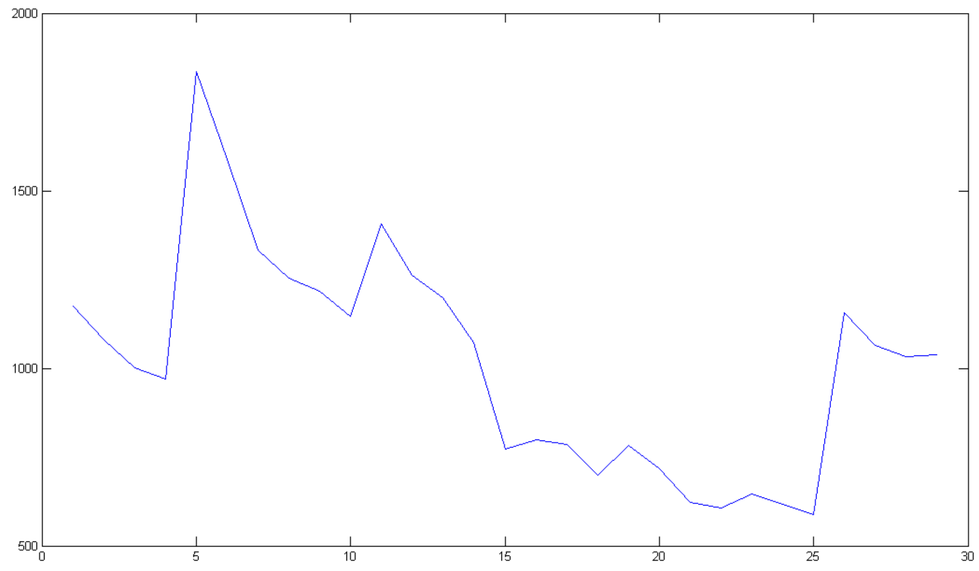
- Posisjonen til musepekeren, denne informasjonen lagres 100 ganger i sekundet. Med andre ord, [X, Y] og tids data.
- For dot testen, lagres også posisjonen til dottene, siden disse blir plassert tilfeldig hver gang.

Takk, hvis det skulle være noen spørsmål, vennligst kontakt meg via
email: christopher.fullu@hig.no or kjfull@online.no

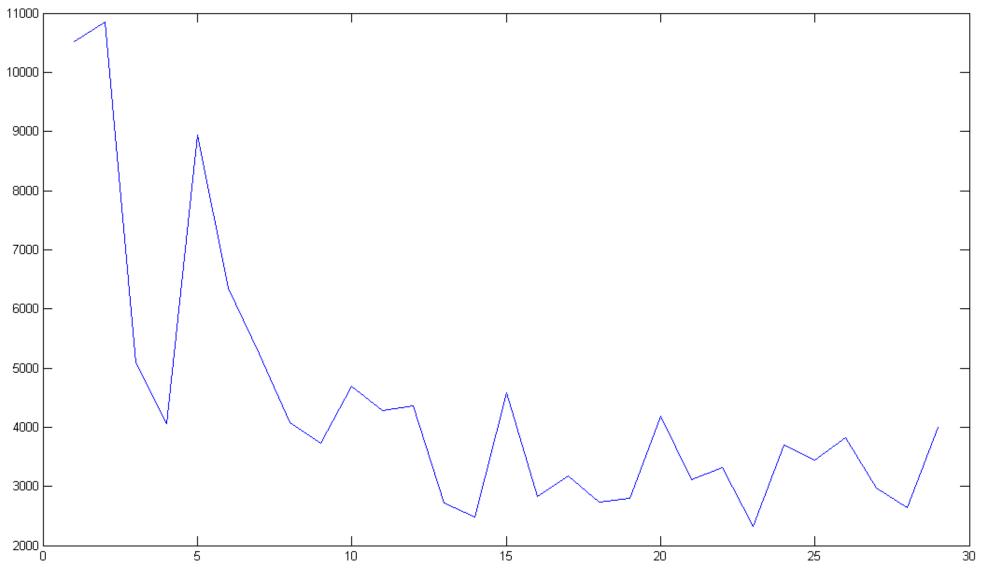
E Learning curves based on labyrinth completion time

This section will present some graphs that indicate some of the learning curve effects of the labyrinth task. These graphs are produced by looking at the number of samples per session for a participant. The idea is to look at whether a participant performs the task faster when as the session increases. This graph could indicate that a participant learns the task after having performed it some times, thus present an opportunity to illustrate this effect graphically. This could also be a indication on how stable over time a user is with the mouse. High variations of sample length might indicate that a user varies a lot. This parameter might be a feature valid for a distance metric.

Some users seem to indicate a learning curve, but most of the users seem to vary randomly. This could indicate that learning curve problem is individual, but that most users do not have this artifact. An interesting observation is that this effect may be used in a distance metric. An idea might be to use this feature in a distance metric, looking at the variance in sample time. A user might have a personal sample length variance. This is a feature that could be further researched.

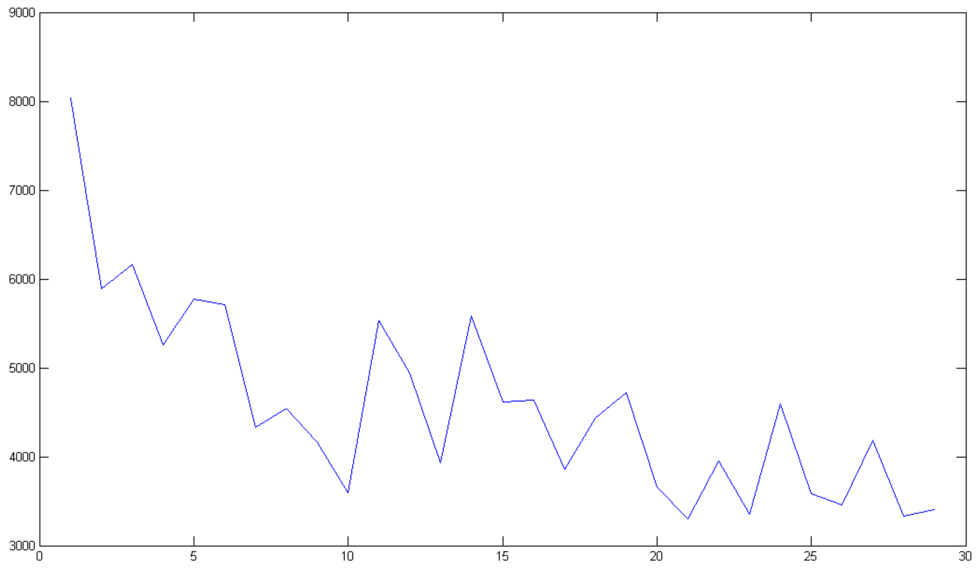


(a) User A

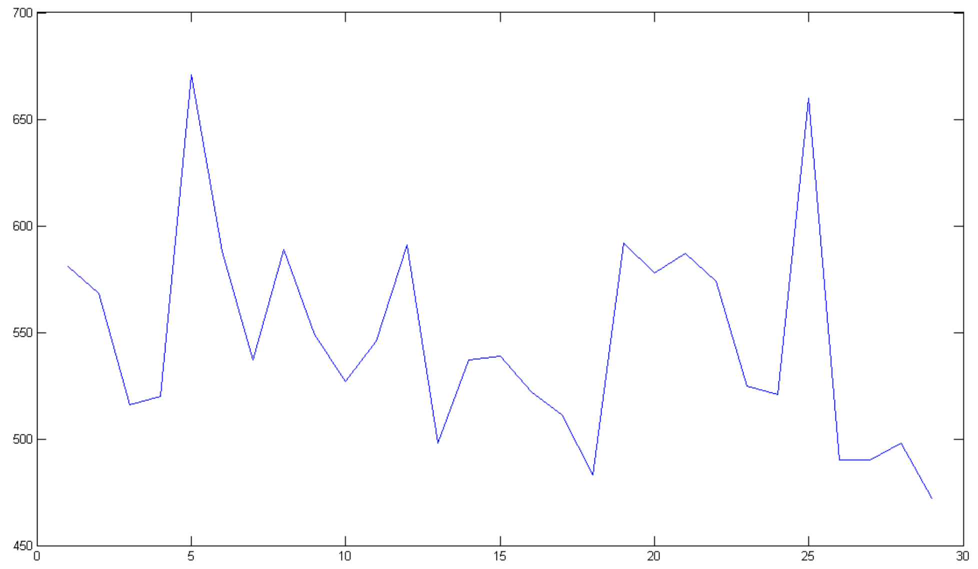


(b) User B

Figure 25: Number of samples over time for different participants - 1

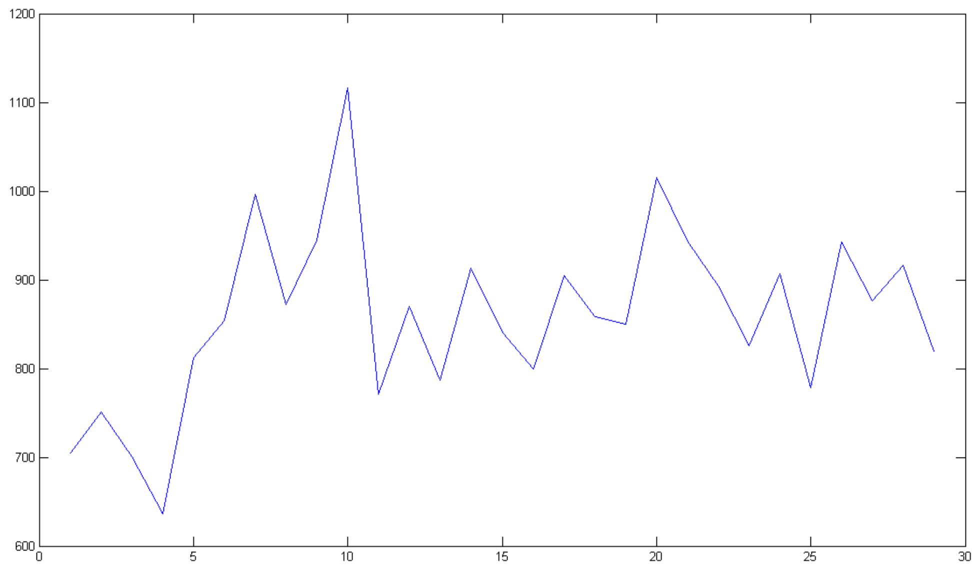


(a) User C

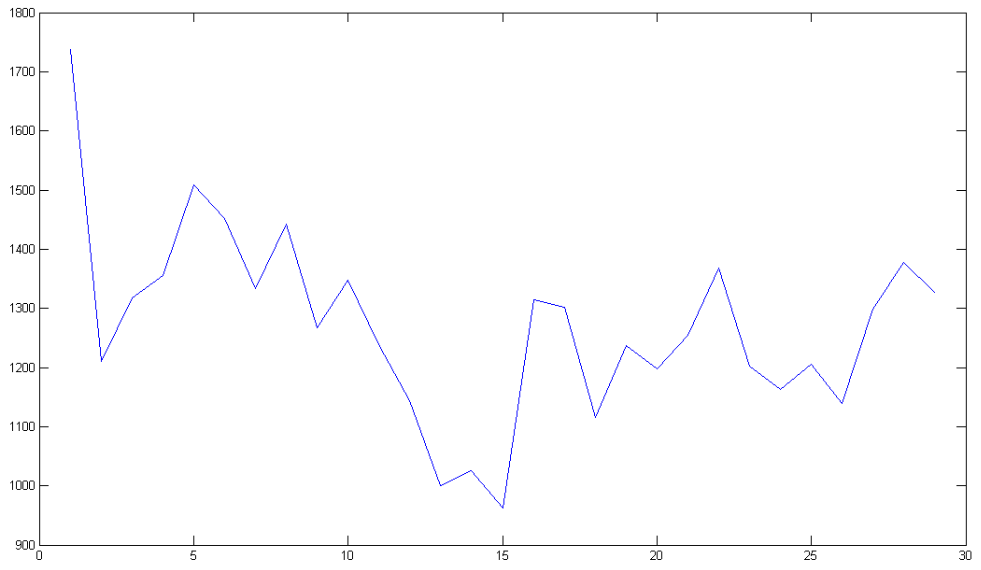


(b) User D

Figure 26: Number of samples over time for different participants - 2

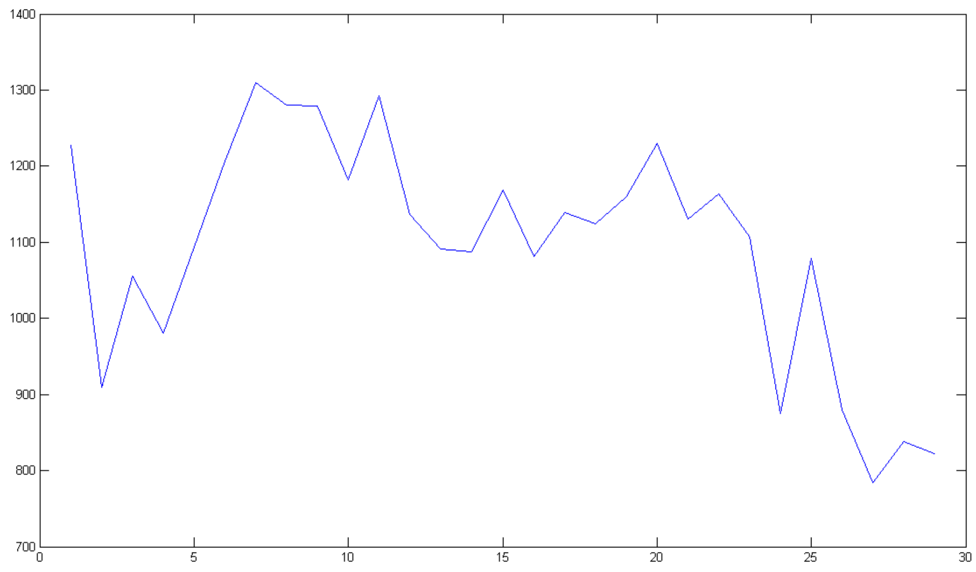


(a) User E

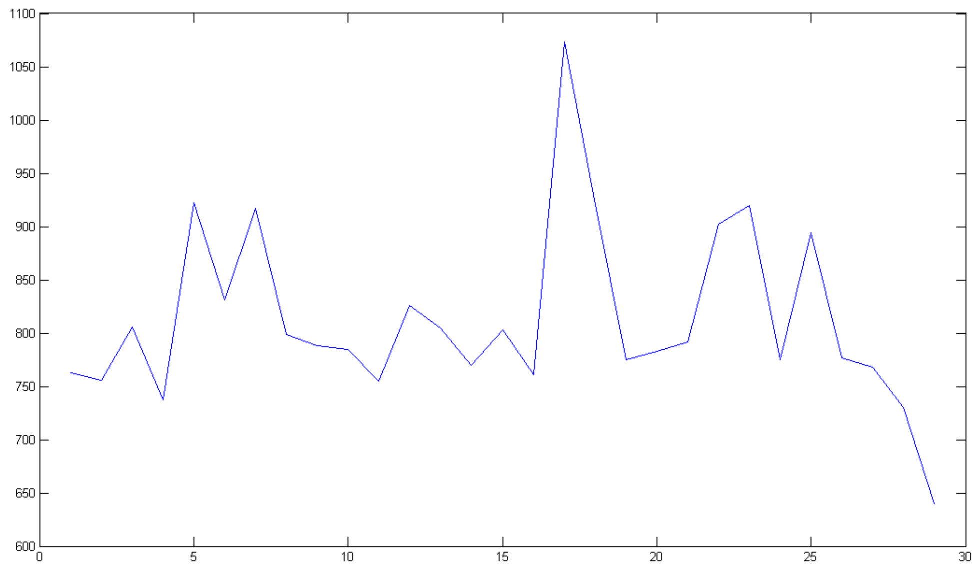


(b) User F

Figure 27: Number of samples over time for different participants - 3

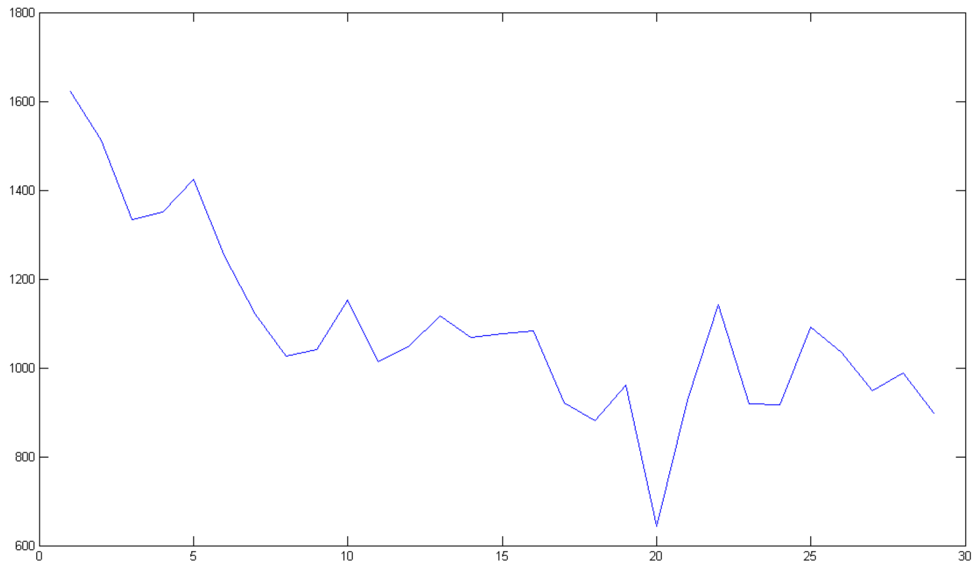


(a) User G

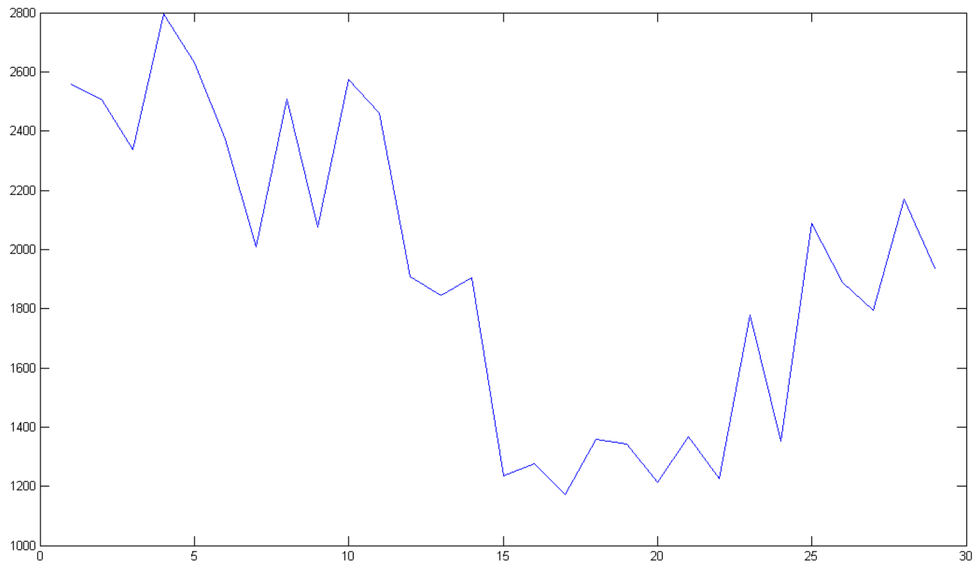


(b) User H

Figure 28: Number of samples over time for different participants - 4

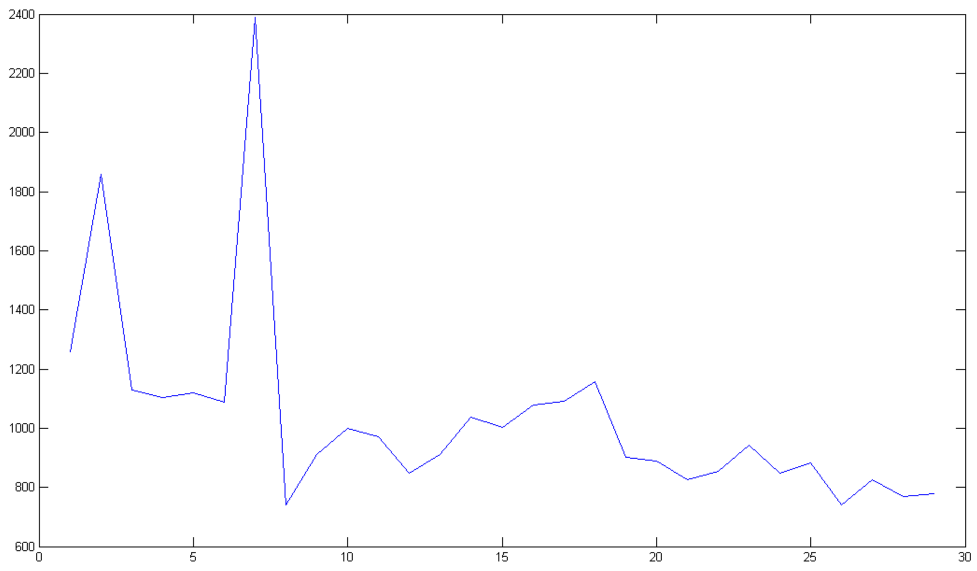


(a) User I

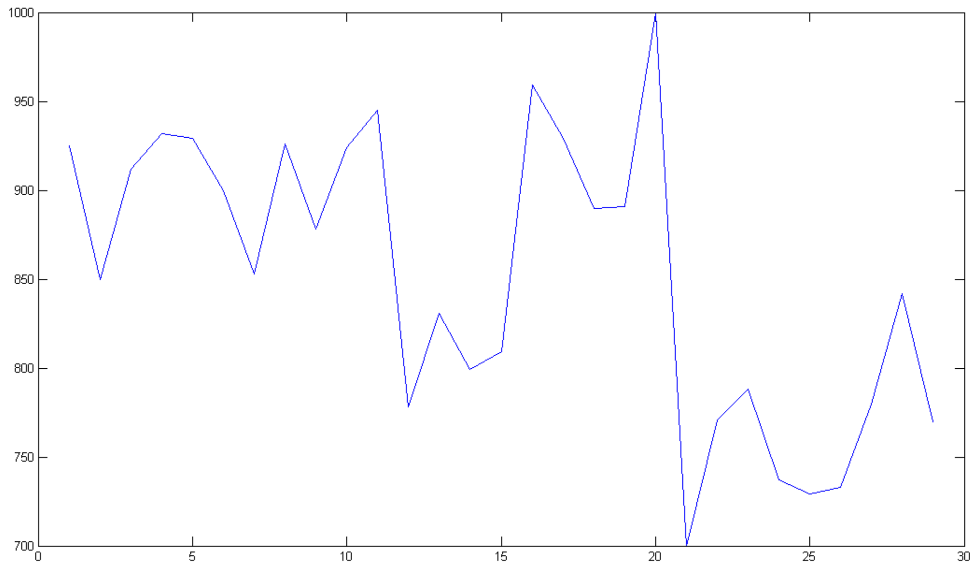


(b) User J

Figure 29: Number of samples over time for different participants - 5

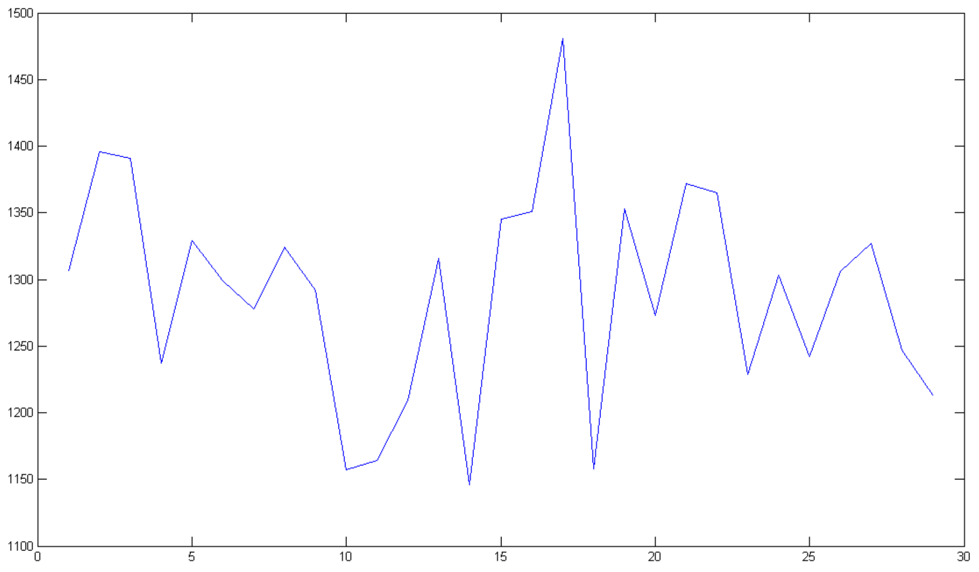


(a) User K

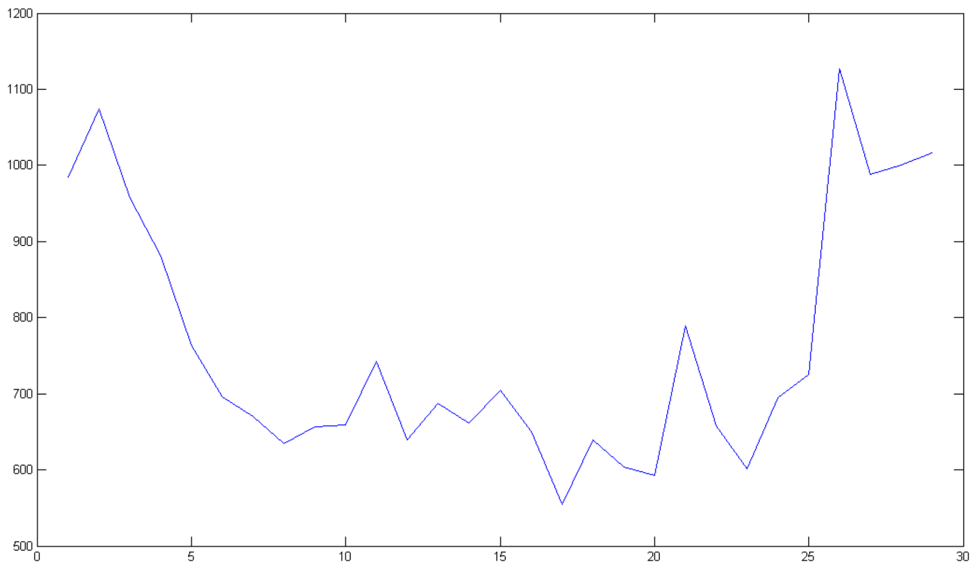


(b) User L

Figure 30: Number of samples over time for different participants - 6



(a) User M



(b) User N

Figure 31: Number of samples over time for different participants - 7

F Levenshtein algorithm

This appendix describes the Levenshtein algorithm (or edit distance) with code. The pseudo code in this appendix is taken from http://en.wikipedia.org/wiki/Levenshtein_distance (valid as of 27.06.2008)

```
int LevenshteinDistance(char s[1..m], char t[1..n])
  // d is a table with m+1 rows and n+1 columns
  declare int d[0..m, 0..n]

  for i from 0 to m
    d[i, 0] := i
  for j from 0 to n
    d[0, j] := j

    for i from 1 to m
      for j from 1 to n
      {
        if s[i] = t[j] then cost := 0
          else cost := 1
          d[i, j] := minimum(
            d[i-1, j] + 1,     // deletion
            d[i, j-1] + 1,     // insertion
            d[i-1, j-1] + cost   // substitution
                          )
      }

  return d[m, n]
```