BEN GURION UNIVERSITY OF THE NEGEV
FACULTY OF ENGINEERING SCIENCES
DEPARTMENT OF INDUSTRIAL ENGINEERING AND MANAGEMENT

Biometric Identification by Hand Motion Signature

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE M.Sc DEGREE

By: Omri Mendels

SEPTEMBER 2011

BEN GURION UNIVERSITY OF THE NEGEV
FACULTY OF ENGINEERING SCIENCES
DEPARTMENT OF INDUSTRIAL ENGINEERING AND MANAGEMENT

Biometric Identification by Hand Motion Signature

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE M.Sc DEGREE

By: Omri Mendels

Supervised By: Dr. Sigal Berman
Prof. Helman Stern

Author:…….. Date:……

Supervisor:……..… Date:……

Supervisor:…*Helman Stern*…… Date: *25.09.2011*

Chairman of Graduate Studies Committee:…… Date:……

SEPTEMBER 2011

ABSTRACT

This research deals with the development of a biometric system based on hand motion. For this purpose, a system that identifies individuals based on the attributes of their hand motion while performing a signature gesture in free air was developed. The users of the system are free to choose the hand trajectory that serves as their signature. Hand trajectories are captured using a video camera, and spatial and temporal features of the hand trajectory are extracted. Users train the system by performing a plurality of their signatures, and the system identifies the user by comparing the distances of a test signature to all signature classes stored in the database. The distance metric from a sample to a user's class is learned by Neighborhood Components Analysis. An interactive enrollment algorithm using sequential clustering is proposed which allows the system to inform the user if a chosen signature is too variant or too similar to a signature stored by another user. Three validation tests were conducted to test the system's accuracy: All users using a single pre-defined gesture (independent), each user using a personal gesture (dependent) and copycat tests for examining robustness against forgery. The accuracy obtained for identifying a single user out of user cohorts of sizes 3 to 7 was 91 to 77 percent and 98 to 92 percent for the independent and dependent systems, respectively. For three trials of attempted forgeries, correct rejection and correct acceptance rates of 81 and 94 percent were obtained. The proposed system can be integrated into a hand gesture recognition interface and used for security purposes, content adaptation, parental control, customization and more.

Index Terms— Biometrics, Distance Metric Learning, Gesture Recognition, Clustering.

ACKNOWLEDGEMENTS

I wish to express my gratitude to my supervisors, Dr. Sigal Berman and Prof. Helman Stern, who guided me throughout this project, introducing the unfamiliar territory of computer vision and machine learning. Their experience, professionalism and willingness to help assisted me in guidance and in finding the best methods and ways to perform this analysis; I would also wish to thank my lab colleagues, Mrs. Eti Vaturi, Ms. Merav Shmueli, Mr. Tom Godo, Mr. Kiril Smilanski, Ms. Shani Talmor, Mr. Noam Geffen and Dr. Darya Frolova, who assisted in brainstorming, patience, ideas and encouragement.

This research was partially supported by Deutsche Telekom AG.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

HMI	*Human Machine Interaction*
HCI	*Human Computer Interaction*
PCHIP	*Piecewise cubic Hermite polynomial*
PCA	*Principal Components Analysis*
PDM	*Point Distribution Model*
ASM	*Active Shape Models*
NCA	*Neighborhood Components Analysis*
FAR	*False Acceptance Rate*
FRR	*False Reject Rate*
kNN	*k-Nearest-Neighbor classifier*
SVM	*Support Vector Machines classifier*
HMM	*Hidden Markov Models classifier*
BSAS	*Basic Sequential Algorithmic Scheme for clustering*
TTSAS	*Two-Threshold Sequential Algorithmic Scheme for clustering*
x	*x coordinate of a point*
y	*y coordinate of a point*
z	*z coordinate of a point*
A	*Projection Matrix of NCA*
b_k	*Shape vector of sample k (PDM)*
x_k	*feature vector of sample k*
X	*feature matrix*
P	*Eigenvectors matrix (PDM)*
c	*Curvature vector*
D_j	*Distance from a sample to user j*
D	*Minimum distance from a sample to a user*
F_i	*The expected number of points on segment i*
θ_1	*The minimum distance threshold for accepting a sample to a cluster*
θ_2	*The maximum distance threshold for creating a new cluster*
θ_3	*The minimum distance threshold for merging two clusters*

CHAPTER 1: INTRODUCTION

1.1 Background

Biometric systems allow identification or verification of an individual based on physical or behavioral traits. Such systems are required in many settings mainly for security purposes but also for customization of interfaces and content. Although systems based on physical information are considered reliable, emerging systems that utilize the behavior of an individual offer increased intuitiveness and usability. These systems are based on traits such as voice, typing rhythm and motion. In many cases motion based systems allow the user to be identified from distance, and do not require any special devices. The recognition can be performed using a video camera, motion sensor or even a computer mouse.

Identifying persons using behavioral data is a major challenge. Jain et al. [1] note that a user identification system must follow the four following rules: universality, distinctiveness, permanence and collectability. These rules are easy to handle with physical based systems, but challenging when it comes to behavioral based systems. Distinctiveness, for example, requires that the system will be able to distinguish between many different users. As in many other systems, increasing the number of users (or classes) usually results in a reduction in accuracy.

Current human machine interaction (HMI) devices, e.g., mice, keyboards and remote controls are considered effective, but still many problems arise when using them. More advanced HMI are being developed for the purpose of better interactions between people and machines, giving the ability to use the computer without having to practice or study a new language, to an increasing number of people. Other catalyzers of the HMI industry are computer games, telerobotics, artificial intelligence methods and augmented reality devices.

In contrast to a few decades ago, when system designers considered the machine first, nowadays designers and researches understand that a system must put the user at the center in order to make the interaction usable, universal, intuitive, efficient and satisfying. Gesture recognition systems are one of these user-centered HMI. A gesture is a form of communication between individuals and as such, it brings more intuitiveness than using a mouse of any other device common today.

1.2 Research Objectives and Contribution

This research aims to combine the two worlds of user-centered HMI and user identification. Manually entering a password or pin code common today is not necessarily here to stay. Other methods of identification such as face identification or identification using gestures can be much more intuitive. A hand motion based identification system can be used, for example, in an operating system's welcome screen that has multiple users. This type of system can also be used for gesture recognition for TV systems such as the one proposed in [2], giving the ability to customize content, menus and features. Behavioral based systems have another advantage over several physical based ones: they don't keep true physical information on the user. Attributes of the movement are the only information kept inside the system.

The main objective of this research is to develop a system that can identify a predefined number of individuals using a position sensor, based on the characteristics of a hand motion signature made by them. So far, not many attempts to identify persons using characteristics of movement were made. Another major objective of this research is to show that different individuals indeed perform gestures in a different manner. This research aims to allow the user to select the hand motion signature of his or her choice.

There are two major innovations involved in this research. The first deals with the ability of the user to select his or her own signature without any constraint of a predefined shape. The second innovation is an interactive learning algorithm that provides online feedback during enrollment, and offers information on whether the selected signature is too similar to an already stored signature, or whether it is too noisy for the system to handle.

It is presumed that individuals can be distinguished only by the characteristics of their hand gesture using a robust tracking and classification system. The main hypothesis of this research is that a hand motion signature that is perceived to be identical when performed by two individuals can be distinguished using the methods and equipment used in this research.

In order to examine this hypothesis, two major experiments were made: An experiment in which participants performed 3 predefined shapes: X, O and 'line'; and an experiment whose participants selected the signature of their choice. The first experiment was conducted for testing the possibility of identifying users according to the characteristics of their hand movement. The second was conducted for the purpose of examining a system with practically no a *priori* data on the input. A short video demo of the system is available at: http://bit.ly/OmriMendels

In addition to the academic interest of testing the above hypothesis, this system can also be adapted for commercial use. For that purpose, a patent (named "Real time user identification by hand motion signatures") was issued for the system and is currently pending.

1.3 Research Scope and Limitations

In this study the ability to identify a user out of a group of 3 to 7 users, was examined. The classification was based solely on features of the hand trajectory. No other movement or physical information was gathered. Experiments were made on independent participants and not on family members where gestures of siblings of the same gender and of a similar age may be more difficult to distinguish. A working system was developed and optimized based on the experiment's results, yet the final and working system was not put to an extensive field test. The enrollment system was not tested with the interactive recommendation system in a real-time scenario.

1.4 Thesis Outline

This thesis is arranged as follows: Chapter 2 provides a literature review on methods that were used, ideas that were considered and systems that are similar to the developed system. Chapter 3 describes a prototype system with a preliminary experiment to test it. Chapter 4 describes the full methodology developed for this thesis. Chapter 5 describes the final experiments, chapter 6 describes the results of the experiments and a discussion, and the conclusion is provided in chapter 7.

CHAPTER 2: BACKGROUND

2.1 Overview

This literature review describes the methods that were used during the research process. Most of the methods that are described in this chapter were used in the final system. Some of the methods and ideas were considered throughout the course of development, but are mentioned here since they were used during the process of understanding the problem and finding a proper solution to it. These also supply a preliminary understanding to topics such as human motion, geometry and machine learning.

A survey of motor control studies is described here, even if none were eventually used for the identification system. Section *2.2* describes some relevant studies in the field of user identification. Sections *2.3* and *2.4* describe general subjects concerning this work such as biometrics, human machine interaction and gesture recognition. Section *2.5* generally describes movements – the way individuals perform movements and how these movements can be modeled and used for the purposes of this research. The additional sections discuss different machine learning, geometric and interpolation methods that are to be used in this research.

2.2 Biometrics

"A wide variety of systems requires reliable personal recognition schemas to either confirm or determine the identity of an individual requesting their services." [1]

User identification systems are mainly based on biometrics – the field of statistics in biology. These systems mainly rely on one of two types of information: *Physiological* data that relates to information such as fingerprints, DNA and face recognition; and *Behavioral* data that relates to a behavior of an individual. Examples for behavioral feature recognition are gait recognition [3] where the opportunities, advances and challenges in gait recognition are described; voice recognition [4], Rabiner's famous speech recognition and Hidden-Markov-Models work; and typing rhythm [5] where a keystroke based user authentication that is based on compact data is proposed. Additionally, a survey of solutions to identify theft using keystroke dynamics is proposed in [6]. User identification that is based on behavior is

often called "Behaviometrics" (e.g., in [7], where a model of user verification using kinematics is described).

User identification systems must satisfy the following requirements: [1]

- *Universality*: each user should have the characteristic
- *Distinctiveness*: any two persons should be sufficiently different in terms of the characteristic
- *Permanence*: the characteristic should be sufficiently invariant over a period of time
- *Collectability*: the characteristic can be measured quantitatively.

2.2.1 User identification systems

There are two types of user identification system: *verification* systems and *identification* systems.

- *Verification* – A one to one comparison of a captured biometric with a stored template to verify that the individual is who he claims to be.
- *Identification* – A one-to-many comparison of the captured biometric against a biometric database in attempt to identify an unknown individual.

As mentioned in [8], User identification requires two phases: enrollment (training) and recognition (the latter can be verification or identification). A block diagram of each step is shown in figure 1. In a verification task, an enrolled user claims an identity and the system verifies the authenticity of the claim based on her biometric feature. An identification system identifies an enrolled user based on her biometric characteristics without the user having to claim an identity.

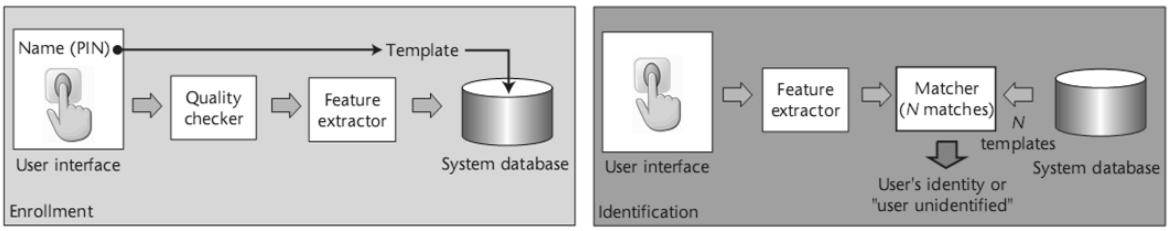

Figure 1 - Block diagram of enrollment and identification in user recognition [8]

Biometric Recognition is similar to any pattern recognition system, since in both a feature extraction or selection is to be made, followed by a classification of the features to a certain class. Gesture recognition systems classify the features as gestures, while biometric recognition classifies the features as a certain individual activating the system.

2.2.2 User identification by movement traits

The major problem with movement is the inability to perfectly replicate it. Fingerprints, for example, are almost constant and two samples will produce the same results with very little variance. Movement, on the other hand, has variance and in order to use movement to identify an individual, one has to statistically estimate a representing model of the movement that was made. Most identification systems nowadays rely on input with less variance (e.g., iris recognition or even voice recognition).

One example of user identification using movement is gait recognition. In a research by [3], a method for biometric identification using gait (i.e. a particular way or manner of moving on foot) for security purposes is proposed. The general architecture of this system is similar to the one used in gesture recognition or other biometric recognition systems, as seen in figure 2.

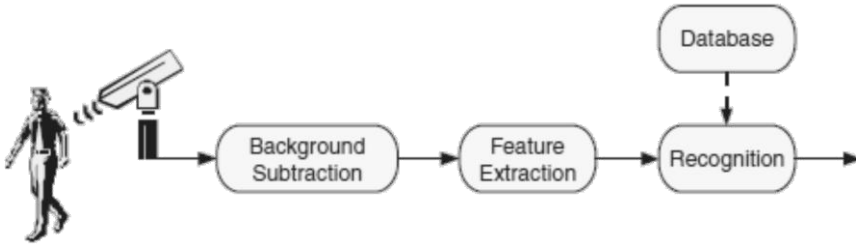

Figure 2 - gait recognition system architecture
[3]

The features used in this project are for example: contours, angles, projections, silhouettes. This research offers different paths for gait based identification, and does not describe a working system that actually identifies different persons. In their conclusion, Boulgouris [3] claim that the identification using gait is still an open research topic and current systems are not robust enough to accurately identify an individual out of many. A survey of gait recognition systems in [9] shows an average accuracy of 88 percent for identifying a person walking indoors.

Another example of user identification using human kinematics is presented by [7]. Sriwarno offers a method to classify human behavior using movements. In this research, videos of subjects squatting were analyzed, and different types of squatting techniques were noticed. The author showed that since different participants used different squatting techniques, they can be identified by the way they perform the movement. Figure 3 shows the motion profiles of 3 different subjects.

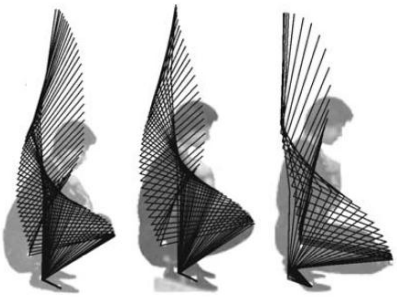

Figure 3 - Movement models used for identification
[7]

There were many studies aiming to use the computer mouse as a motion based user identification device. A feasibility study made by Weiss et al. [10] asked users to click on 25 different buttons and a system collected different events such as clicks and mouse movements. This served as a feature vector for a kNN classification algorithm. In [11], mouse movements for re-authentication of users were used. Mouse events such as moves, clicks and wheel spin were collected and a decision tree based classifier decided whether this behavior was anomaly or not.

Several studies attempted to use hand motion traits for identification. A feasibility study for such a system appears in [15] in which users perform one of four pre-defined signature gestures. This system and the one developed by [16] use accelerometers for capturing the motion. No studies were found where the hand movement is captured by camera using computer vision algorithms. A relevant field of study that usually employs similar methods to the proposed system is the user verification by written signatures. These systems such as the one proposed in [17] utilize tablet screens. In this system spatial and temporal features are extracted and a dynamic time warping technique is applied for the classification of the signature.

2.2.3 Motion based person identification by humans

An interesting approach in artificial intelligence is to compare computerized methods to human ones. A few studies on the human capabilities of identifying people using motion were held. These researches rely on the fact that humans have a lifetime of experience watching other people move. Such extensive visual experience is thought to selectively enhance visual sensitivity to the human movement [12] . Different studies, such as the one made in [13] checked the connection between motion and emotions perception and interpretation. Subjects were shown point light depictions of themselves, their friends and

strangers. Results showed that sensitivity to one's own movement was the highest, and friends were also discriminated from strangers. In [14], a similar research was performed on gait movements. 7 individuals served as walking models and 41 markers were placed on their body. 18 observers tried to discriminate between the 7, in 3 different angles of view (frontal, half-profile and profile). Different parameters were normalized (size, frequency of walking, shape) and tests were made with normalization and without it. Performance saturated at approximately 90%.

2.3 Human Machine Interaction

Human Machine Interaction (HMI) / Human Computer Interaction (HCI) is defined by Baecker et al. as "*a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them*" [18]. These interactions include current input & output interfaces such as keyboards, mice, remote controls, printers and monitors.

The work on developing HMI includes design and implementation of interactive systems, involving users and machines. This necessarily involves work that has traditionally developed separately in various disciplines, including computer science, psychology, design theory, social sciences, work domain analysis and creative design [19].

Recently, a new field called IHCI or HCII (Intelligent Human Computer Interfaces) is emerging. This field combines HCI with other technologies such as artificial intelligence and augmented reality [19]. HCII focuses more on the user experience and its usability. Gesture recognition based HCI is one example of an intelligent human computer interface.

2.3.1 Usability in human machine interfaces

Usability is the capability in human functional terms to be used easily and effectively by the specified range of users, given specified training and user support, to fulfill a specified range of tasks, within the specified range of environmental scenarios [20].

In spite of changes in the nature of computing, remnants of old thinking still remain. In former days, when the CPU was at the heart of a system, designers naturally talked of ‘‘terminals” and ‘‘peripherals”. This was in this period that people began to use the term ‘‘end user”. The unconscious symbolism is both a symptom and a cause; the ‘‘end” user at the ‘‘terminal” was often the last person to be considered in the design of the system. It is

important to develop a new view of computing systems, and to look at the user in a different light. Taking this view of computing, the centre of a system is the user. [21]

2.3.2 Gesture based HMI

The evolution of user interface (UI) witnessed the development from UIs based on keyboards to a GUI (graphical user interface) based on mice. New computerized features such as augmented and virtual reality require better interaction between the user and the machine. Moreover, an intuitive language of communication needs to be implemented in order to reach many more populations such as elderly persons. In current applications devices such as keyboards, mice, wands, joysticks and remote controls are still the most popular and dominant. However, they might be inconvenient and unnatural.

The use of human movements, especially hand gestures, has become an important part of HCII in recent years, which serves as a motivating force for research in modeling, analyzing and recognition of hand gestures. Many techniques developed in HCII can be extended to other areas such as surveillance, robot control, teleconferencing, security [22], a survey of gesture recognition systems [23]; and medicine such as [23] where a gesture recognition system was developed as an HCI in an operating theater, allowing the surgeon to handle images without having to touch any device.

2.4 Gesture Recognition Systems

Gesture Recognition is a general name for devices that allow the user to interact using hand, fingers or body gestures. A gesture is defined as the use of motions of the limbs or body as a means of expression; a movement usually of the body or limbs that expresses or emphasizes an idea, sentiment, or attitude [26]. In the following paragraphs, a typical topology of gesture recognition systems will be described.

Recognizing gestures is a complex task which involves many aspects such as motion modeling, motion analysis, pattern recognition, machine learning, and even psycholinguistic studies. [24]. In such systems, the designer must think of proper gestures to be used that will be easy and ergonomic for the user, and that will be identifiable by the system. Stern et al. [25] constructed a framework for developing a gesture vocabulary that considers both the user side and the system side aspects of the problem.

In order to construct a gesture based HMI, one must develop different programs that will collect the data from the camera, extract information from the input and decide which gesture if any was performed.

Figure 4 describes a typical gesture recognition system, with a gesture model, gesture analysis, gesture recognition and output functions. The following paragraphs will explain the analysis, feature extraction and the classification of gestures.

Figure 4 - Typical dynamic gesture recognition system architecture

The interpretation of gestures requires that dynamic and/or static configurations of the human hand, arm, and even other parts of the human body, be measurable by the machine. First attempts to solve this problem resulted in mechanical devices that directly measured hand and/or arm joint angles and spatial position. This group is best represented by the so-called glove-based devices. The interface requires the user to wear a cumbersome device that hinders the ease and naturalness with which the user can interact with the computer controlled environment. Even though the use of such specific devices may be justified by a highly specialized application domain, (e.g., simulation of surgery in a virtual reality environment) the “everyday” user will certainly be deterred by such cumbersome interface tools. This has spawned active research toward more natural HMI techniques.

Potentially, any awkwardness in using gloves and other devices can be overcome by using video-based noncontact interaction techniques. This approach suggests using a set of video cameras (or a single camera) and computer vision techniques to interpret gestures. The advantages of the vision-based interface have resulted in a burst of recent activity in this area. Other factors that may have contributed to this increased interest include the availability of fast computing that makes real-time vision processing feasible and recent advances in the necessary hardware, as well as the reduction of the price of these instruments.

Research and development of gesture recognition systems started with a focus on the recognition of static hand gestures or postures. A variety of models, most of them taken from approaches in pattern recognition have been utilized for that purpose [24]. Many researches nowadays focus on dynamic hand gestures, which are captured by forming a trajectory of the

hand using segmentation of the hand and an algorithm for tracking the center of the hand's pattern.

2.4.1 Gesture analysis

The goal of gesture analysis is to estimate the parameters and features of the gesture model using measurements from the video images of a human operator.

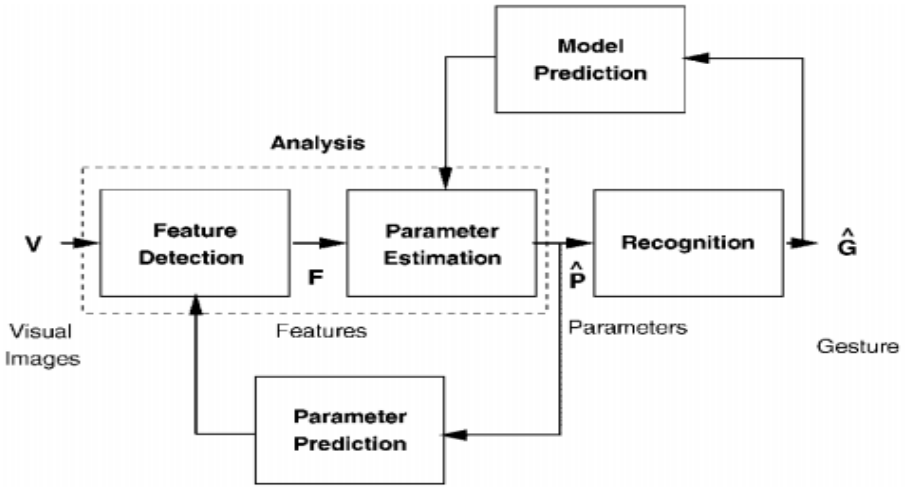

Figure 5 - Gesture analysis and recognition
[24]

As seen in figure 5, features are extracted from visual images. Model parameters are estimated and possibly predicted. Gestures are recognized in the recognition stage. Recognition may also influence the analysis stage by predicting the gesture model at the next time instance.

2.4.2 Feature extraction

Prior to classification, a phase called feature extraction is usually performed. Fukunaga [27] describes the feature extraction as a process of mapping the original measurements into more effective features. If the mapping is linear, the mapping function is well defined and our task is simple. Unfortunately, in most cases the features are not linear functions of the measurements and then the problem is to find a proper nonlinear mapping function of the given data. Since there is no general algorithm to generate nonlinear mapping function, the feature extraction becomes very much problem oriented.

These features must define the model in a meaningful and compact way. Meaningful features will make the classification stage much easier and can turn a problem from an infeasible one to a feasible one.

2.4.3 Gesture classification

Gesture recognition is the phase in which the data analyzed from the visual images of gestures is recognized as a specific gesture. The trajectory (in case of dynamic gestures) in the model parameter space (obtained in the analysis stage) is classified as a member of some meaningful subset of that parameter space. Features from the analysis stage are inserted into a feature vector, and this vector is compared to features of different gestures predetermined in the system. This stage is called classification. A classification algorithm compares the feature vector's data with data in different classes (each class represents a gesture).

Depending on the system in question, all of the extraction and classification functions have varying success rates that are influenced by the input received. For example, a glove or sensor based system has more robustness than a vision based system, and a 3D vision system is inherently more robust and accurate than a 2D vision system, since hand segmentation can be done by depth and not only by other cues such as motion or color.

2.5 Motor Control

Motor control is one of the most amazing phenomena in nature. Moving a robot with the same amount of degrees-of-freedom as the human hand requires a descent amount of computational power and memory. Moreover, performing this action in an optimal way can be very difficult. How humans perform an action in a near optimal way is not fully known to scientists. Winter [28] claims that the scientific approach to biomechanics has been characterized by a fair amount of confusion, but many of them agree that this type of movement is indeed optimal. There are a few examples to this optimality: There are infinite ways to move the hand from one point to another, but the central nervous system (CNS) picks the shortest one; It is assumed that the CNS tries to minimize the jerk (change in acceleration) during a movement [29] and the torques used [30].

Kawato et al. [31] claim that in order to perform motor actions, the CNS must solve the following questions:

1. The determination of a desired trajectory in the visual coordinates.
2. The transformation of its coordinates to the body coordinates.

3. The generation of motor command.

Feldman [32] distinguishes between different variables of the movement. The mechanical variables of the actions that the CNS cannot control are called state variables (SVs) (e.g., the gravity of earth). The variables that are controlled by the CNS are called control variables (CVs) (e.g., moving a muscle). In order to understand the nature of voluntary motor action, we must first understand which CVs are used by the CNS and which SVs are being taken into consideration while performing the movement [32].

2.5.1 Computational motor control (CMC)

Computational motor control is the science of constructing motor control models using mathematical and computational techniques. These techniques involve control engineering methods such as adaptive control, computational methods such as artificial neural networks and geometrical approaches such as affine spaces [33] and geometric algebra [34]. The following paragraphs will describe different models of CMC.

2.5.2 Minimum jerk method

The Minimum Jerk Method was presented by [29] . The purpose of this research was to find a mathematical model describing a two axis movement. The basic idea behind it is that the CNS tries to perform an optimal movement, and a certain mathematical optimization model will probably act the same way. In order to achieve optimality, it was assumed that the CNS tries to minimize the *jerk*. The jerk is the 3rd derivative of the trajectory.

The mathematical model describing the movement is described in equation 1:

$$C = \frac{1}{2}\int_0^{t_f}\left(\left(\frac{d^3x}{dt^3}\right)^2 + \left(\frac{d^3y}{dt^3}\right)^2\right)dt$$

Equation 1 - the movement's proposed model [29]

The optimization results in a fifth order polynomial in time both for x(t) and y(t).
Assuming the movement to start and end with zero velocity and acceleration, the following expressions for hand trajectory are obtained:

$$x(t) = x_0 + (x_0 - x_f)(15\tau^4 - 6\tau^6 - 10\tau^3)$$
$$y(t) = y_0 + (y_0 - y_f)(15\tau^4 - 6\tau^6 - 10\tau^3)$$

Equation 2 - the two coordinates values as a function of t [29]

where $\tau = t / t_f$, x_0, y_0 are the initial hand position coordinates at t=0 and x_f, y_f are the final hand position coordinates at $t=t_f$.

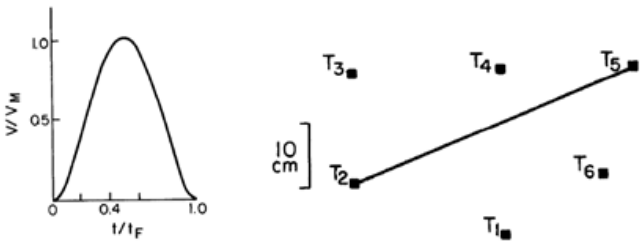

Figure 6 - Velocity and path profiles
[29]

Figure 6 shows the hand trajectory and velocity according to the model. Except for straight lines, curved lines were also examined and a mathematical model was developed.

2.5.3 Minimum torque method

Uno et al. [30] used the minimum jerk model to establish a new model. Since the minimum jerk model does not take any dynamics into consideration, they offered a new model that attempts to find an optimal trajectory with minimum torques. This is under the assumptions that the CNS prefers smooth muscle torques, and tries to minimize this value instead of changes in acceleration. The minimum torque model is defined as follows:

$$C_T = \frac{1}{2}\int_0^{t_f}\sum_{i=1}^{n}\left(\frac{dz_i}{dt}\right)^2 dt$$

Equation 3 - the proposed minimum torque model [30]

Where Z_i is the motor command (torque) fed to the i-th actuator (muscle) out of *n* actuators. C_T is the sum of square of the rate of change of torque integrated over the entire movement. This model is closely related to the minimum jerk model proposed by [29] mainly because acceleration is locally proportional to torque at zero speed.
Since describing n actuators is extremely complicated, Uno et al. offered a robotic manipulator for the test instead of the real musculoskeletal system.

The following functions enable us to calculate the actuated torque of the two joints (z_1 and z_2)

$$z_1 = (I_1 + I_2 + 2M_2 L_1 S_2 \cos\theta_2 + M_2 (l_1)^2)\ddot{\theta}_1$$
$$+ (I_2 + M_2 L_1 S_2 \cos\theta_2)\ddot{\theta}_2$$
$$- M_2 L_1 S_2 (2\dot{\theta}_1 + \dot{\theta}_2)\dot{\theta}_2 \sin\theta_2 + b_1\dot{\theta}_1$$

$$z_2 = (I_2 + 2_2 L_1 S_2 \cos\theta_2)\ddot{\theta}_1 + I_2\ddot{\theta}_2$$
$$+ M_2 L_1 S_2 (\dot{\theta}_1)^2 \sin\theta_2 + b_2\dot{\theta}_2$$

Equation 4 - the torque in joints z_1 and z_2 [30]

Where M_i, L_i, S_i, and I_i represent the mass, length, distance from the center of mass to the joint and the rotary inertia of the link *i* around the joint, respectively. b_i and z_i represent the coefficients of viscosity and the actuated torque of the joint i.

2.5.4 Affine and Equi-Affine geometrical spaces

A recent study of geometry and invariance in motions show that the human brain uses a mixture of geometries when planning motion [33]. These geometries are the Euclidean geometry, the affine geometry and the Equi-Affine geometry, which is similar to the affine but have the constraint of preserving an object's area under linear homogenous transformations. The affine geometry is more suitable for modeling human movements since it is possible to deal with curves and points in an intrinsic way [35]. Together with the conclusions of [33] , it is reasonable to perform human analysis not only in the Euclidean geometrical space.

2.5.5 Additional movement features

Rhythmic movement such as walking, chewing or waving the hand, is treated differently than other discrete movements. Apparently, the motor control of such movements is different than the control of discrete movements. Schaal et al. [36] define the differences between a rhythmic movement and a discrete movement (e.g., grasping). Some differences can be seen in figure 7.

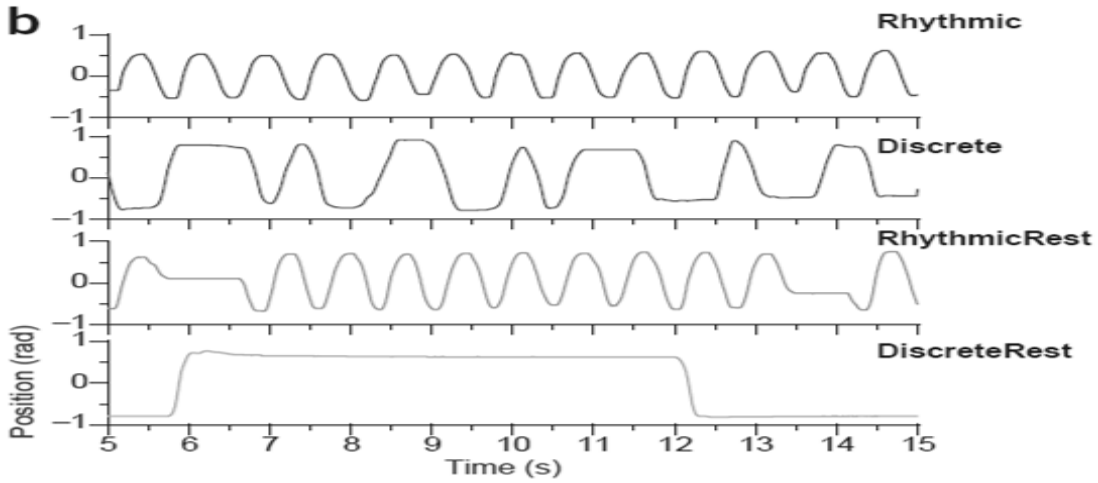

Figure 7 - Differences between rhythmic, discrete, rhythmic with a rest and discrete with a rest movements. [36]

More features that define the human arm movement and can be used for analysis of this movement are velocity profiles, trajectories, time to completion (of a gesture or movement), maximum velocity, and acceleration. Some of these features are addressed in a work by Atkeson et al. [37].

Velocity and motion can also be modeled through *Motor Algebra*, as described in [34]. Since velocity of a rigid body is not a generalization of a point of mass velocity, a better representation needs to be developed. The geometric algebra model offers a formulation both for the dynamics and the kinematics of the human motion as a rigid body.

Curvatures are also features that can be used. Curvatures and hand speed are described in [29]. Hand Speed T is defined as $T = \sqrt{(\dot{x})^2 + (\dot{y})^2}$. Trajectory curvature is defined as $C = (\dot{x}\ddot{y} - \dot{y}\ddot{x}) / (\dot{x}^2 + \dot{y}^2)^{3/2}$ where $\dot{x}$ and $\dot{y}$ are the time derivatives of the x and y coordinates of the hand in the plane and $\ddot{x}$ and $\ddot{y}$ are the corresponding accelerations.

2.6 Geometric Alignment

For a reduction of unnecessary variability caused by the location, distance and angle of the user in front of the camera, it is advisable to align samples to the same initial state. Two methods for alignment will be described: Procrustes analysis [38], a straightforward method for alignment, and Active Shape Models [39], where a mean shape is calculated, and all samples are translated, scaled and rotated in order to minimize the total Euclidean distance

between the sample and the mean shape. The process is iterated until convergence of the mean shape, and gives more weight to stable points.

2.6.1 Procrustes analysis

Procrustes analysis or Procrustes superimposition is a method of removing the translation, scaling and rotation components of a shape in order to obtain a similar placement in size and location, which should be achieved prior to a comparison of shapes. Procrustes analysis is a straightforward method that aligns a shape to a mean shape uniformly. It is originally described in [38] but a more intuitive explanation will be given here. For translation, the values of each point's coordinates of the mean shape are subtracted from the point's coordinates values of the aligned shape:

Given k shapes of two dimensions: $(x_1, y_1),(x_2, y_2),...,(x_k, y_k)$, the mean shape is $(\bar{x},\bar{y})$: $\bar{x}=\frac{1}{k}\sum_{i=1}^{k} x_i$, $\bar{y}=\frac{1}{k}\sum_{i=1}^{k} y_i$. By applying $(x_i, y_i)\rightarrow(x_i-\bar{x}, y_i-\bar{y})$ we get the new translated shape $(x_i-\bar{x}, y_i-\bar{y})$.

Following translation, the scale component can be removed by scaling the object such that the root mean square distance from the shape (which is described as a n dimensional vector) to the translated origin is 1. Let s_i be the Euclidean distance from shape *i* to the translated origin (in this example shape *i* has 2 dimensions): $s_i = \frac{1}{2}\sqrt{(x_i-\bar{x})^2+(y_i-\bar{y})^2}$. By dividing the translated point by s, the scale component is equal to 1 and therefore removed. $(x_i, y_i)\rightarrow((x_i-\bar{x})/s,(y_i-\bar{y})/s)$.

2.6.2 Active shape models

Active Shape Models (ASM) are described in [39]. It allows translation, scaling and rotation and gives more weight to points that are stable throughout the training set of shapes.

Let x_i be a vector describing the n points of the i^{th} shape in the set:

$$x_i = (x_{i,0}, y_{i,0}, \ldots x_{i,n-1}, y_{i,n-1})^T$$

Let $M(s,\theta)[x]$ be a rotation by q and scaling by s. Given two similar shapes, x_i and x_j, we can choose q_j, s_j and a translation (t_{xj}, y_{xj}) mapping x_i into $M(s,\theta)[x]+t_j$ so as to minimize the weighted sum in equation 5:

$$E_j = (x_i - M(s_j,\theta_j)[x_j] - t_j)^T W (x_i - M(s_j,\theta_j)[x_j] - t_j)$$

$$\text{Where } M(s,\theta)\begin{bmatrix} x_{jk} \\ y_{jk} \end{bmatrix} = \begin{pmatrix} (s\cos\theta)x_{jk} - (s\sin\theta)y_{jk} \\ (s\sin\theta)x_{jk} - (s\cos\theta)y_{jk} \end{pmatrix}$$

$$\text{And } t_j = (t_{x1}, t_{y1}, \ldots t_{xj}, t_{yj})^T$$

Equation 5 - Minimization function for a two sample alignment [39]

W is a diagonal matrix of weights for each point. These weights can be proportional to the tendency of the points to move inside the training set. Points that move the least will get a small weight in comparison with points that have a large distribution inside the training set. Weights can be calculated using equation 6 where V_{Dki} is the variance of Euclidean distances over the set of shapes:

$$w_k = \frac{1}{\sum_{i=1}^{n} V_{Dki}}$$

Equation 6 - forming the weights for the minimization function [39]

The algorithm for aligning the set until convergence:

1. Rotate, scale and translate all shapes to the mean (or to the first shape in the set)
2. Calculate a new mean shape
3. Realign every shape to the mean shape
4. Repeat 2-3 until convergence.

A least-square approach (differentiating with respect to the variables ax, ay, tx, ty) leads to a set of 4 equations:

$$\begin{pmatrix} X_2 & -Y_2 & W & 0 \\ Y_2 & X_2 & 0 & W \\ Z & 0 & X_2 & Y_2 \\ 0 & Z & -Y_2 & X_2 \end{pmatrix} \begin{pmatrix} a_x \\ a_y \\ t_x \\ t_y \end{pmatrix} = \begin{pmatrix} X_1 \\ Y_1 \\ C_1 \\ C_2 \end{pmatrix}$$

Where:

$$a_x = s\cos\theta \qquad C_1 = \sum_{k=0}^{n-1} w_k (x_{ik}x_{jk} + y_{ik}y_{jk})$$

$$a_y = s\sin\theta \qquad C_2 = \sum_{k=0}^{n-1} w_k (y_{ik}x_{jk} + x_{ik}y_{jk})$$

$$X_i = \sum_{k=0}^{n-1} w_k x_{ik} \qquad Z = \sum_{k=0}^{n-1} w_k (x_{jk}^2 + y_{jk}^2)$$

$$Y_i = \sum_{k=0}^{n-1} w_k y_{ik} \qquad W = \sum_{k=0}^{n-1} w_k$$

Equations 7 - the least square approach for minimizing the sum Ej [39]

2.6.3 Justification for scaling shapes

In human perception and motor control, it is assumed that individuals perceive gestures made in different sizes as the same gesture. This is based on the human motor control characteristic

of movement invariance [40] and the human brain use of affine and equi-affine geometries that allow stretching or scaling of movements [33]. Therefore a basic assumption is that the alignment of shapes, and particularly scaling, does not affect the system's identification capabilities.

2.7 Interpolation

Interpolation is the process of constructing a continuous curve that best fits a series of data points, similar to regression analysis. Three common interpolation techniques are: linear interpolation; splines; and cubic Hermite interpolation. One of the first non linear interpolation techniques was the polynomial interpolation. Nowadays polynomial interpolation is mostly of theoretical value. Faster and more accurate methods such as splines and Hermite splines were developed. These methods are piecewise polynomial, i.e. constructed of several polynomials that are connected to form one continuous curve. [41]

2.7.1 Polynomial interpolation/regression

The linear regression model is $y = X\beta + \varepsilon$. A polynomial regression model could for example be:

- A second order polynomial in one variable: $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \varepsilon$.
- A second order polynomial in two variables: $y = \beta_0+\beta_1 x_1+\beta_2 x_2+\beta_{11} x_1^2+\beta_{22} x_2^2 +\beta_{12} x_1 x_2 +\varepsilon$.
- Any order polynomial with any number of variables.

2.7.2 Piecewise polynomial interpolation (splines)

Piecewise polynomials are useful when a low-order polynomial provides a poor fit to the data, regardless of the order of the polynomial. This problem usually arises when the function behaves differently in different parts of the range of independent variable. A usual approach is to divide the range of x into segments and fit an appropriate curve in each segment.

Spline functions offer a useful way to perform this type of piecewise polynomial fitting. The connections or borders between polynomials are called 'knots'.
For example, a cubic spline [42]:

h knots, $t_1 < \ldots < t_h$, with continuous first and second derivatives:

$$E(y) = S(x) = \sum_{j=0}^{3} \beta_{oj} x^j + \sum_{i=1}^{h} \beta_{oj} (x - t_i)_+^3$$

$$(x - t_i)_+ = \begin{cases} (x - t_i) , & x - t_i > 0 \\ 0, & x - t_i \leq 0 \end{cases}$$

Equation 8 - Cubic Spline [42]

2.7.3 Piecewise cubic Hermite interpolation

The piecewise cubic Hermite interpolation is a third degree spline that consists of two control points and two control tangents for each polynomial. Using the interval t=(0,1), the cubic hermite interpolation is defined as $p(t)$ in equation 9:

$$p(t) = (2t^3 + 3t^2 + 1)p_0 + (t^3 - 2t^2 + t)m_0 + (-2t^2 + 3t^2)p_1 + (t^3 - t^2)m_1$$

Equation 9 - A cubic Hermite polynomial

Where p_0 and p_1 are the control points, and m_0 and m_1 are the control tangents.

By creating such polynomial for each two points, we get a piecewise polynomial that interpolates the data points.

By integration, one could find arc lengths and place points with an equi-distance spacing.

2.8 Pattern Recognition

Statistical classification is the act of differentiating between different types of data. Input data is placed into groups based on quantitative information on one or more characteristics inherent in the data. Statistical classification is also called machine learning. Applications of these methods can be found in data mining, computer vision, gesture recognition [25], speech recognition [4] and biometric identification [43]. Pattern recognition is usually separated to supervised and unsupervised learning, whereas other methods such as semi-supervised learning or reinforcement learning also exist.

2.8.1 Approaches to pattern recognition

There are three main types of statistical classification algorithms: supervised classification, unsupervised classification and reinforcement learning. In supervised classification, the designer of the system gives the learning algorithm samples with an output. The classifier changes its parameters (e.g. weights) or entire model in order to fit the input vector to the result needed. When classifying a new sample, it is assumed that the new sample's features are similar to the training data's features and it will be classified to the most suitable class.

Unsupervised learning uses elements known from the data in order to group the data into clusters.

2.8.2 Supervised Learning

Supervised (or classification) uses a training data acquired during the initiation of the system together with the class information or label of each sample. New data that needs to be classified will be classified according to the classes constructed in the training phase. Training data includes input data and a desired output (i.e. the true classification of the each input sample). In this kind of classification, the designer of the system decides a *priori* on the different classes that will be used.

A basic scheme for building a supervised classification model is offered in [44] and described in figure 8.

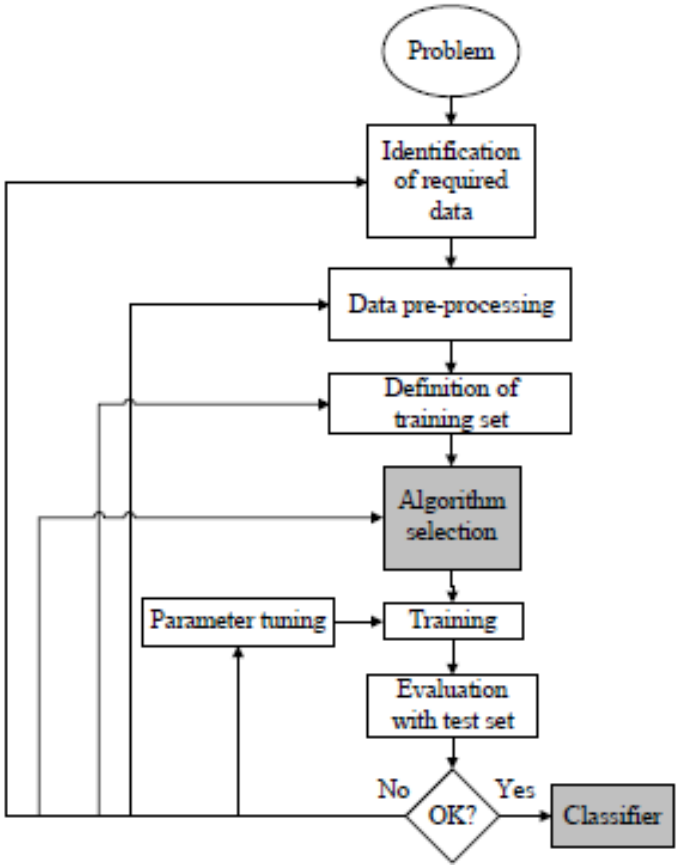

Figure 8 - The process of supervised machine learning
[44]

2.8.3 Unsupervised Classification

In contrast to the supervised classification, unsupervised classification requires very few inputs into the classification processes. The computer selects natural groups of data based on their features. However, an unsupervised classification algorithm still requires user interaction. This occurs after the classification has been performed. In unsupervised classification, the user attempts to assign information classes (i.e. meaningful names or properties of classes) to the classes the system has created. In this process, several potential problems exist. The first is that some of the classes may be meaningless as they don't relate to any information class. In other instances, a single informational class may be split among two computer made classes [45]. Cluster analysis is a form of unsupervised classification, and will be described in the next sections.

2.8.4 Pattern recognition methods

In this section four different pattern recognition methods will be presented: Support Vector Machines (SVM), Hidden Markov Models (HMM) and k-Nearest-Neighbor. In the next section, cluster analysis will be described.

Support Vector Machines

Support Vector Machines (SVM) is a relatively new classification method [46]. In this method, features are non-linearly mapped to a very high dimension feature space. In this feature space a linear decision surface is constructed. Special properties of the decision surface ensure high generalization ability of the learning machine. Intuitively, a good separation is achieved by the hyper plane that has the largest distance to the nearest training data points of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier. A special characteristic of SVM is that the solution to a classification problem is represented by the support vectors that determine the maximum margin hyper-plane. A basic illustration of the process can be seen in figure 9.

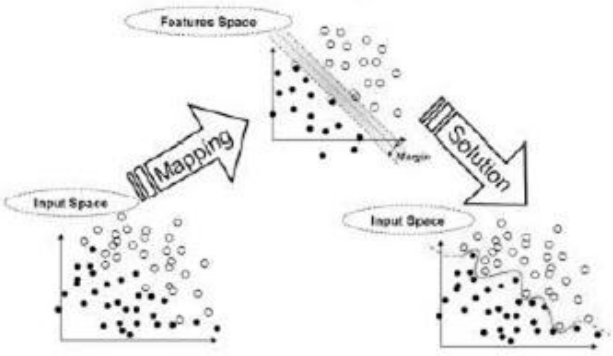

Figure 9 - An illustration of the SVM algorithm

The work in [47] compare SVM with 12 classifiers on 21 data sets, where SVM yielded good performance but did not outperform the other methods on each dataset. Moreover, it tends to require long training time and when the input environment changes in time. In this case, the accuracy decreases because the weights are fixed, preventing it from adapting to the changing environment

Applications of SVM

An example of a work with SVM and 3D objects can be seen in [48]. In this work SVM is used to recognize 100 different 3D patterns over 7200 images. A SVM model was developed for that purpose and the achieved results are excellent (average error rates of 0.03%). In [49] a type of SVM called MEB-SVM is used for gesture (hand pose) recognition. Berman et al. [50] used SVM and PCA for action identification. The purpose of this research was to develop a new approach for *teleoperations* – performing medical procedures using telerobotics. Human motor actions were categorized using different classifiers in the context of the object they were being performed on. Two different SVM algorithms were used: a two class soft margin SVM with linear kernel, and a two class soft margin SVM with a Gaussian kernel. The SVM models performed better than two other classifiers (J48 decision tree and 1-Nearest-Neighbor) in 5 out of 6 tasks tested.

Hidden Markov Models

Hidden Markov Models (HMM) is a common way for dynamic (or temporal) gesture recognition [51]. It is also used in speech recognition [4], and handwriting recognition. In this model, the system is represented as a markov process with unobserved states. Because of their stochastic properties (the states are considered hidden, with a probability to be in each state), the HMMs have the ability to model non-stationary signals or events. In hand

movements and gestures, the signal is represented by motion measurements of the part of the body that moves. Recognition in this method is performed by choosing the model with the highest probability after evaluating the probability value for all competing models.

In order to define an HMM completely, the following elements are needed [52]:

- The number of states of the model, *N*.
- The number of observation symbols in the alphabet, *M*. If the observations are continuous then *M* is infinite.
- A set of state transition probabilities $\Lambda = \{a_{ij}\}$,

  $a_{ij} = p\{q_{t+1} = j \mid q_t = i\},\ \forall i, j \quad 1 \le i, j \le N$ Where q_t denotes the current state.

  Transition probabilities should satisfy the normal stochastic constraints,

  $a_{ij} \ge 0,\ 1 \le i, j \le N$ and $\sum_{j=1}^{N} a_{ij} = 1,\ \forall i,\ 1 \le i \le N$

- A probability distribution of each of the states $B=\{b_j(k)\}$

  $b_j(k) = p\{o_t = v_k \mid q_t = j\}\ \forall j,\ 1 \le j \le N\ \forall k,\ 1 \le k \le M$

  Where v_k denotes the k^{th} observation symbol in the alphabet, and o_t the current parameter vector. Following stochastic constraints must be satisfied:

  $b_j(k) \ge 0,\ 1 \le j \le N$ and $\sum_{j=1}^{N} b_j(k) = 1,\ \forall j,\ 1 \le j \le M$

Traditional HMM-based gesture recognition systems require a large number of parameters to be trained in order to give satisfying recognition results. An *n* state HMM requires n^2 parameters to be trained for the transition probability matrix, which limits its usability in environments where training data is limited.

K-Nearest-Neighbor (kNN)

K-nearest-Neighbor is a common and simple machine learning algorithm. The main idea is to assign a class to a new sample by finding which class is dominant among its *k* neighbors, i.e. the class that is assigned to the largest set of the *k* closest neighbors. In this case, different distances (e.g., Euclidean) can be used as well. *k* is a parameter and selecting different *k*s can significantly change the result of the classification. A common way of selecting k is cross-validation, where a few samples are kept for testing the accuracy of the learning phase. Since some samples are more significant than others, scaling the samples usually yields better results. One scaling option is to use evolutionary algorithms. [53]

2.9 Cluster Analysis

Cluster analysis [54] divides data into groups (clusters) that are meaningful, useful, or both. If meaningful groups are the goal, then the clusters should capture the natural structure of the data. In some cases, however, cluster analysis is only a useful starting point for other purposes. Whether for understanding of utility, cluster analysis has long played an important role in a wide variety of fields: social sciences, biology, statistics, pattern recognition, information retrieval, machine learning and data mining. Main advantages of cluster analysis are: [55]

- Supervised learning is costly
- Features may change slowly in time. These changes can be tracked by a classifier running in an unsupervised mode and an improved performance can be achieved
- Unsupervised methods can be used to find features that will then be useful for categorization
- Clustering methods are valuable in early stages of investigation, and thereby gain some insight on the nature of the data.

Cluster analysis can take place once all samples are collected, or sequentially update its model and parameters for each new sample that is gathered. The following sections include a description of the common k-means clustering algorithm, and an overview of sequential clustering methods.

K-Means clustering

K-means is an exclusive clustering algorithm (a certain data that belongs to one cluster cannot belong to another cluster). After deciding on the desirable number of classes, a random mean points are determined, and Voronoi regions are constructed – if computationally or theoretically. Samples that lie inside a certain Vonoroi region are classified to this region. On the next iteration, new mean points are determined according to the mean of the samples in each Voronoi region. A variance parameter J is calculated: $J = \sum_{j=1}^{k}\sum_{i=1}^{n}\left\| x_i^{(j)} - c_i \right\|^2$. Once J has converged, final Voronoi regions are determined and new test samples can be classified according to the location of mean points. [56]

The distance $\left\| x_i^{(j)} - c_i \right\|$ is a chosen distance between a point x_i and the cluster center c_i. The chosen distance can be Euclidean, Manhattan/Rectilinear, Mahalanobis (based on correlation) or any other valid distance measure.

Sequential Clustering

In cases where the final number of clusters is unknown, or the designer wishes to change the model with every new sample, sequential clustering techniques are useful. In these algorithms, samples are presented to the algorithm once or more, and the final solution is usually dependent of the order of presentation. These algorithms are mostly helpful in cases where the system is online (i.e. does not stop learning through time) and when the number of samples is large [57].

BSAS is a basic sequential algorithmic scheme [57]. This simple scheme collects samples, and for each sample looks for the closest cluster. If the distance exceeds a threshold (θ), then a new cluster is formed, otherwise it adds the sample to the closest cluster. This method greatly depends on the presentation order of the samples. Another scheme that depends less on the order is TTSAS – Two-Threshold Sequential Algorithmic Scheme [57]. This scheme uses two thresholds: θ_1 and θ_2 ($\theta_1 < \theta_2$). The first threshold, θ_1 is used for determining whether a sample should join a cluster (i.e. when the distance between the sample to the closest cluster is smaller than θ_1). The second threshold, θ_2, is the smallest distance between a sample and a cluster for creating a new cluster. All samples that their distance to the closest cluster lies between θ_1 and θ_2 are kept aside for the next iteration, where clusters might change and are more rigid. An example of the use of TTSAS (called TTSC in this study) is brought in [58] where sequential clustering is used for background reconstruction and subtraction in images.

Since the TTSAS does not have a limit on the number of clusters, a merging procedure of clusters can be applied after clustering. This procedure iteratively merges clusters that are close to each other. One can also determine the number of final clusters by altering the threshold of distance according to which clusters are merged.

2.10 Distance Metric Learning and Dimensionality Reduction

Prior to classification or clustering, one has to decide on the similarity/dissimilarity metric to be used. It is also possible to use a distance metric method that will find the metric by learning from samples. Moreover, the distance metric methods usually inherently include the possibility of dimensionality reduction since the new features are decreasing in their importance, and it is possible to select a sub set of the most important features.
In this section different distance metrics and two methods for distance metric learning and dimensionality reduction are described: Point Distribution Model [39] and Neighborhood Components Analysis [59].

2.10.1 Distance/Similarity metrics

Most classification algorithms will calculate a distance of a sample from different classes in the feature space (e.g. K-means, kNN). When talking about distance, one must decide which distance metric to use. The most common distance metric is the Euclidean distance. In this metric, the distance is the length of the straight line between two points. In a two-dimensional space it equals $D=\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$. In cases where there is a constraint of moving in lines with right angles between them, we would use the Manhattan distance or Rectilinear distance, where $D=|x_1-x_2|+|y_1-y_2|$.

When using these distances for classification, we often wish to use statistical properties of the sets, in order to find the distance between a new sample and a certain set. If the statistical property in question is the covariance or correlation between two samples, we could use the Mahalanobis distance, where $D=\sqrt{(x_1-\mu)^T\Sigma^{-1}(x_1-\mu)}$, x is the sample and μ is the set's mean value. This distance reduces to the Euclidean distance if there is no variance or covariance in the set ($\Sigma=I$). Other distances such as the Chebychev distance only use the dimension that has the biggest distance in it for evaluation [60].

2.10.2 Dimensionality reduction

In order to avoid the 'curse of dimensionality' and since some features may be irrelevant and incorporate noise, it is desirable to reduce the number of features. Dimensionality reduction can be achieved by selecting a subset of features, or by extracting a new set of features from a the set of initial features that were measured. By finding a projection of the feature space to a

new space, it is possible to reduce the dimensionality of the problem. This reduction is desirable since the solution space of a problem grows exponentially with the number of features, and by reducing the number of features the classification process is simplified. Moreover, many features can incorporate noise or correlation with other features, and therefore avoiding the use of these features allows a more robust classification. There are both linear and non-linear dimensionality reduction methods. In this research, two linear methods that are used both for distance metric learning (see the next section) and dimensionality reduction: Principal Components Analysis (PCA) and Neighborhood Components Analysis (NCA) will be discussed and evaluated. Linear methods were selected since their simplicity is desirable when there is no a priori information regarding expected feature values and provide less of a computational load desirable for real time use.

2.10.3 Distance metric learning

Metric learning aims to learn an appropriate distance/similarity function for a given problem, such as a machine learning problem. Prior to constructing a classification or clustering problem, one has to decide on the similarity/dissimilarity metric to be used. Together with the distance metrics described in the previous section, it is possible to learn a distance metric. For example, Bar-Hillel et al. [61] proposes to learn a Mahalanobis distance using equivalence constraints, as a method of unsupervised learning. A projection matrix is usually used for transforming the feature space to fit the new metric. These transformation are usually linear (e.g., in Relevant Components Analysis [61] and Neighbourhood Components Analysis [59]) but could also be non-linear. Principal Components Analysis (PCA) to be described in the next section can also be used as a form of distance metric learning.

Principal Components Analysis

Each sample contains features that we are interested in. These features are grouped together in a vector called "feature vector". Since there is variance between two movements of the same type (e.g., the same person performing a "Hello" gesture twice, each time on a different day), we would like to use the features that explain the gesture the most.

Principle Components Analysis (PCA) is a method that has two main objectives: reducing dimensionality and better interpretation of the data. Assuming that some of the variables contain more unexplained variation than explained variation, we would like to build a new model that will give the heaviest weight to the feature that best explains the model (in

terms of variance) and minimum weight to the feature that does not explain the model. This goal is achieved by transforming all explanatory variables to a new set of variables, called the principal components (PCs). These new variables are uncorrelated, and are ordered so that the first few retain most of the variation present in all of the original variables. Mathematically, the analysis is performed in the following way:

From k original variables: $x_1, x_2, \ldots, x_k$:

Produce k new variables: $y_1, y_2, \ldots, y_k$:

$$y_1 = a_{11}x_1 + a_{12}x_2 + \ldots + a_{1k}x_k = \mathbf{a}_1^{\mathrm{T}}\mathbf{x}$$

$$y_2 = a_{21}x_1 + a_{22}x_2 + \ldots + a_{2k}x_k = \mathbf{a}_2^{\mathrm{T}}\mathbf{x}$$

$$\ldots$$

$$y_k = a_{k1}x_1 + a_{k2}x_2 + \ldots + a_{kk}x_k = \mathbf{a}_k^{\mathrm{T}}\mathbf{x}$$

Where $\mathbf{a}_i^{\mathrm{T}}\mathbf{a}_i = 1 \ \forall \ i$

such that:

y_k's are uncorrelated (orthogonal)

y_1 explains as much as possible of original variance in data set

y_2 explains as much as possible of remaining variance

y_k explain as much as possible of the remaining variance etc. [62]

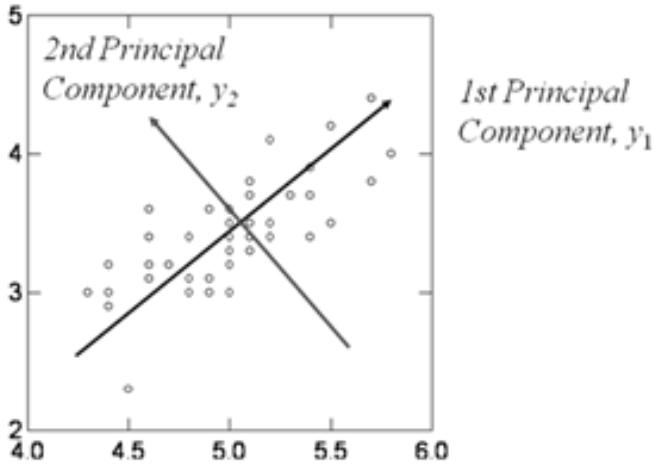

Figure 10 - PCA with two dimensions
[62]

The model can be represented graphically, as showing in figure 10. After transforming the problem into a new one, one should choose the number of principal components that has sufficient variance used to explain the model. A common method is to choose the largest

eigenvalues of the covariance matrix (Σ). The matching eigenvector of the *i*-th eigenvalue is the *i*-th row in the Σ matrix. The model involves optimization using Lagrange multipliers.

Point Distribution Model

The point distribution model (PDM) describes a certain shape by addressing its geometrical properties and the variance of a set of such shapes. In medical applications for example, the shape of organs can vary considerably between individuals and through time [39]. This model allows performance of pattern recognition on objects that are not rigid. The model receives a set of points over the shape (called landmarks) on each sample in a training set, and builds a new set of shapes that are the principal components of the training set, using Principal Components Analysis (PCA).

Several methods were developed for preprocessing – preparing the sample set for the PDM phase. These methods include "Hand Crafted" Models, such as the one made by [63]. In this method, the pattern is first approximated and refined by changing different parts of the model, one at a time. Active Contour Models made in [64] use spline curves modeled to the original shape. The deformation of the object used for constructing the new set is done by changing the location of control points, which are the parameters of the spline curves.

Active Shape Models allow a simple and effective scheme for aligning the training set, and applying a PDM on it.

The statistical properties and assumptions in the PDM method

Every landmark has its variance inside the training set. We would like to model this variance, and to build a set of shapes that are similar to the training set but has less dimensions. This is the essence of the Point Distribution Model technique. Instead of using a big training set, we could use the Principal Components Analysis technique and to pull out most of the variance in the training set with less samples. In figure 11, Cootes et al. [39] bring an example of the alignment phase and the PDM phase.

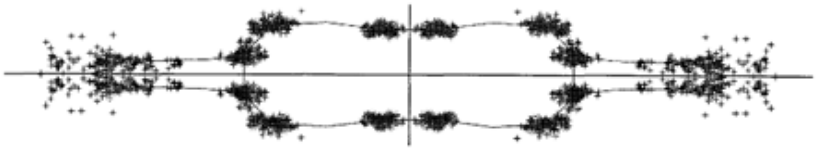

Figure 11 - Landmark distribution on an electrical resistor

[39]

This is the shape of an electrical resistor, after finding the landmarks. The cloud around each landmark is its distribution in the training set. It is obvious that some of the landmarks are more distributed than others. Since the points on each shape belong to a rigid object, it is clear is that there is a partial correlation between points in each sample, and the location of one point affects the location of another. In a Euclidean space, we can treat this distribution as an ellipsoid. Each sample can pull the distribution in its direction, but the final shape of the distribution will be an ellipsoid. This is a statistical assumption important for the next parts of performing the statistical analysis. Another assumption used is the linearity of the coefficients in the principal components analysis [65]. If every principal component is not a linear combination of the initial landmarks, we could use other methods such as non-linear PCA [66].

Classification using PDM

Once principal shapes (components) are found, an identification of a new sample can be done in several different ways. Normally we would use the b vector, the deformable object's set of parameters [67] as a feature vector, and then use a common classification method in order to classify a new sample according to this feature vector. Like any other sample in the training set, the new sample might need an alignment phase. Therefore an alignment of the sample to the mean shape of each class is performed prior to classification.

PDM based pattern recognition systems use in many cases the Mahalanobis distance metric [68], where hand gesture classification was made; kNN algorithm such as in [69] where a new segmentation method using optimal features is proposed; multi-layer perceptron in [66] or different common classification methods, such as discriminant analysis where [70] proposes a method of image search using Active Shape Models and Gray Level Information.

Neighborhood Components Analysis

Neighborhood components analysis [59] is a distance learning method based on the nearest neighbor classifier. It linearly projects the feature space in a way that optimizes a stochastic nearest neighbor criterion using leave-one-out. In other words, NCA looks for a projection where the nearest neighbor rule performs well. The Mahalanobis distance between samples x_1 and x_2 appears in equation 10.

$$d(x_1, x_2) = (x_1 - x_2)^T A^T A (x_1 - x_2) = \\ = (Ax_1 - Ax_2)^T (Ax_1 - Ax_2)$$

Equation 10 - A Mahalanobis distance with a symmetric matrix A

By finding an optimal projection *A*, we maximize the probability of a sample to be classified to its true class. The probability for sample *i* to belong to the same class as sample *j* is calculated using a softmax activation function over the Euclidean distance, as demonstrated in equation 11.

$$p_{ij} = \frac{\exp(-\|Ax_i - Ax_j\|^2)}{\sum_{k \neq i} \exp(-\|Ax_i - Ax_k\|^2)}, \quad p_{ii} = 0$$

Equation 11 - A softmax probability of sample *i* to be classified to the same class of sample *j*

The probability of sample *i* to be correctly classified is given in equation 12, where C_i is the *i*th class.

$$p_i = \sum_{j \in C_i} p_{ij}$$

Equation 12 - The probability of a sample *i* to be correctly classified

The objective function in equation 13 is the sum of the probabilities in equation 12 for all training samples.

$$f(A) = \sum_i p_i$$

Equation 13 - the NCA objective function

Maximization of the objective function would result in an optimal projection of the feature space. The gradient rule can be seen in equation 14. Note that $x_{ik} = (x_i - x_k)$.

$$\frac{\partial f}{\partial A} = 2A\sum_i (p_i \sum_k p_{ik} x_{ik} x_{ik}^T - \sum_{j \in C_i} p_{ij} x_{ij} x_{ij}^T)$$

Equation 14 - the gradient of the objective function of NCA

The probability function *f(A)* is not convex, resulting in a possible sub-optimal projection. Due to this fact, the time complexity of finding a proper solution could be non-polynomial and hence limits the practicality of the signature system that is trained online by the users. In such a scenario, the optimization process cannot last more than a few seconds and is therefore impractical for online interactive systems. For offline systems, however, one can use different optimization techniques for obtaining an optimal projection matrix using NCA. It is also possible to compute different solutions using NCA and compare them using a set of calibration samples.

A modification of NCA, the regularized NCA, requires additional *a priori* data but on average yields better results. An application of the regularized objective function is described in a work by Singh-Miller et al. [11], where NCA was used as a dimensionality reduction method in a speech recognition system. The regularized version of NCA was further studied in by Yang et al. [12], where it was found to be more robust than the ordinary NCA and less prone to over-fitting. This correction is recommended since the assignments of p_{ij} decay rapidly with distance. With the magnitude of *A* growing, the number of neighbors (*k*) for the k-nearest-neighbor optimization may be too small. In the regularized form, λ is a constant chosen empirically. The regularized function is shown in (7). $A_{j,k}$ indicates the element at the j^{th} row and k^{th} column of matrix *A*.

$$f_{reg}(A) = \frac{1}{N}\sum_i p_i - \lambda \sum_{j,k} A_{j,k}^2$$

Equation 15 - The regularized version of NCA

By restricting *A* to be a non-square matrix of size *dXD* where *d* is the desired dimensionality and *D* is the current dimensionality of the feature space we obtain a linear dimensionality reduction of the feature space. This is desirable since the ratio between the number of features and the number of training samples for each user, is extremely high.

CHAPTER 3: PROTOTYPE SYSTEM

3.1 Overview

The methodology of this research involved the development of a working system for user identification, and several new methods that support the system. A prototype system was first created in which three users performed a predefined signature. This system used the geometric characteristics of the predefined signature for better distinctiveness between users. In the final stage, a system capable of gathering any signature was created.

A prototype of the system includes a Point Distribution Model (PDM) for the each user's training set which was used for enrollment, and a Mahalanobis distance based classification used for identification. Prior to applying PDM, the user's signature is recorded and features extracted. The system allows *enrollment* – a construction of a training set gathered by a position sensor, and *identification* – recognizing a new shape, by comparing it to the classes of previously enrolled users.

3.2 Prototype System Method

The prototype system was designed to identify a user performing a predefined signature- the 'X' shape. The X samples that were recorded were divided into 5 segments, and each segment was fitted using piecewise cubic Hermite interpolation polynomial (PCHIP) with equidistant points. The points' coordinates along the new curve were used as features. Following the curve fitting, samples were aligned using Active Shape Models and a Point Distribution Model was constructed for each user. Finally, Mahalanobis distance was used to classify test samples (see figure 12 for a flowchart of the system). The left hand side of figure 12 describes the enrollment procedure, and the right hand side describes the user identification procedure.

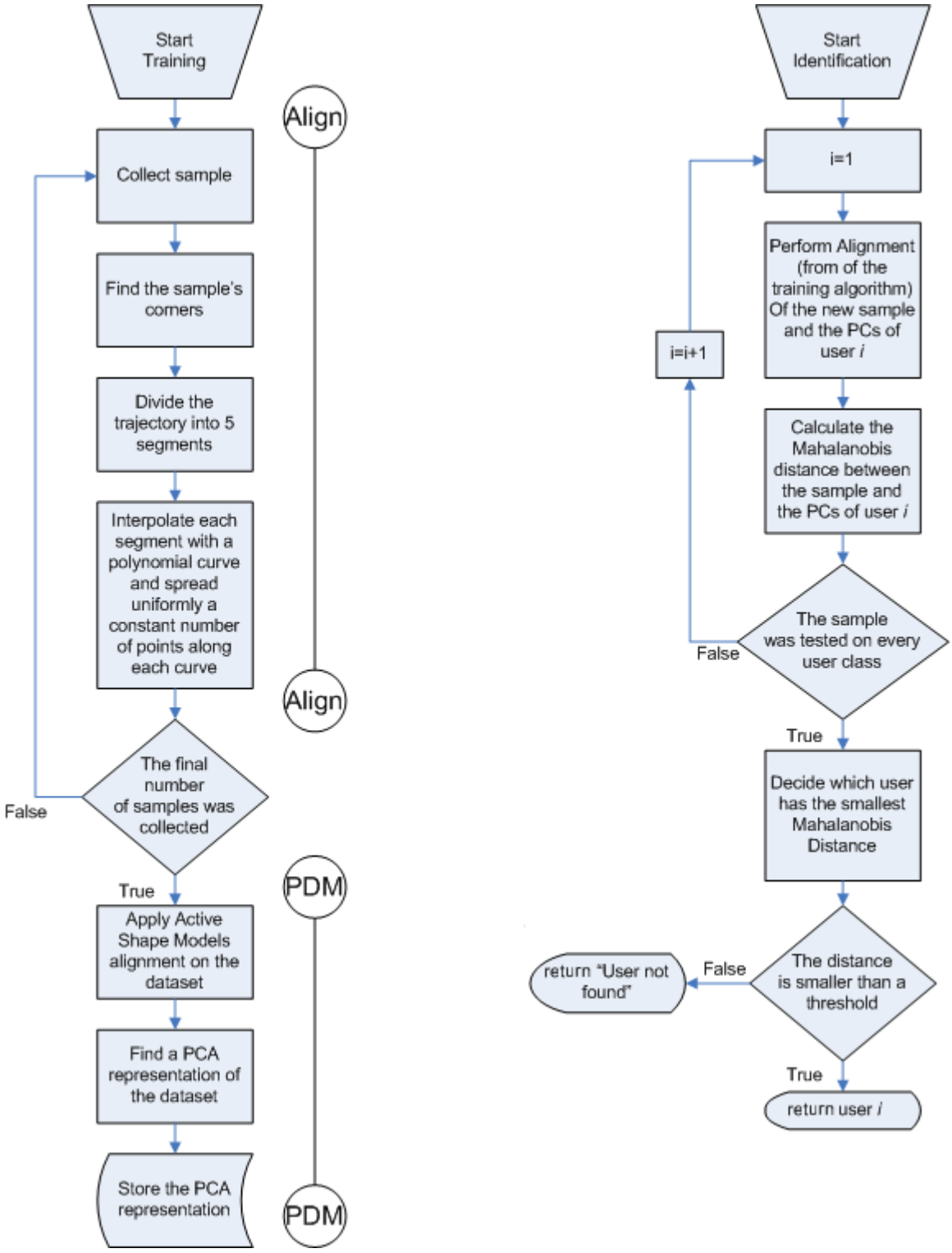

Figure 12 - A general flowchart for the enrollment and identification parts.
Left- Training (Enrollment), Right- Classification (Identification)

3.2.1 System training (Enrollment)

Active Shape Models

Active shape models (ASM) [39] is a common method used in pattern recognition. This method is suitable for this research since it does not assume a rigidness of the pattern. Hand gesture trajectories have a large within-subjects variance, and the alignment (preprocessing) phase of this method allows reducing this variance and performing pattern recognition on the aligned sample set.

Preparation of the trajectories for the ASM method

In order to use the ASM method, the trajectory needs to have certain "landmarks" on it. These landmarks are points that represent the location of specific points on the pattern created by the trajectory. ASM aligns the shape according to each landmark, and therefore the landmarks should be placed accurately and in important spots on the pattern. In the analysis of the "X" signature, it has been decided to place landmarks throughout the entire trajectory, and focus on the curves at the corners of the shape. Since we wish to compare two trajectories, we must obtain the same number of landmarks on each trajectory, regardless of its initial number of frames captured. Therefore landmarks were distributed evenly into the 5 segments shown in figure 13C.

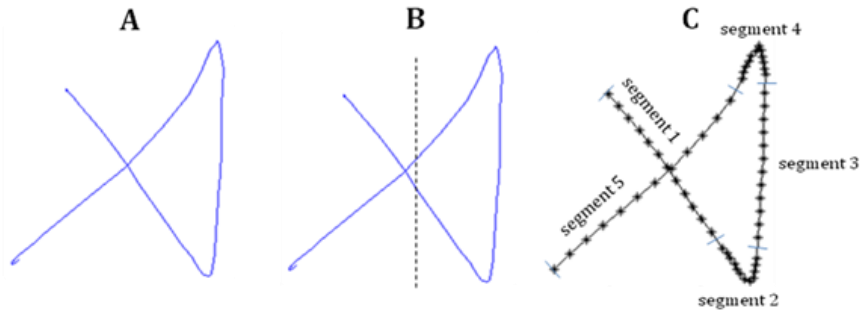

Figure 13 – Division into segments in the prototype system

A: the input trajectory; B: the boundary for finding the corners; C: a synthetic gesture with 5 interpoated segments

Segments selection is the following: Place a segment around each corner inside the X trajectory (the upper right and lower right corners of the trajectory in figure 13C); Then place a segment between the two segments constructed around each corner; Place a segment starting from the beginning of the trajectory; and finally place a segment that starts from the last point on the trajectory.

Finding the Corners

In order to divide the gesture into segments, the initial corners landmarks needs to be found. A preliminary step is to rotate all trajectories to fit to the direction illustrated in figure 13, since the "X" gesture can be performed in 8 different ways (e.g., starting from the top left, then bottom right, then bottom left and then top right).

For finding the corners, a boundary between the left hand side of the gesture and the right hand side of the gesture was formed (see figure 13B). The boundary was placed according to the following rule shown in equation 16:

$$X_{Border} = \frac{\bar{x} + 0.5(x_{\max} - x_{\min})}{2}$$

Equation 16 - Placing the border in the middle of the X trajectory

The points with a maximum *y* value and minimum *y* value on the right side of the border are the top right corner and bottom right corner, respectively. The points with a maximum *y* value and minimum *y* value on the left side of the border are the top left corner and bottom left corner, respectively.

Interpolation

After finding the corners, a segment is constructed around each of the right hand side corners (segments 2 and 4 in figure 13C). *k* points from each side of the corner are selected and the 2*k*+1 points are considered the segment around that corner. The following 3 segments (segments 1, 3, and 5 in figure 13C) are the remaining parts that do not belong to a corner's segment.

On each segment, a piecewise cubic Hermite interpolation polynomial is fitted, and points are placed on it with equal distances between each point. The interpolation is done using the *'pchip'* and '*interparc'* methods in Matlab. *'Pchip*' creates the polynomial and *'interparc'* uses an ODE solver to integrate the distance along the curve itself and this way to uniformly distribute points along the curve.

Validation of the result

The input trajectory and the output of the interpolation phase are shown on the screen, for user validation. If the gesture was not performed as needed, or the interpolation was inaccurate, the user can dispose of this sample and enter a new one. This is done by the user screen showing in figure 15. When using a vision based tracker, the output might contain points that do not belong to a trajectory due to a false classification of a moving part or a skin

colored pixel (e.g. the tracker jumps to another person or to the face of the user). In that case the shape of the trajectory that is received is completely different, but this can be fixed since the Euclidean distance to these points is relatively larger than the distance between points on the trajectory. For that purpose, the screen showing in figure 15 also allows the user to decide if to remove any points, by changing a threshold for distance between points. If a point is more distant than $k \cdot \sigma_{Distance}$ from its adjacent points, it will be removed and the two adjacent points will be connected to each other. The slider on the bottom of the screen changes the values of the parameter *k*.

Alignment

Once samples are interpolated and contain a constant number of points it is possible to find a mean point for each point set and therefore a mean shape, built from all mean points. As a preprocessing phase, we wish to align the training set in order to minimize any unwanted variance within the set that is caused by difference in translation, rotation and scaling of the samples, prior to applying a PDM. The alignment phase is iterative, where each shape is aligned to the mean shape. The next step is to calculate a new mean shape, and re-align all shapes again until convergence.

Point Distribution Model

After aligning the set of shapes, a representative vector consists of points, $\vec{x} = (x_1, y_1, x_2, y_2, \ldots x_{60}, y_{60})$ is created for each shape and a matrix $X = (\vec{x}_1, \vec{x}_2, \ldots \vec{x}_N)$ is built for the N samples of the user. The mean of *X*, $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ and the covariance of *X*, $S = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})(x_i - \bar{x})^T$ are also calculated for the use of the PCA. Following this calculation, the eigenvector matrix *P* and the eigenvalues λ are found using a Principal Components Analysis on the covariance matrix *S.* After retrieving theses values, the number of principal components representing 98% of the variance was chosen. Since different users require different number of principal components in order to fulfill this constraint, the maximum number of PCs needed for all users to represent 98% of the variance was taken for all users, i.e. 10 PCs.

According to [70], shape parameters can be assigned using the *b* matrix, where $b = P^T(X^T - \bar{X})$, columns in P represent eigenvectors of S sorted by importance, X^T is the matrix of shapes, and each column in $\bar{X}$ is the average over rows of *X* (columns in $\bar{X}$ are equal, since we wish to find $\Delta = x_i - \bar{x}$ for each *i*)

See equation 17 for an example calculation of the shape matrix b.

$$\mathbf{P^t \times (X^t - \bar{X})}$$

$$\begin{pmatrix} a_1 & a_2 & a_3 & a_4 \\ b_1 & b_2 & b_3 & b_4 \\ c_1 & c_2 & c_3 & c_4 \\ d_1 & d_2 & d_3 & d_4 \end{pmatrix}^t \times \left(\begin{pmatrix} x_{11} & y_{11} & x_{12} & y_{12} \\ x_{21} & y_{21} & x_{22} & y_{22} \\ x_{31} & y_{31} & x_{32} & y_{32} \end{pmatrix}^t - \begin{pmatrix} \bar{x}_{j1} & \bar{x}_{j1} & \bar{x}_{j1} \\ \bar{y}_{j1} & \bar{y}_{j1} & \bar{y}_{j1} \\ \bar{x}_{j2} & \bar{x}_{j2} & \bar{x}_{j2} \\ \bar{y}_{j2} & \bar{y}_{j2} & \bar{y}_{j2} \end{pmatrix} \right)$$

Equation 17 – an example calculation of the shape parameters, *b*

a_j - the coefficient of the first *x* coordinate of the first landmark. *j* is the principal component.

b_j - the coefficient of the first *y* coordinate of the first landmark.

x_{ij} - the x value of the *j*-th point's *i*-th sample.

$\bar{x}_{j1}$ - the mean of all samples of x_{i1} where j=1...n

The equation $b = P^T(X^T - \bar{X})$ multiplies the coefficient vector (a,b,c,d in the example) with $\Delta x = (x - \bar{x})$. The result is one mode of variation, a linear combination of the deviation (Δx) of each point. The b value is mainly used for classification, but PDM also allows us to generate new shapes with the properties of the training set. Construction of the principal component shape is done by $PDM = \bar{X} + Pb$. By using $X_{new} = \bar{x} + k\sqrt{\lambda_i}P_i$. By changing *k*- the number of standard deviations, and *i*, the index of each eigenvector, we can generate an unlimited number of new shapes.

3.2.2 Classification

In order to recognize which user performed the gesture, there is a need to compare the new signature to each class. Then, a value for each class is determined and the new sample is classified to the class which is most likely to include that signature. Prior to this comparison, the sample passes a pre-processing phase of finding the corners, interpolation and alignment described in section 3.2.1. A threshold test is conducted in order to allow user verification.

The classification algorithm is the following:

Procedure user = Classify()
Begin
For each user class *j*:
Align the test sample to the mean shape of class *j*.
Calculate $b_j = P_j^T (x_{new} - \bar{x}_j)$
Find the covariance matrix of b_j, Σ_{bj}
Calculate the Mahalanobis distance $D_j = (b-\bar{b})\Sigma_b^{-1}(b-\bar{b})^T$
If $D_j = \min_k \{D_k\}$ and $D_i - D_j \geq Threshold, \forall i \neq j, i \in N$, assign the sample to class *j*.
End

3.2.3 Graphical User Interface (GUI)

The system's graphical user interface was developed in Matlab. It has two main windows: the entrance window, which allows the user to select the Enrollment part where one can add samples to a training set, and the identification window where a user can classify a new gesture to one of the user classes in the database. The GUI can be seen in figure 14. The second window is the validation window seen in figure 15, where a user can validate that a gesture was collected correctly by the system.

Figure 14 - the system's graphical user interface
The system allowes both identification of sample sets and adding samples to the training set.

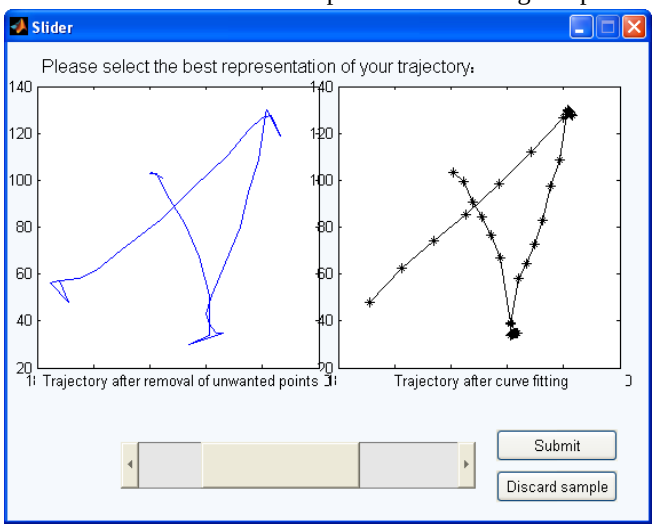

Figure 15 - The validation window
Allows the user to see the input gesture, the interpolated gesture and decide on the number of distance-between-points standard variation, for removing unwanted points.

3.2.4 Prototype Sample Collection Program

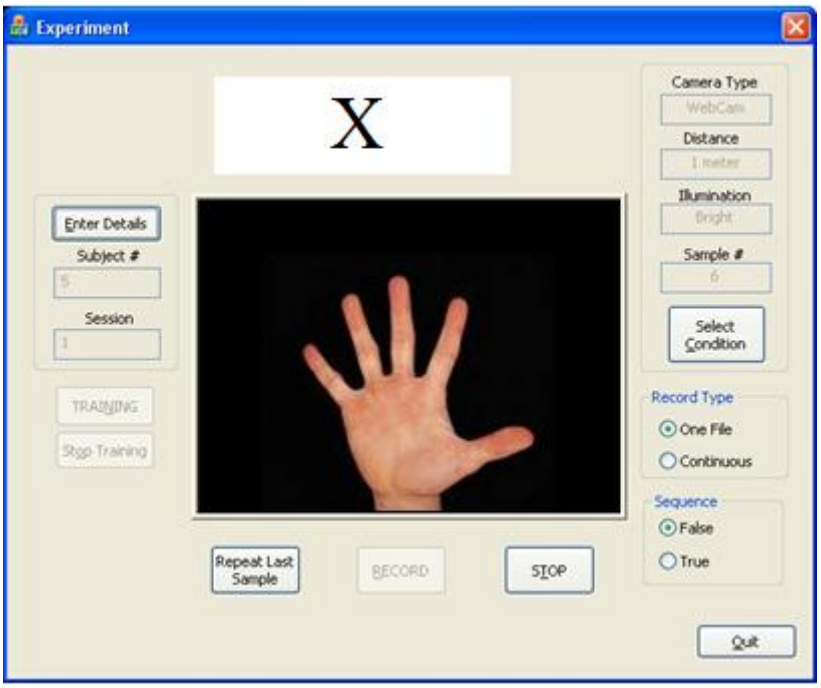

Figure 16 - The sample collection application GUI

An Application for automatically recording hand signatures was developed for the use of this research. The program was built in C++ and MFC (See figure 16 for the graphical user interface). The application automatically samples different gestures and saves them on file. File names contain all information needed from the sample: the subject number, session

number, distance, camera type, light and the type of gesture recorded. The program has a training option that lets users practice their gesture without having it recorded.

3.3 Preliminary Experiment

The tracking in the prototype system was made using a Polhemus 3Space FastTrak position sensor that plots 3D locations of the hand. The outputs of this sensor were collected and entered as text files to the system. Using the position sensor, 25 'X' gestures were collected from 3 users: ages 24, 27, 38; two left handed and one right handed; two males and one female. 20 gestures were used for a training set, and the rest for a validation set. Since some of the gestures were performed incorrectly, the validation set eventually was smaller, and contained 7 gestures: 2 of user 1, 2 of user 2 and 3 of user 3. These samples were kept aside for accuracy testing.

3.4 Results for the Preliminary Experiment

Table 1 shows results for the 7 test samples classification, using the current methods.

Test #	User	Output	Mahalanobis Distance			Difference in distance between best and 2nd best	Ratio between best and 2nd best
			D_1	D_2	D_3		
1	1	1	19.95	67.96	90.42	48.01	0.29
2	1	1	17.48	216.20	33.59	16.11	0.52
3	2	2	51.05	2.92	74.95	43.42	0.06
4	2	2	59.79	7.62	76.89	52.16	0.13
5	3	3	102.77	226.78	14.63	88.13	0.14
6	3	3	81.58	154.63	14.14	67.44	0.17
7	3	3	105.75	154.61	14.42	91.33	0.14

Table 1 - The 7 test samples and their distances from each set.

The difference in distance column shows the difference in the distance between the best fit and the 2nd best fit. The ratio distance shows the ratio between the best fit class and the 2nd best. These values can be used for a threshold for non-classified.

Table 1 shows that all 7 samples (100%) were classified correctly. We can also see that the difference between the best match and the 2nd best match is relatively large. Therefore it is possible to decide on a threshold for the difference in the distance. For example, if the difference in distance would have been less than 10, the gesture would have been non-

classified. It is also possible to use a threshold on the ratio between the first and second best. Here, we would use a rule that if the ratio is bigger than 0.75, for example, the gesture would be classified as non-classified.

Figure 17 shows the training sets of the 3 users after applying PDM with 10 principal components. Figure 18 shows the 10 principal components of user 2.

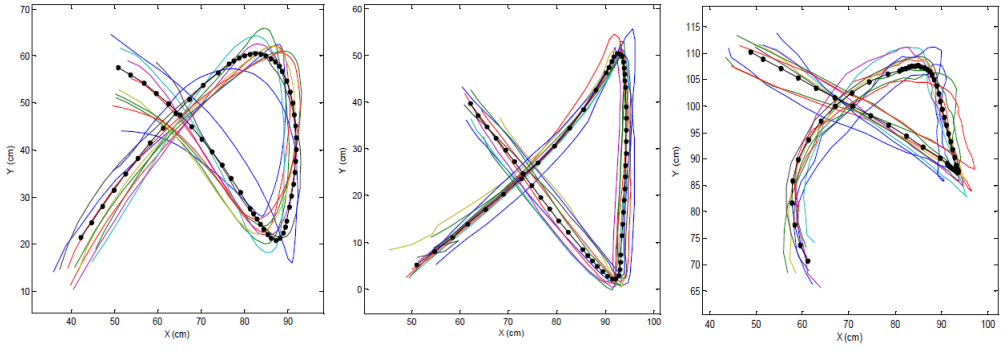

Figure 17 - 10 PCs and their mean for 3 users

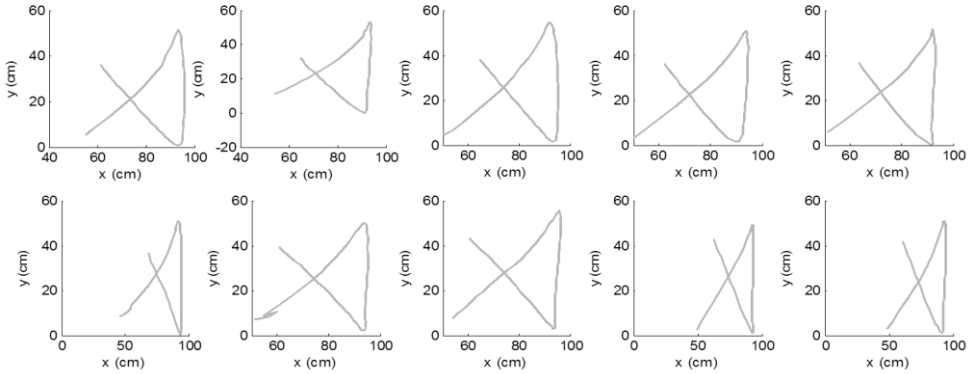

Figure 18 - the 10 PCs of user 2
(top left - PC 1, top right – Pc 5, bottom right – PC 10)

3.5 Discussion

The prototype system and the preliminary experiment reflect the ability to differentiate between a small number of users performing the same motion signature. It can be seen in figure 17 that although all participants were asked to perform the same shape, and they all performed it with similar geometric characteristics (i.e. started from the same point and had the same location for the connecting line between the diagonals of the X), the trajectories are different.

Since the shape was known to the system, an extraction of landmarks was performed accordingly, by finding the corners of the X and placing a segment around each corner and on the straight lines of the shape. This knowledge allows the user to have greater accuracy rates, but also limits the possibility of identifying an unknown signature. The challenge of the final system is to keep similar identification rates even when the shape is unknown to the system once a user starts to enroll.

CHAPTER 4: METHOD

4.1 Overview of the Method

The biometric user identification system is comprised of an enrollment/training stage and an identification (run-time) stage where a user defines a signature gesture and enrolls to the system. In the run-time (i.e. identification) stage the user performs the signature when prompted by the system and the system returns the identity of the user. An enrollment system was developed which allows each user to select his or her own signature. This system interactively ensures that the user's signature is consistent and that it is not similar to the signatures of other users already enrolled in the system.

A flow chart of the user identification system is presented in figure 19. The left hand side describes an enrollment stage, and the right hand side describes the classification stage. The initial data collection operation involves hand tracking, trajectory extraction and alignment (performed for every recorded signature).

For user identification the system collects data from a 3D sensor and extracts the 3D location of the user’s hand for every frame. The resultant trajectory is the input to the user identification system. After the signature is performed by the user, features are extracted from the captured trajectory, and the extracted features are compared to features in a database of enrolled users. A distance threshold is used for user authentication. The signature is pre-defined by the user during the enrollment phase, allowing personalization for better intuitiveness and comfort.

During enrollment, for each sample, the system builds a fitted curve that is based on the initial trajectory. Following fitting, the system aligns all signatures to a common ground, since the signature can be performed in different locations in front of the camera. Features collected are the x and y coordinates of points along the curve, and additional velocity and curvature values for segments along the trajectory. An online learning method that uses sequential clustering finds different forms of the user's signature for better classification and for interactive training – the system can advise the user whether his/her signatures are too noisy, or whether there is a similarity to a signature stored by a different user. Two methods were evaluated for dimensionality reduction and for learning a distance metric: Neighborhood Components Analysis [59] and Point Distribution Model [39].

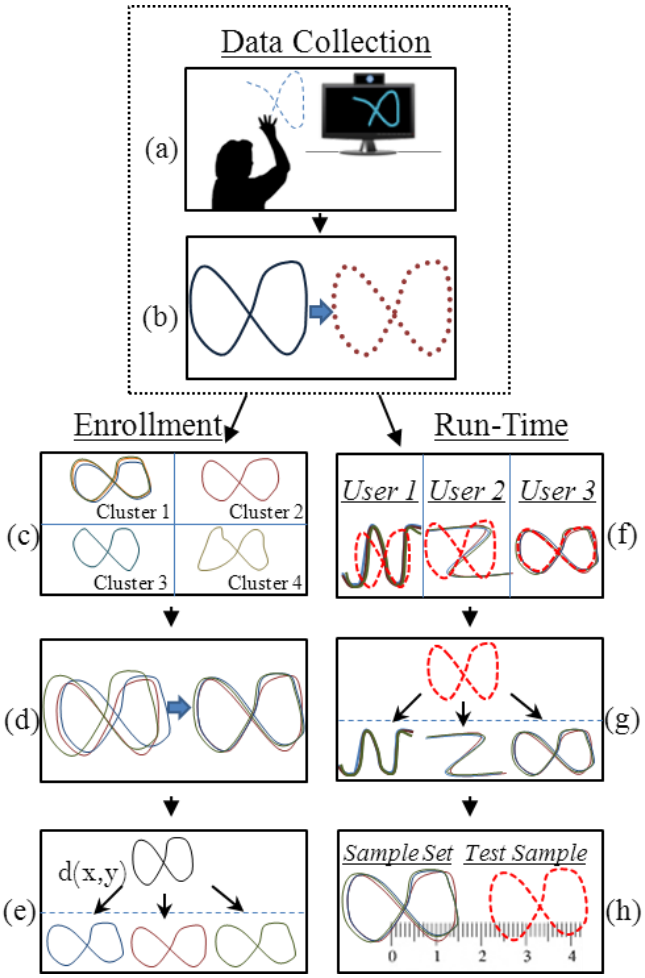

Figure 19 - Illustration of the final system.

- Data collection (top): The hand is tracked and a trajectory is preprocessed (a) and aligned (b).
- Enrollment sample (left): The sample is clustered (c) to find different forms of movement in each of the user's signatures. Each cluster is aligned (d) and a distance metric learning phase (e) takes place using samples from all users.
- Test sample for classification (right): The sample is first aligned to the clusters of all users (f), the distance to every cluster is calculated (g) and if the distance does not exceed a threshold (h), the sample is classified as belonging to the user with the closest cluster. Otherwise, it is not classified.

4.2 Data Collection – Trajectory Preprocessing and Alignment

The features of a trajectory are points (x and y coordinates) along the fitted curve, and average velocity and curvature values of fixed length segments of the initial trajectory. These features reflect both spatial information such as the signature's shape, and temporal information such as velocity, since the shape itself might not be sufficient for user verification. Preprocessing of the signature trajectory is necessary for providing a consistent representation, and therefore each trajectory is re-sampled uniformly with equidistant spacing using a cubic Hermite interpolation for ensuring a constant number of features per signature. Prior to re-sampling, the hand trajectory is segmented based on the distance of each point from the start and end points, and initial and final movement segments are disregarded as they were found to be too noisy. A diameter d of the bounding circle of the entire trajectory is determined, and points lying within a distance of $0.1d$ from the first or last points of the trajectory are erased. See figure 20 for an illustration of the deletion process.

For additional reduction of variability caused by the location and distance of the user in front of the camera, all samples were translated and scaled to the mean shape using Procrustes analysis [38], which is a straightforward alignment method. Another alignment method tested is Active Shape Models (ASM) [39], where a mean shape is calculated, and all samples are translated, scaled and rotated in order to minimize the total Euclidean distance between the sample and the mean shape. The process is iterated until convergence of the mean shape, and gives more weight to stable points.

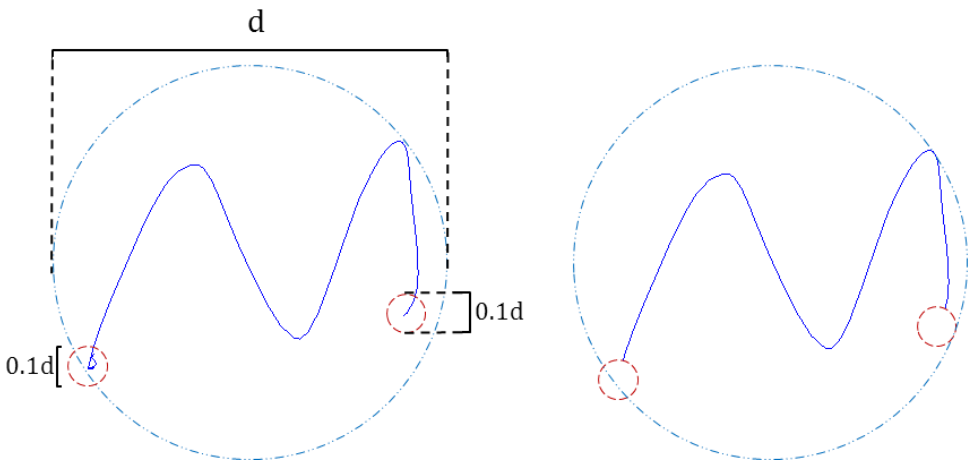

Figure 20 - An illustration of the noise deletion process.
Points within the red circles are to be removed.

4.2.1 Division into segments

When recording a trajectory, we assume that the frame rate is constant and therefore we could use the distance between points as a velocity feature. The existence of many points on a short arc length reflects slow movement, and vice versa. By dividing the initial trajectory to segments prior to interpolation, we could obtain both the shape of the trajectory, and the velocity values on each segment. See figure 21 for the motivation of dividing into segments.

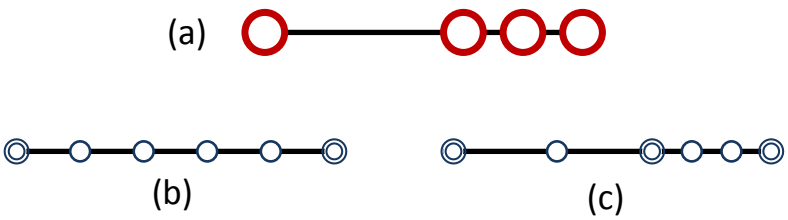

Figure 21 – Illustration of the difference between fitting an entire trajectory and fitting segments of the trajectory.
(a) the points on the initial trajectory. (b) a fitted curve with 6 equidistant points for the entire trajectory. (c) a fitted curve with 6 points for two segments of the trajectory, that best represent the location of the points on the initial trajectory. Double circles represent start and end of segments.

The division into segments is based on points of interest, i.e. points with high curvature values. In this method, points are declared as 'knots' between segments, and segments are built around these points. Points of interest are found by calculating the curvature value of each point and comparing the value to a curvature threshold. Around each point of interest a segment is placed and referred to as 'critical segment'. Additional segments are placed between critical segments already placed for points of interest. If k points are found as points of interest, then a maximum of $2k+1$ segments are placed. See figure 22 for an illustration.

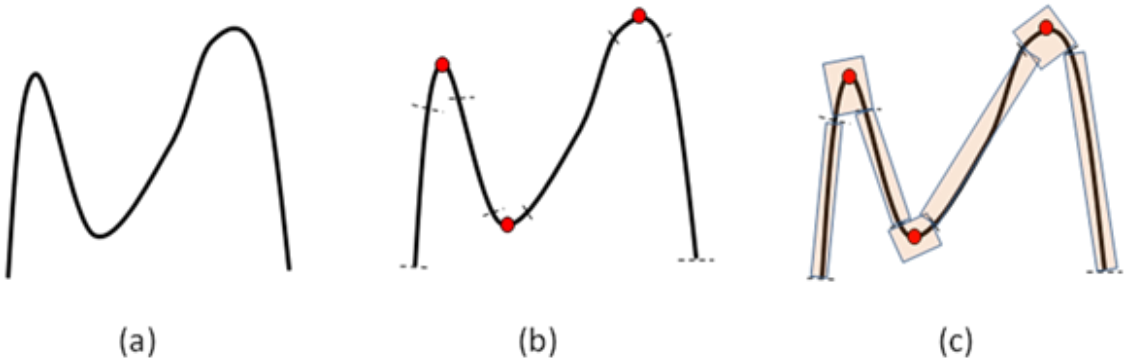

Figure 22 - Illustration of the division into segments based on points of interest.
(a) curvature values are found for each point on the trajectory. (b) points with curvature values higher than a threshold are used as points of interest. (c) segments are constructed around points of interest, between previously defined segments and between the start point to the first segment and the last point to the last segment.

Curvature values are found using the method described in [73] and in equation 18. For each point with an angle of curvature that is higher than a threshold, a segment is created. The number of points higher than a threshold is defined as k. A constant number of points, n, of the entire trajectory, is distributed between the different segments using the following algorithm:

a) Find *L*, the total length of the trajectory. Each segment around a critical point will be of length *L/k*.

b) Place segments around critical points, and segments between critical segments. In case of two segments overlapping, find the middle point between segment centers as the new border between segments. Let *N* be the final number of segments.

c) Find the percentage m_i of points of segment *i* in the initial trajectory, out of the total number of trajectory points.

d) Calculate F_i the expected number of points on segment *i* of the fitted trajectory: $F_i = m_i(n+2k)$. $2k$ additional points are added to handle points on segment borders.

e) Each segment *i* receives $\lfloor F_i \rfloor$ Points. The remaining points $(n - \sum_{i=1}^{N} \lfloor F_i \rfloor)$ are distributed iteratively to segments according to $\min_{1 \le i \le N} \{ \lceil F_i \rceil - F_i \}$ until all *n* points have been distributed.

f) In case of too many (according to an empirically set threshold) segments are identified, e.g. in case of many curved sections, the system iteratively attempts to construct the fitted curve with *k=k-1* points of interest (combining the smallest segment with the smaller adjacent segment).

g) Erase duplicate points at the 'knots' between segments.

$$c = \frac{\left| x' y'' - y' x'' \right|}{\left[(x')^2 + (y')^2 \right]^{3/2}}$$

Equation 18 – Calculations of curvature values.
1st and 2nd derivatives of the *x* and *y* coordinates are found using linear and spline interpolations

4.3 Dimensionality reduction

In order to avoid the 'curse of dimensionality' and since some features may be irrelevant and incorporate noise, it is desirable to reduce the number of features. As no information is a priori available regarding the signatures that will be selected by the users or the differences between users, a distance metric learning method is needed that will be robust for all signatures. Another limitation is that the number of training samples should be kept small to

make the system practical. For dimensionality reduction, two methods were tested: Point Distribution Model (PDM) [39], an unsupervised method based on principle component analysis (PCA), and Neighborhood components Analysis (NCA), [59] , a supervised method. Linear methods were selected since their simplicity is desirable when there is no a *priori* information regarding expected feature values.

PDM is unsupervised and straightforward while NCA is supervised and based on the k-nearest-neighbors classifier that was found useful in similar systems such as in [9], where the k-nearest-neighbors classifier and leave-one-out cross validation, both incorporated in NCA, are used for gait recognition.

Point Distribution Model (PDM)

Point Distribution Model is an unsupervised method in which a model is built independently for each user. Using equation 19, a shape vector b_j is calculated for each sample j of the user's signature based on the eigenvectors found by the PCA analysis. Let f_j be the feature vector of sample j. the matrix P represents eigenvectors of the covariance matrix of the user's samples and $\bar{f}$ is the vector of average values for each feature across samples of the user.

$$b_j = P^T(x_j - \bar{x})$$

Equation 19 - Finding the shape vector in PDM
[70]

The number of PCs (i.e. the number of columns to be used in matrix P) chosen is defined a *priory and is* constant for all users. The shape matrix b can be used for calculating a Mahalanobis distance $D_j = (b_{new} - \bar{b})\Sigma_b(b_{new} - \bar{b})^T$ where $\bar{b}$ is the mean shape of the signature to which the sample is compared, and Σ_b is the covariance matrix of b.

Neighborhood Components Analysis (NCA)

Neighborhood Components Analysis is a Mahalanobis distance learning method. The ordinary and regularized Neighborhood Components Analysis is described in chapter 2.

The method doesn't require any prior knowledge (e.g., a Gaussian distribution) about the training data, and has the advantage of robustness to errors caused by different inputs (e.g., no matrix inversion is needed). For the distance metric learning, a regularized NCA [72] was applied.

4.4 Enrollment / System training

Training samples are gathered sequentially in an interactive enrollment phase. Each user records a small number of samples when enrolling to the system. The system lets the user record each sample for a constant duration. The system also shows the fitted signature in a graphical user interface (GUI) and allows the user to remove a sample in case of an error. The samples are clustered using a sequential clustering algorithm described subsequently. The first set of samples taken in the interactive enrollment is used for distance metric learning using NCA, and additional samples are used for a further optimization of the system. The additional samples are used as a calibration set both for threshold determination and for finding an accurate feature space projection using NCA, and not used for clustering.

In order to avoid suboptimal results of NCA, the projection matrix is recalculated until a proper solution is found. For a predefined number of iterations, the NCA yields a projection matrix and the accuracy is measured. If the accuracy of evaluating the calibration samples is satisfactory, e.g., above 98 percent, the projection matrix is kept for future classification. If not, another iteration of NCA takes place and the accuracy is compared to previous iterations. The best projection, or the first projection that is above a desired accuracy, is selected for future classification.

The system selects a threshold for each user instead of using a predefined threshold. The threshold chosen is determined by comparing each calibration sample to the user's training set and measuring the distance from the sample to the user. The threshold is set as the maximum distance. This threshold can be used for user validation, i.e., samples with a distance larger than a threshold will not be classified to that user.

On line Interactive Enrollment

Within a user's training set, there could be more than one form of movement, i.e., the user might perform his/her signature in more than one way due to fatigue or different variables derived from motor control. In order to make sure the user's sample is classified correctly, the on line interactive enrollment employs a two-threshold sequential clustering [57] method. The first threshold (θ_1) is used for accepting a sample to an existing cluster. The 2nd threshold (θ_2) is used for rejecting a sample from an existing cluster and $\theta_1 < \theta_2$. Samples between θ_1 and θ_2 are kept aside for future decision.

The sequential clustering method performs the following functions:

1. Interacts with the user.

2. Divide the samples into clusters, in case the user performs the signature in different forms
3. Outlier detection

In the clustering algorithm a sample can be assigned to one of the following: (a) an existing cluster, (b) a new cluster, and (c) a temporary storage location. Samples in a temporary storage are retained until all the samples have been processed, at which point they are introduced again into the clustering algorithm. At this point all existing clusters are more stable since clusters are composed of more samples. A flow chart of the algorithm is presented in appendix IV. By checking if any sample was clustered in the previous iteration, we assure the convergence of the process in which all samples will eventually be clustered. i.e., if no sample was clustered in iteration *q*, in iteration *q+1* a stored sample will be clustered to a new cluster. After all samples were introduced to the procedure and the storage is empty, the process ends. A pseudo code for the algorithm is given below:

Procedure [C] = InteractiveEnrollment()
Begin
While not all samples were clustered
  For each sample j
   If this is the first iteration **Then**
    Collect Sample x_j
   Else
    Use a sample x_j already collected in a previous iteration
   If x_j is the 1st sample **AND** x_j was yet to be clustered **AND** no change took place at the last iteration **Then**
    Form a new cluster c_1 and add x_j to c_1
   Else If the sample was yet to be clustered **Then**
    For each cluster $c \in C$ *//where C is the cluster set of all enrolled users*
      Align x_j to the mean shape of *c*
      Calculate the distance between x and the mean of c
    Find $d=Min(distance(x_j,c_{i,k}))$ *//the minimum distance between x_j and c, where c is cluster i of user k.*
    If $d<\theta_1$ **AND** the winner cluster belongs to the enrolling user **Then**
      Add x to the winner cluster
    If $d<\theta_1$ **AND** the winner cluster belongs to a different user **Then**
      Advise the user to select a different signature
    If $d>\theta_2$ **Then**
      Form a new cluster c_{k+1}
      Add x_j to c_{k+1} where k is the current number of clusters
    If $\theta_1<d<\theta_2$ **Then**
      Store x_j in storage
Merge() *// Merge clusters that are closer than θ_3*
End

A limit on the number of clusters per user is added (θ_3), since too many clusters may indicate a high variability in the user's motion. In case of a small number of training samples, even small clusters (i.e. containing one or two samples) are kept and used for classification. When the number of training samples is high, it is possible to remove clusters smaller than a threshold which have a high probability of being outliers, and by that to remove unwanted variability in the training set.

The traditional motivation to use online clustering algorithms is to handle a large amount of data sequentially instead of in one pool. The large amount of data assures near-optimal learning. In this case, the small amount of samples reflects in suboptimal clustering. In order to overcome this weakness, a merging procedure takes place once the user finished enrolling. In this phase, clusters that have a distance smaller than a threshold θ_3 are merged into one cluster. In this stage there is also a limit on the maximum number of clusters per user.

4.5 Run-time / Identification

In order to classify a sample, it is first aligned to each cluster, and the distance to the cluster is calculated. A distance threshold is used for user verification. If the distance does not exceed a threshold, the sample is classified to the closest class/cluster, otherwise it is unclassified. The classification algorithm is given below:

Procedure [result] = Classify()
Begin
  For each user *i*
    For each cluster *j*
      Align the test sample to the mean shape of cluster *j*
      Calculate the Mahalanobis distance:
      $D_j = (b - \bar{b}_j)\Sigma_{b_j}(b - \bar{b}_j)^T$ where $b_j = P^T(F_{new} - \bar{F}_j)^T$ *//for PDM* and:
      $D_j = (x_{new} - \bar{x}_j)^T A^T A (x_{new} - \bar{x}_j)^T$
      // *for NCA.*
  $D = \min(D_j)$
  If $D >$ *Threshold* **Then** classify the sample as "not found".
  Else return the closest user's ID.
End

CHAPTER 5: EXPERIMENTS

5.1 Subjects and Experimental Procedure

Three different experiments were conducted. In all experiments, subjects were industrial engineering students in Ben-Gurion University of the Negev. A different pool of subjects was used for each experiment. The experimental protocols were approved by the Ben-Gurion human subjects research committee. In the first experiment an independent signature system was tested in which all subjects performed the same signature gesture. The experiment was repeated three times, each time using a different signature shape. The shapes were a circle, a line and an X. In the second experiment a dependent system was tested in which each subject picked his/her own signature. In the third experiment, 6 out of the 23 subjects of the 2nd experiment were asked to imitate another participant's signature, in order to test how robust the system is to forgeries.

In all experiments, subjects sat 1.5 meters away from a 3D camera. A recording system recorded each signature for 7 seconds. Prior to recording, there was a countdown of 3 seconds. The subjects were recorded using the PrimeSense 3D camera and hand tracker. The camera captures both RGB and depth data using structured light and the tracker provides 3 coordinates of the hand's location. In our experiment all the subjects choose to perform planner signatures in the frontal plane although no such limitations were imposed. Thus in the current research the frontal plane coordinates (x, y) were used for signature representation and the depth coordinate (z) was disregarded. The subjects were recorded in two different weeks, in order to see the shape difference and system accuracy due to the users' ability to replicate their signatures over time. Since the system's accuracy is influenced by the number of users (i.e. the probability of identifying one user out of N), users were divided into groups of varying sizes, in order to test the effect of group size. The total number of samples collected in each experiment is shown in table 2.

	# of users	Samples			
		Training	Calibration	Test	Imitated signatures
Experiment I	18	18	4	6	0
Experiment II	17	20	5	10	0
Experiment III	6	20	5	5	15

Table 2 - The total number of samples for each experiment

Experiment 1: The following shapes: x , ○ , →. Experiment II: Signatures. Experiment III: Signatures and imitating another user's signature.

5.2 Experiment I: Independent System (Predefined Signatures)

Sample signatures were collected from 18 subjects, age 25±1; 14 females, 4 males; 17 right handed, 1 left handed. Subjects recorded a total of 28 samples each of shapes X, O and 'line' (i.e. where subjects were asked to perform a horizontal line to the right and back to the left). Subjects were instructed to draw the shape in the air, without demonstration from the experimenter in order to avoid imitation. The samples were collected in two sessions: 22 and 6 samples in the first and second sessions, respectively.

Eighteen of the samples of the first session were batched as a training set, and the additional 4 were set aside and used for calibration. Samples were reviewed and deleted in case of tracking error or a severe human error in performing the requested shape. Samples from the training set that were deleted were replaced by one out of the 4 calibration samples. After evaluating the samples, all users had at least 18 samples for a training set and 1 calibration sample.

The performance of the system was measured as the accuracy in identifying the user performing a shape. This experiment had two main objectives: (a) to analyze the capability (accuracy) of identifying a user based on the motion characteristics of a given signature, and (b) to study the effect of shape complexity on user identification. For free hand gesturing X is considered complex to perform, the circle is simpler and the line is the simplest. It was hypothesized that it is more difficult to correctly identify a user as the number of different users performing the same gesture is increased. To study the effect of such a “scale up”, tests were made for different user cohort sizes, starting from size 3 and increasing incrementally up to size 7 (referred to subsequently as “group size”). This was repeated using each of the

shapes independently. Thus, for five group sizes and three shapes a total of a 15 tests were made, where in each test, the accuracy based on a sample of 18 groups was evaluated.

5.3 Experiment II: Dependent System (User Defined Signature)

In this experiment, 23 different subjects: age 26±1.5; 13 females, 10 males; 17 right handed, 6 left handed recorded their own signatures. The subjects were asked to think of a signature that is simple to replicate but difficult to imitate. On the first session, subjects recorded 20 samples as a training set and 5 samples as a calibration set. On a second session a week later, subjects were recorded again performing their selected signature for 10 times. These samples were used as test samples.

5.4 Experiment III: Forgery

Six subjects out of the 23 of experiment 2: age 29±2.4; 3 females, 3 males; 4 right handed, 2 left handed were used to try to imitate (forge) the signatures of other subjects. Each of the 6 performed 3 repetitions of 5 signatures of other subjects, resulting in 90 fake signatures overall. Prior to imitation, the subjects (imitators) observed each of the five users they were to imitate, logging into the system using his/her signature for 5 times. For each participant enrolling to the system, the other 5 subjects sat in the room and watched him perform his signature in front of the camera. Following the user's enrollment, the 5 imitators performed his signature one after the other.

5.5 Parameter Selection and System Evaluation

Samples taken in the experiment were used for: a. the training of each user's model; b. for finding the best system parameters and c. for evaluating the system's accuracy. For the training of each user's model, the number of dimensions was reduced to 7 in both methods. Second session test samples of 10 users out of the 23 in experiment II (signature experiment) were used for finding the parameters of the regularized NCA (λ), the thresholds for the sequential clustering algorithm (θ_1,θ_2,θ_3) and for determining the alignment strategy (i.e. Procrustes analysis or ASM alignment). The second session test samples of the additional 13 users were used for the evaluation of the system. Therefore, for the system evaluation, samples from 13 users were used for evaluating a system with the clustering ability (i.e.

interactive enrollment) and regularized NCA based system. 23 users were used for testing non-regularized NCA and PDM.

In order to test the system's accuracy, subjects were divided into 18 groups of 3 up to 7 users, and the system's identification accuracy based on the 2^{nd} session test samples was measured. For each group size, users were randomly assigned to a group, in a procedure similar to bootstrapping. Using this procedure, a larger number of groups at each size could be tested, each time with a randomly selected group of users. For example, 23 users were randomly divided into 7 groups of size 3, and 2 users were not assigned to any group. The accuracy of each system with 3 users was measured. Following that, the users were randomly reassigned into groups of 3, and the accuracy was measured again. For each group size, we obtained a sample of 18 groups.

Regularized NCA and online learning required optimization that was performed using samples of 10 users. The 13 additional users were divided only into groups of 3 or 4 since the probability of having two similar systems of bigger groups is high (i.e. having two systems with the same enrolled users).

In experiment III, thresholds for distances were introduced. The accuracy measures showing in experiment I and II do not use these thresholds, i.e. there is no class of "Not Found".

CHAPTER 6: RESULTS AND DISCUSSION

6.1 Experiment I: Independent System (Predefined Signatures)

The system's accuracies for the different shapes and different group sizes are shown in figure 23. For every group size, the O shape has the highest identification rates. As expected, the 'line' shape is the least classifiable. The differences between shapes were tested using a repeated measures ANOVA test and are significant (p-value=0).

We assume that the 'line' shape is too simple for classification, and the 'X' shape is too complex for repeatedly performing identical shapes. The 'O' shape is the tradeoff between the simplicity of performance and repeatability. In figure 24, three different 'X' shapes, circles and 'line' shapes are shown. The shape itself is in many cases different than the shape the participant was asked to perform, as can be seen for the 'line' shape, which is in most cases not straight. The correct identification accuracy of a single user among many users decreases at a rate of approximately 0.03 for each additional user (based on linear regression as shown in figure 23). Even though the 'O' signature provided the best result, its best accuracy was only 90 percent. As can be seen in the next experiment, the use of more complicated signatures increased the accuracy significantly.

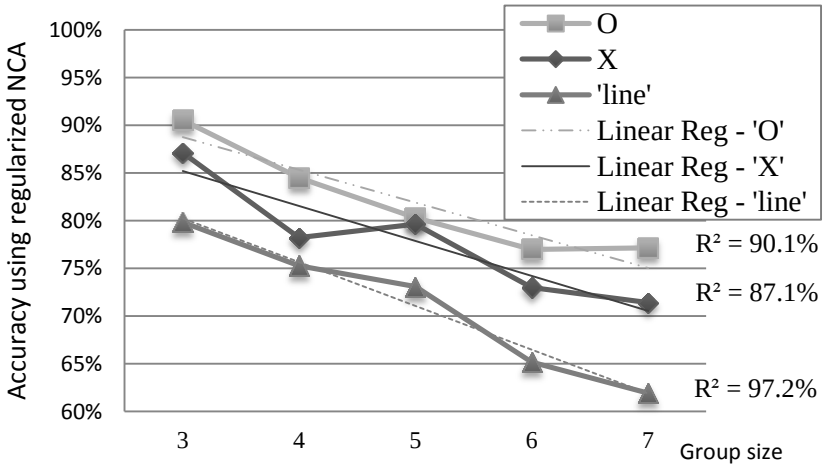

Figure 23 - Accuracy results using regularized NCA for different group sizes and different shapes.

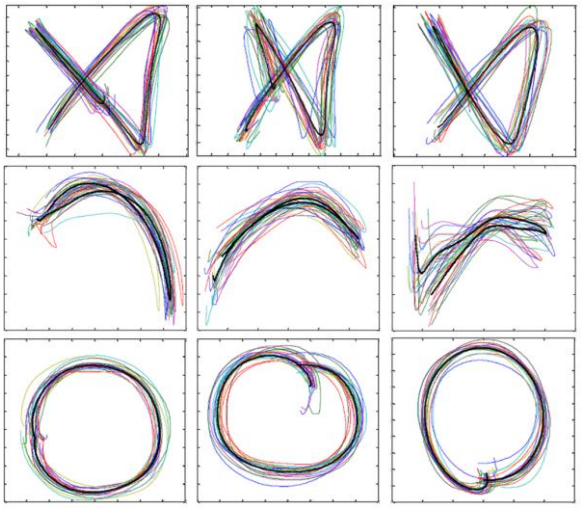

Figure 24 – Various Xs, circles and lines performed by participants

Top row: 'X' shapes performed by three users. These shapes were found to be complex, in a way that prevented the repeatability of performing it. i.e. on the second session, subjects forgot the exact X form that they performed on the first session.

Middle row: 'line' shapes performed by three users. Even though participants were asked to perform a straight line, most users performed curved segments. There is a higher variation between samples in the 'line' shape than in the 'O' shape.

Bottom row: O shapes performed by three users. The shapes have different start points and different curvature values along the trajectory, which allow the differentiation between users.

6.2 Experiment II: Dependent System (User Defined Signature)

In this experiment users selected their own signature and the identification accuracy for user groups with different sizes was determined. Some signatures chosen by subjects are shown in figure 25. As can be seen in figure 26 and table 3, the identification accuracy of all group sizes was above 92 percent using the NCA method. The differences between accuracies in the different group sizes are significant (Kruskal Wallis test p-value=0.01). Two additional tests with regularized NCA and interactive enrollment (NCA reg + Cluster) achieved better accuracies. In order to examine the regularized NCA and the interactive enrollment options which require prior optimization, an evaluation of the accuracy was done using the 13 users whose samples were not used for optimization. These 13 users were only divided into groups of 3 and 4.

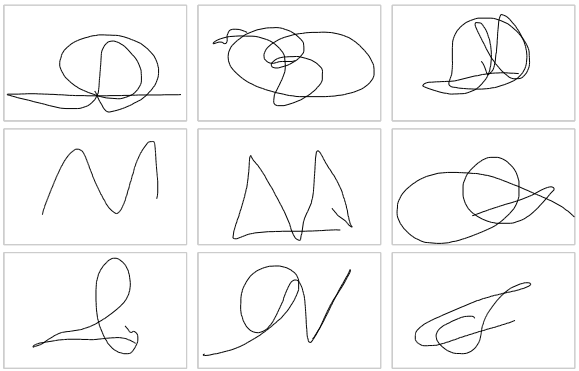

Figure 25 - Examples of signatures chosen by different participants.

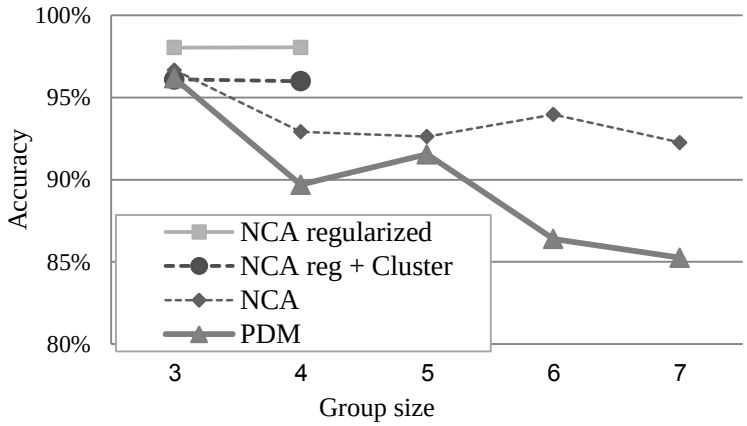

Figure 26 - The accuracy using different distance metric learning methods

	3 Users	4 Users	5 Users	6 Users	7 Users
NCA regularized	98.0%	98.1%	N/A	N/A	N/A
NCA regularized + Clustering	96.1%	96.0%	N/A	N/A	N/A
NCA	96.7%	92.9%	92.6%	94.0%	92.3%
PDM	96.2%	89.7%	91.5%	86.4%	85.3%

Table 3 - Accuracy of the system

For PDM and NCA – accuracy results of 18 groups with 3-7 users enrolled. For regularized NCA and regularized NCA with online learning (clustering), accuracy results for 18 groups of size 3 and 4. Note that these results were calculated without a threshold on the distance, used for user verification.

When analyzing groups with accuracies lower than 80 percent, we found that one specific user was enrolled to all of these groups. The training set, calibration set and test set of this user are presented in figure 27. This exemplifies a user without signature repeatability over time. In a second session (2nd week set) the user performed the signature differently (see the additional curve to the signature in the right part of figure 27). This changed the geometry of the signature and resulted in poor recognition rates for the groups to which the user belonged.

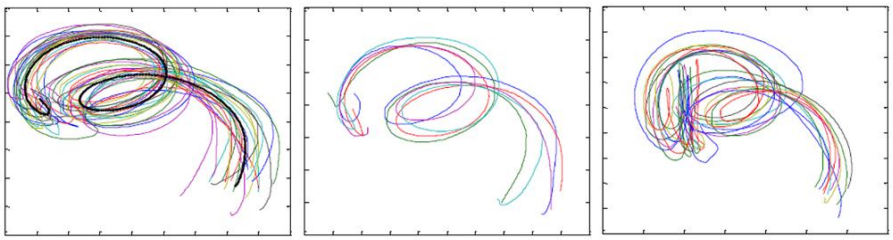

Figure 27 – Differences in the signatures throughout sessions

Training set (left), calibration set (center) recorded on the first week and test set (right) recorded a week later of user 8.

Another test compared the accuracy based on samples of the first week compared with the accuracy of samples recorded on the second week. Since in NCA we take advantage of the first set (i.e. the calibration set) to find an accurate projection of features, we cannot use NCA for this comparison. Instead, we used the PDM method that does not use the calibration set for optimization.

The accuracy of the system with 10 users out of the 13 that were not used for model selection was measured using PDM for the 1st week set's samples and for the 2nd week set's samples. It was found that the accuracy of identifying one user out of 10 is 99 percent using the 1st week set and 81 percent using the 2nd week set, as seen in Table 4. When reducing the number of enrolled users from 10 to 5, the accuracies of both sets increase and reach 100 percent in the 1st week set and 91.5 percent in the 2nd week set. The main reason for this difference is the subjects' difficulty to replicate their signatures. The signature trajectories for some users show big differences between the first week and in the second week.

	First week	**Second week**
Average accuracy	99%	81%
Half width	0.02	0.22

Table 4 - Accuracy in the 1st week and the 2nd week

Differences in accuracy when evaluating 1st week samples vs. 2nd week samples in a system with 10 users enrolled. Half width is $t_{0.975,9} \cdot \sigma_{accuracy}$.

6.3 Experiment III: Forgery

Table 5 describes the results for the entire set of 6 users of experiment III. Note that in Table 4, the results are given for different thresholds. Details of the various thresholds used are described below.

A threshold is a nearness criterion of the distance between the test sample and the training data. Different automatic threshold options were checked (see Table 5) in order to capture the ability to forge a motion signature. The threshold is not fixed and calculated automatically based on each user's calibration set. The threshold options tested include: the distance of the furthest sample out of the 5 calibration samples, which were recorded immediately after training; the 2nd most distant sample; the distance to the furthest sample*1.1; the distance to the furthest sample*1.5; the distance to the furthest sample*2 (i.e. twice the distance to the furthest sample).

	All users (6)		2 stable users	
Threshold	**FAR**	**FRR**	**FAR**	**FRR**
2nd furthest sample	3%	77%	3%	40%
Furthest sample	7%	60%	3%	20%
Furthest +10%	8%	60%	3%	20%
Furthest +50%	13%	47%	10%	10%
Furthest +100%	23%	33%	27%	10%

Table 5 - Forgery experiment results.

Forgery experiment results for different thresholds using the regularized NCA. False Acceptance Rate (FAR): the probability of a forged sample to be accepted. False Rejection Rate (FRR): the probability for a genuine sample to be rejected. All users: All 6 users whose signatures were imitated. 2 stable users: 2 users out of the 6 who were able to repeat their signature accurately in the 2nd session.

The probability of successfully forging a user's signature (i.e. FAR - false acceptance rate) is for most threshold rules below 10 percent. A problem arises with the false rejection rate (FRR). Since the samples that were used for this calculation were captured a week after training, there was a difference between the training set and the test samples. We thus, looked at the two most stable users, i.e. these two users (out of the 6) who had good repeatability of their motion signature as reflected by the smallest distances for the 2nd session's samples). From column 5 of Table 5 we see that the FRR value is lower for all threshold rules. We cannot expect this kind of stability over time from a novice user, but as the user gains experience with the system, we assume the FRR will decrease. In any case, there is an obvious tradeoff between the user's ability to be correctly identified, and the probability of a forgery. For different applications of the system, a different threshold rule can be used. When observing a system allowing 3 attempts to log in, and by looking at the two stable users with the furthest sample threshold as an example, we see that the probability for correct identification in one of three attempts is 99.2 percent, while for an intruder the FAR probability is 9 percent. For 3 attempts using all 6 users, we receive FRR and FAR rates of 6 percent and 19 percent, respectively. Another illustration of the difference between stable users and all users is given in figure 28, where a Receiver Operator Characteristic (ROC) curve is shown. A perfect ratio between True Positive (TP) and False Positive (FP) would yield a point at the top left corner of the graph (i.e., TP=1, FP=0). We can see that for 2 stable users, this TP/FP ratio for any given threshold is better than the ratio of 6 users.

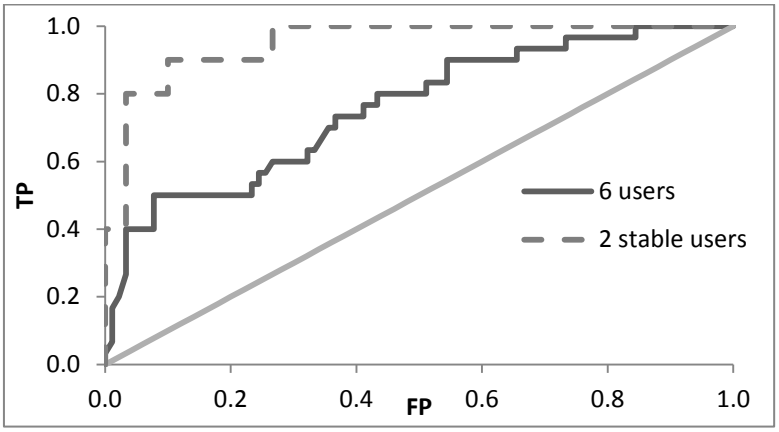

Figure 28 - ROC curves for 6 users (full line) and 2 stable users (dashed line)

6.4 Additional Results

6.4.1 Alignment

For aligning the shapes, several methods were tested: a. Procrustes analysis consisting of translation and scaling; b. translation only and c. Active Shape Models alignment consisting of translation, scaling and rotation of samples. Another strategy was to first divide the trajectory into 3 segments, and align each segment independently of other segments. Figure 29 shows an example of different alignment strategies.

Procrustes analysis in 29(d) was found to be the best strategy. Even though the alignment in 29(e) appears to be very compact, the division into segments also reduces the distance from samples other than the ones performed by the user since the test sample is also divided into segments, and each segment is compared to its counterpart in the model. Therefore the error rate increases. The signature changes its shape in 29(f) since the Procrustes analysis subtracts the mean shape from each sample, and therefore brings all segments to the same location.

Accuracy values for different alignment strategies can be seen in table 6 and figure 30. The division into segments did not yield better results when combined with both Procrustes and ASM. The best strategy was found to be Procrustes analysis, with scaling and translation to the mean shape. This strategy also had the least amount of outlier clusters. The comparison was performed with the regularized NCA classifier on the data of experiment II of user defined signatures.

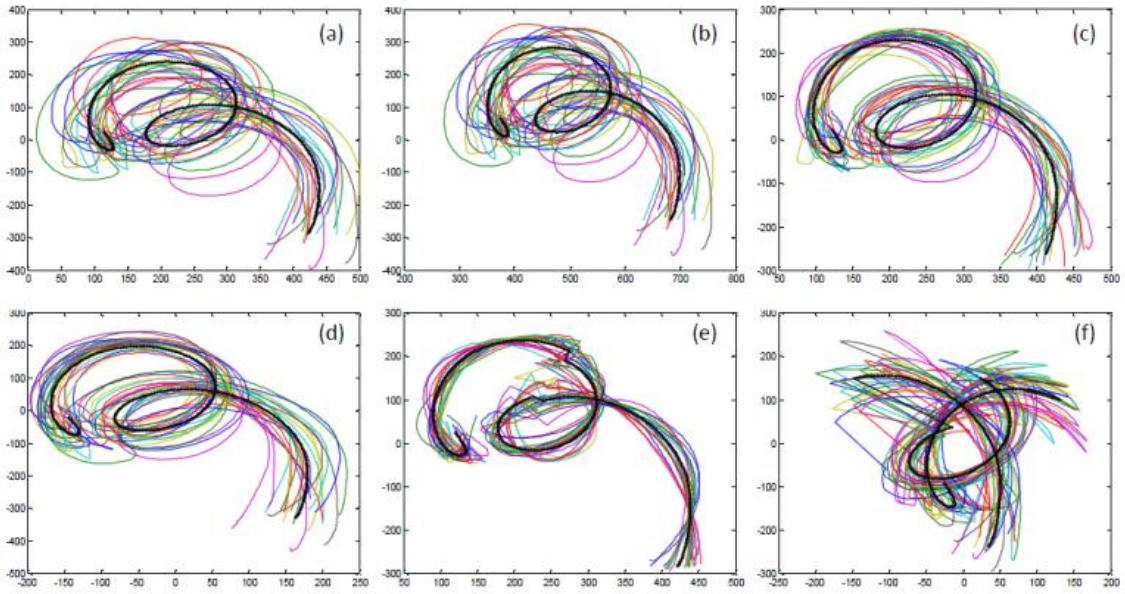

Figure 29 - Different alignment strategies.
Training set before alignment; (b) Translation only; (c) Active shape models alignment; (d) Procrustes analysis (Translation and scaling); (e) ASM alignment performed on 3 segments of the trajectory; (f) Procrustes analysis performed on 3 segments of the trajectory.

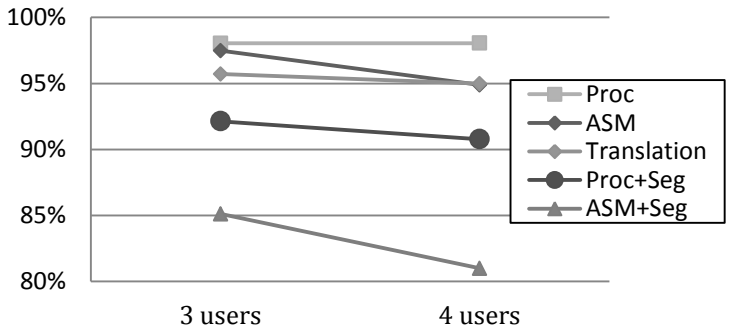

Figure 30 - Accuracy for different alignment methods

	3 Users	4 Users
Procrustes	98%	98%
ASM	97%	95%
Translation	96%	95%
Procrustes + Segments	92%	91%
ASM + Segments	85%	81%

Table 6 - Accuracy for different alignment methods.
The accuracy of the system using different alignment strategies for groups of 3 users and groups of 4 users. Procrustes: Scaling and translation; ASM: Active shape models alignment (scaling, rotation, translation); Translation: translation only; Procrustes + Segments: Scaling and translation for 3 segments; ASM + Segments: ASM alignment for 3 segments.

6.4.2 Online learning / Clustering

The interactive enrollment phase was not a part of the above experiments, and therefore, users did not stop their enrollment nor picked a different signature due to the warnings of the

system. Here we emulate the enrollment process on the samples used in the experiment and use these results to help explain the experimental accuracies obtained. As can be seen in Table 3: NCA reg+cluster option (regularized NCA + online learning), the clustering of samples obtained by the interactive enrollment phase does not improve the accuracy of the system. Even though results have not been improved, we found that the warnings during enrollment were given mostly to users belonging to groups with low accuracy results. For every group with recognition rate of less than 80 percent, warnings of both similarity and variance were given to at least one user. Moreover, all groups whose users did not get any warning eventually were identified with a 100 percent success. Even though, in some cases warnings were given even if the system eventually identified all testing samples (100 percent accuracy), resulting in a false alarm. Out of 18 groups of size 3, at least one user in 22 percent of the groups received a false warning during training (i.e. a warning even when the system's accuracy was 100 percent). Out of the 18 groups of size 4, at least one user of 17 percent of the groups received a false warning.

An important outcome of the enrollment process is to find different forms of movement and outliers in the training set. Figure 31 shows two users whose training samples were clustered to more than one cluster. The top user's samples were clustered to 4 clusters representing different forms of movement. The bottom user's training set was clustered to one primary cluster and 3 additional clusters that reflect outliers in the training set.

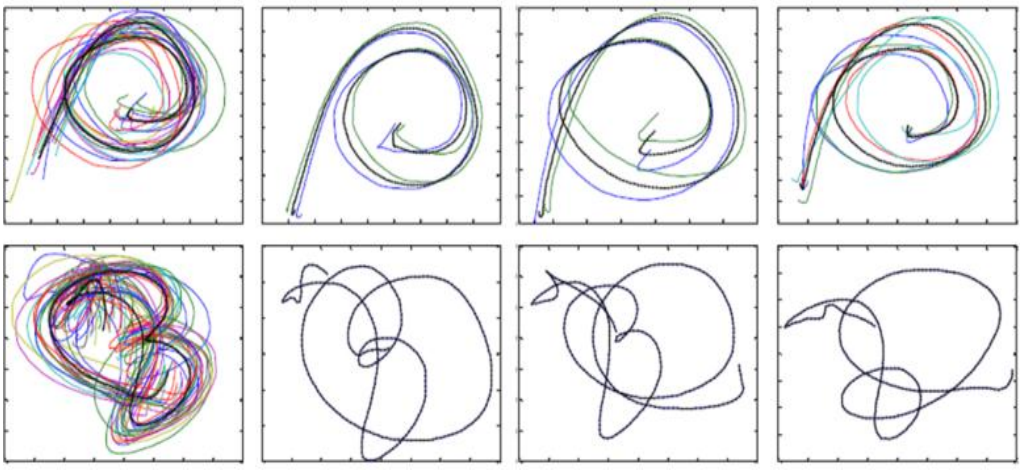

Figure 31 - Clustering of training samples.
Top: Four different forms of movement in the user's training set found by clustering.
Bottom: three outlier samples that do not belong to the primary cluster were found during clustering.

6.4.3 Division into segments

Table 7 show the accuracy of the system with divided segments. The results are good but this technique did not outperform the identification based on the entire trajectory. The main advantage of using high curvature points is the use of additional geometrical information from the trajectory. Curved areas are more complex and therefore it is assumed that more difference between individuals will be found there. The disadvantage is that the variance within an individual's training set can cause different points to be selected for different samples of the same signature. In that case, noise is incorporated into the training set since point locations along the trajectory are changed significantly. Figure 32 show the distribution of points for curvature based segments and equi-distant spacing.

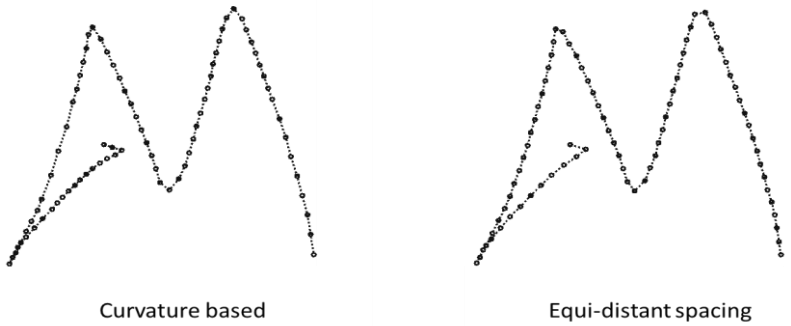

Figure 32 – Distribution of points for the curvature based method and equi-distant spacing.

	3 Users	4 Users
Mean	96.2%	96.3%
S.D.	4.7%	5.2%

Table 7 - Accuracy for trajectories divided into segments

6.5 Statistical Analysis

Three statistical tests were performed for the results analysis:

1. Linear regression for the difference in group size in experiment 1 of predefined signatures.
2. Anova Repeated measures for the difference between different shapes in experiment 1 of predefined signatures.
3. Kruskal Wallis test for the difference between group sizes in experiment 2 of user defined signatures.

6.5.1 Linear regression – experiment 1 – predefined signatures

This test evaluates the reduction in accuracy as a function of the group size.

Hypothesis:

H_0: There is no difference between groups of varying sizes

H_1: Otherwise

Statistical results:

- For the **'O'** shape, the linear regression function was found to be **y=-3.4x+92.2**
- For the **'X'** shape, the linear regression function was found to be **y=-3.7x+88.9**
- For the **'line'** shape, the linear regression function was found to be **y=-4.6x+84.8**

The corresponding r^2 values are: {0.901, 0.871, 0.972} respectively.

The slope in each function is the decay in accuracy for increasing the group size. Since the slope is different from 0 and the r^2 values are all above 0.85, we can say that there is a difference between systems of different sizes. The intercept is the theoretical accuracy for a group with 0 users and irrelevant in this case. See figure 23 for the average accuracy values for each group size and the linear regression trend line. The test was conducted in Microsoft Excel 2010.

6.5.2 ANOVA Repeated Measures – experiment 1 – predefined signatures

This test evaluates the difference in accuracy for different shapes. Since subjects performed the three shapes sequentially, we tested the hypothesis using a general linear model – the Anova repeated measures.

Hypothesis:

H_0: The accuracy is different for each shape

H_1: Otherwise

Statistical results:

See appendix B for the output. Sphericity was found in a significance of 0.19, and therefore a correction of the degrees of freedom was needed. Two corrections were checked: Greenhouse-Geisser, Huynh-Feldt. An additional test with sphericity assumed was also examined. Under all tests, the p-value for the difference between shapes was .00.

The test was conducted in PASW statistics (SPSS) 17.

6.5.3 Kruskal-Wallis test – experiment 2 – user defined signatures

This test is a non-parametric test for the difference between 3 or more populations. We cannot assume that the accuracy has a Gaussian distribution, so we used this non-parametric test.

Hypothesis:

H_0: There is a difference between groups of sizes 3 to 7 in experiment 2

H_1: Otherwise

Statistical results:

The difference between groups was found to be significant with p-value of 0.014.

See table 8 for a detailed output and figure 32 for a box plot of the data. The test was conducted in Matlab R2009b.

Source	SS	df	MS	Chi-sq	Prob>Chi-sq
Columns	14338.2	4	3584.55	12.48	0.0141
Error	122327.3	115	1063.72		
Total	136665.5	119			

Table 8 - Kruskal Wallis test output

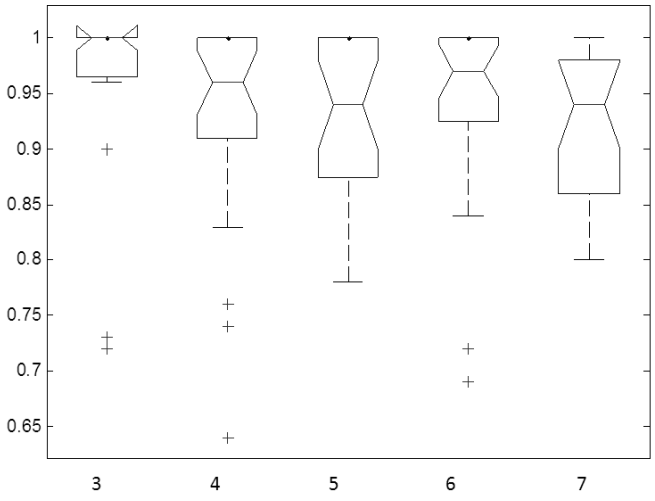

Figure 33 - Box plot for the accuracy of different group sizes

CHAPTER 7: CONCLUSION AND FUTURE WORK

The main hypothesis of this research is that individuals can be distinguished by the attributes of their motion. This hypothesis is validated by a system designed to identify individuals by their hand motion in the form of a free space written signature. It was found that the system is able to identify a user out of many. However, the accuracy relies on the repeatability of the user's hand motion signature.

In a test of an independent system (i.e. all users use the same signature) it was found that for a relatively simple hand gesture (an 'O' shape), the system can identify a user out of a group of 3 users with 90 percent success. However, for a dependent system (each user has a different signature) where more complicated signatures were used, the accuracy significantly increased to over 98 percent for 3 user groups.

In addition, tests for copycat users (forgeries) were made. For users with high repeatability, one can see that both the false rejection rate and false acceptance rate give good results with 91 percent probability for an intruder not being able to log in and 99 percent for a user being able to authenticate in one out of three trials. Users without proper signature repeatability (large motion variability) were more susceptible to forgeries. It is thus suggested that such users undergo training to develop more consistent free space signature motions. This is similar to the phenomena of script variations on paper.

Another development of this research is an interactive signature enrollment procedure, which allows users to select signatures of their choice, for better intuitiveness and comfort. This system interactively ensures that the user's signature is consistent and not similar to the signatures of other users already enrolled in the system. Development of an online real-time enrollment system provided a significant challenge as it required continuous shape realignments, dynamic reconfiguration of the recognition system by automated sequential clustering of the signature classes, and the projection of unknown trajectory configurations into lower dimensional space. This is in contrast to most gesture recognition system, in which the system is designed with a priori knowledge of a fixed number of known signature gestures. The proposed gesture signature system can complement vision-based hand gesture recognition systems for content adaptation, customization, parental control and security.

An interesting future research direction would be to develop a dynamic adaptive system, in which users with a lack of experience can improve their performance over time.

Another topic of interest is the evaluation of a system with similar users such as family members or even twins. Moreover, a generalization of this work to entire body motion is an interesting topic, which fits with many studies and applications that attempt to identify an individual by gait, or understand more about the physical or psychological condition of individuals through automatic motion analysis.

REFERENCES

[1] A.K. Jain, A. Ross, and S. Prabhakar, *An Introduction to Biometric Recognition*, *IEEE Trans. Circuits and Systems for Video Technology, vol. 14*, pp. 4-20, 2004.

[2] H. Stern, D. Frolova, and S. Berman, Hand Gesture Recognition for TV Remote Control using Tree-Based Ensemble and LCS Classifiers, *WORLDCOMP'10 The 2010 World Congress in Computer Science, Computer Engineering, and Applied Computing*, 2010

[3] N. Boulgouris, D. Hatzinakos and K. Plataniotis. Gait recognition: A challenging signal processing technology for biometric identification. *IEEE Signal Process. Mag. 22(6),* pp. 78-90. 2005.

[4] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Readings in Speech Recognition 53(3),* pp. 267–296. 1990.

[5] T. Shimshon, R. Moskovitch, L. Rokach and Y. Elovici. Continuous verification using keystroke dynamics. *International Conference on Computational Intelligence and Security*. 2010.

[6] R. Moskovitch, C. Feher, A. Messerman, N. Kirschnick, T. Mustafić, A. Camtepe, B. Löhlein, U. Heister, S. Möller and L. Rokach. Identity theft, computers and behavioral biometrics. *Proceedings of the 2009 IEEE International Conference on Intelligence and Security Informatics*. 2009.

[7] A. B. Sriwarno. The application of motion pattern recognition in the behaviometrics of human kinematics. *Prosiding Seminar Nasional Aplikasi Teknologi Informasi (SNATI)*. 2009.

[8] S. Prabhakar, S. Pankanti and A. Jain. Biometric recognition: Security and privacy concerns. *IEEE Security & Privacy (USA).* pp. 33. 2003.

[9] M. Nixon. Gait biometrics. *Biometric Technology Today 16(7-8),* pp. 8-9. 2008.

[10] A. Weiss, A. Ramapanicker, P. Shah, S. Noble and L. Immohr. Mouse movements biometric identification: A feasibility study. *Proc.Student/Faculty Research Day, CSIS, Pace University, White Plains, NY*. pp. 1-8. 2007.

[11] M. Pusara and C. E. Brodley. User re-authentication via mouse movements. *Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security*. 2004.

[12] M. A. Giese and T. Poggio. Neural mechanisms for the recognition of biological movements. *Nature Reviews Neuroscience 4(3),* pp. 179-192. 2003.

[13] F. Loula, S. Prasad, K. Harber and M. Shiffrar. Recognizing people from their movement. *Journal of Experimental Psychology: Human Perception and Performance 31(1),* pp. 210-220. 2005.

[14] N. F. Troje, C. Westhoff and M. Lavrov. Person identification from biological motion: Effects of structural and kinematic cues. *Percept. Psychophys. 67(4),* pp. 667. 2005.

[15] E. Farella, S. O'Modhrain, L. Benini and B. Riccó. Gesture signature for ambient intelligence applications: A feasibility study. *Pervasive Computing* pp. 288-304. 2006.

[16] J. Liu, L. Zhong, J. Wickramasuriya and V. Vasudevan. uWave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive and Mobile Computing 5(6),* pp. 657-675. 2009.

[17] A. K. Jain, F. D. Griess and S. D. Connell. On-line signature verification. *Pattern Recognition 35(12),* pp. 2963-2972. 2002.

[18] R. Baecker, J. Grudin, B. Buxton, and S. Greenberg. (ed.) *Readings in Human Computer Interaction: Towards the Year 2000,* (2nd ed.). San-Francisco, CA: Morgan-Kaufman. 1995. pp. 1-2.

[19] A. Blanford. Intelligent interaction design: The role of human-computer interaction research in the design of intelligent systems. *Expert Systems* 2002.

[20] J. Bennett. Managing to meet usability requirements: Establishing and meeting software development goals. *Visual Display Terminals.* pp. 161–84. 1984.

[21] B. Shackel. Usability–context, framework, definition, design and evaluation. *Interact Comput 21(5-6),* pp. 339-346. 2009.

[22] Y. Wu and T. S. Huang. Vision-based gesture recognition: A review. *Urbana* 1999.

[23] J. Wachs, H. Stern, Y. Edan, M. Gillam, C. Feied, M. Smith and J. Handler. Gestix: A doctor-computer sterile gesture interface for dynamic environments. *Soft Computing in Industrial Applications: Recent and Emerging Methods and Techniques* pp. 30–39. 2007.

[24] V. I. Pavlovic, R. Sharma and T. S. Huang. Visual interpretation of hand gestures for human-computer interaction: A review. *IEEE Trans. Pattern Anal. Mach. Intell. 19(7),* pp. 677-695. 1997.

[25] H. Stern, J. Wachs and Y. Edan. Optimal hand gesture vocabulary design using psycho-physiological and technical factors. *The 7th International Conference Automatic Face and Gesture Recognition, Southampton, UK*. 2006.

[26] I. Merriam-Webster. *Merriam-Webster's Medical Dictionary* 1995.

[27] K. Fukunaga, *Introduction to Statistical Pattern Recognition* (2nd Edition). Academic Press, 1990. pp. 399-400

[28] D. A. Winter. *Biomechanics and Motor Control of Human Movement* (4th Edition). New Jersey: John Wiley and Sons. 2009. pp. 2.

[29] T. Flash and N. Hogan. The coordination of arm movements: An experimentally confirmed mathematical model. *Journal of Neuroscience 5(7),* pp. 1688. 1985.

[30] Y. Uno, M. Kawato and R. Suzuki. Formation and control of optimal trajectory in human multijoint arm movement. *Biol. Cybern. 61(2),* pp. 89-101. 1989.

[31] M. Kawato, K. Furukawa and R. Suzuki. A hierarchical neural-network model for control and learning of voluntary movement. *Biol. Cybern. 57(3),* pp. 169-185. 1987.

[32] A. Feldman, "The nature of voluntary control of motor actions," in *Motor Control and Learning*. Springer, 2006.

[33] D. Bennequin, R. Fuchs, A. Berthoz and T. Flash. Movement timing and invariance arise from several geometries. *PLoS Comput. Biol. 5.* 2009.

[34] S. Berman, D. G. Liebermann and T. Flash. Application of motor algebra to the analysis of human arm movements. *Robotica 26(04),* pp. 435-451. 2007.

[35] J. H. Gallier. *Geometric Methods and Applications: For Computer Science and Engineering.* New York: Springer. 2001. pp. 6-7

[36] S. Schaal, D. Sternad, R. Osu and M. Kawato. Rhythmic arm movement is not discrete. *Nat. Neurosci. 7(10),* pp. 1136-1143. 2004.

[37] C. Atkeson and J. Hollerbach. Kinematic features of unrestrained vertical arm movements. *Journal of Neuroscience 5(9),* pp. 2318. 1985.

[38] D. G. Kendall. Shape manifolds, procrustean metrics, and complex projective spaces. *Bulletin of the London Mathematical Society 16(2),* pp. 81. 1984.

[39] T. F. Cootes, C. J. Taylor, D. H. Cooper and J. Graham. Active shape models-their training and application. *Comput. Vision Image Understanding 61(1),* pp. 38-59. 1995.

[40] P. Viviani and T. Flash. Minimum-jerk, two-thirds power law, and isochrony: Converging approaches to movement planning. *Journal of Experimental Psychology: Human Perception and Performance 21(1),* pp. 32-53. 1995.

[41] G. Farin. *Curves and surfaces for computer aided geometric design.* (4^{th} Edition), San Diego, CA: Academic Press, 1997.

[42] R. B. Chen, *Lecture notes in regression analysis, Topic: Polynomial Regression Models*. Institute of statistics, National university of Kaohsiung," 2003.

[43] A. Ross and A. Jain. Information fusion in biometrics. *Pattern Recog. Lett. 24(13),* pp. 2115-2125. 2003.

[44] S. B. Kotsiantis. Supervised machine learning: A review of classification techniques. *Editorial Boards, Publishing Council 31*pp. 249-268. 2007.

[45] A. Klein, *lecture notes in Digital Image Processing. Topic: Unsupervised Classification*, Texas University, 2008.

[46] C. Cortes and V. Vapnik. Support-vector networks. *Mach. Learning 20(3),* pp. 273-297. 1995.

[47] D. Meyer, F. Leisch and K. Hornik. The support vector machine under test. *Neurocomputing 55(1-2),* pp. 169-186. 2003.

[48] M. Pontil and A. Verri. Support vector machines for 3 D object recognition. *IEEE Trans. Pattern Anal. Mach. Intell. 20(6),* pp. 637-646. 1998.

[49] Y. Ren and F. Zhang. Hand gesture recognition based on MEB-SVM. *Proceedings of the 2009 International Conference on Embedded Software and Systems*. 2009.

[50] S. Berman, J. Friedman, G. Bakir and T. Flash, "Action identification for teleoperation based on object - action abstraction," in *IEEE SMC International Conference on Distributed Human-Machine Systems (DHMS),* 2008.

[51] A. D. Wilson and A. F. Bobick. Hidden markov models for modeling and recognizing gesture under variation. *Int. J. Pat. Recognit. Artif. Intell. 15(1),* pp. 123-160. 2001.

[52] N. Warakagoda, *Hidden Markov Models*, Internet: http://jedlik.phy.bme.hu/~gerjanos/HMM/node2.html, 10/05/96 [10/08/11], 1996.

[53] F. Nigsch, A. Bender, B. van Buuren, J. Tissen, E. Nigsch and J. B. O. Mitchell. Melting point prediction employing k-nearest neighbor algorithms and genetic parameter optimization. *J.Chem.Inf.Model 46(6),* pp. 2412-2422. 2006.

[54] R. C. Tryon, *Cluster Analysis,* Ann Arbor, Mich.: Edward Bros. 1939.

[55] R. O. Duda, P. E. Hart and D. G. Stork. *Pattern Classification*, New York: Wiley, 2001.

[56] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, 1967.

[57] S. Theodoridis, K. Koutroumbas and R. Smith. *Pattern Recognition* (4th Edition). San Diego, CA: Academic Press, 2009. pp; 627-652.

[58] Mei Xiao and Lei Zhang. A background reconstruction algorithm based on two-threshold sequential clustering. *Computing, Communication, Control, and Management, CCCM '08. ISECS International Colloquium*. 2008.

[59] J. Goldberger, S. Roweis, G. Hinton and R. Salakhutdinov. Neighbourhood components analysis. *NIPS 17* pp. 513-520. 2005.

[60] C. D. Cantrell. *Modern Mathematical Methods for Physicists and Engineers* 2000.

[61] A. Bar-Hillel, T. Hertz, N. Shental and D. Weinshall. Learning a mahalanobis metric from equivalence constraints. *Journal of Machine Learning Research 6(1),* pp. 937. 2006.

[62] I. Parmet. *Lecture notes in selected subjects in statistics, Topic: Principal Components Analysis.* Ben Gurion University, 2009.

[63] A. L. Yuille, P. W. Hallinan and D. S. Cohen. Feature extraction from faces using deformable templates. *International Journal of Computer Vision 8(2),* pp. 99-111. 1992.

[64] M. Kass, A. Witkin and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision 1(4),* pp. 321-331. 1988.

[65] R. Bowden, *non-linear Point Distribution Models*. Internet: http://info.ee.surrey.ac.uk/Personal/R.Bowden/publications/cvonline/nlpdm/index.html, 29/10/98 [01/08/2010]. 1998.

[66] P. D. Sozou, T. F. Cootes, C. J. Taylor, E. C. Di Mauro and A. Lanitis. Non-linear point distribution modelling using a multi-layer perceptron. *Image Vision Comput. 15(6),* pp. 457-463. 1997.

[67] T. F. Cootes and C. J. Taylor. Statistical models of appearance for computer vision. *World Wide Web Publication, February* 2001.

[68] Ahmad, T. Taylor, CJ Lanitis, A. Cootes, TF. Tracking and recognising hand gestures, using statistical shape models. *Image Vision Comput. 15(5),* pp. 345-352. 1997.

[69] Van Ginneken, B. Frangi, A.F., J. J. Staal, B. M. ter Haar Romeny and M. A. Viergever. Active shape model segmentation with optimal features. *IEEE Trans. Med. Imaging 21(8),* pp. 924-933. 2002.

[70] G. Hamarneh. Active shape model: Modeling shape variations and gray level information and an application to image search and classification. *Dept. of Signals and Systems, Chalmers Univ. of Technology, Sweden*. *Citeseer.* 1998.

[71] N. Singh-Miller, M. Collins and T. J. Hazen. Dimensionality reduction for speech recognition using neighborhood components analysis. *Proceedings of Interspeech*. 2007.

[72] Z. Yang and J. Laaksonen. Regularized neighborhood component analysis. *Image Analysis* pp. 253-262. 2007.

[73] S. Stevens, *Calculus of Several Variables lecture notes*. Internet: http://math.bd.psu.edu/faculty/stevens/Old-Courses/MATH231/Notes/rvaTNsKformulas.pdf, 4/01/2010 [10/08/11]. 2003.

[74] D. K. Wagg and M. S. Nixon. On automated model-based extraction and analysis of gait. *Proceedings of the Sixth IEEE International Conference on Automatic Face and Gesture Recognition*. 2004.

APPENDIX I: SYSTEM ARCHITECTURE

Click on the icon to extract a zip file with HTML based help files that contain information and source code. After extraction, open UserIdentification/doc/FinalSystem/index.html
For the system activation, refer to UserIdentification/start.m. Note that the PrimeSense camera has to be installed for the identification system to function.


UserIdentification.zip

The system was developed in Matlab version 2009b using object oriented principles. The system has four classes, and additional external functions. The functions are: System, User, Cluster and Sample. A class diagram is shown in figure 34, and a full description of the code is shown in the attached zip file to this appendix. The System class controls the inter-users variables such as the NCA matrix and the user list. Each user in the user list of class System is an instance of User. Each user has its training and calibration samples that were recorded during enrollment. All samples that were used for identification are kept in a Test Set array of samples. The class Cluster inherits User since it has very similar characteristics: an array of training samples, and representative feature values (mean shape, velocity and curvature values). The class Sample is used for storing information of samples: the initial trajectory; the fitted trajectory; an aligned version to the cluster the sample is currently assigned to; and velocity and curvature values. All operations on the trajectory are functions in the Sample class. In addition to these classes, additional static functions were used for interpolation, optimization, plotting and additional operations that are being used during the activation of the system.

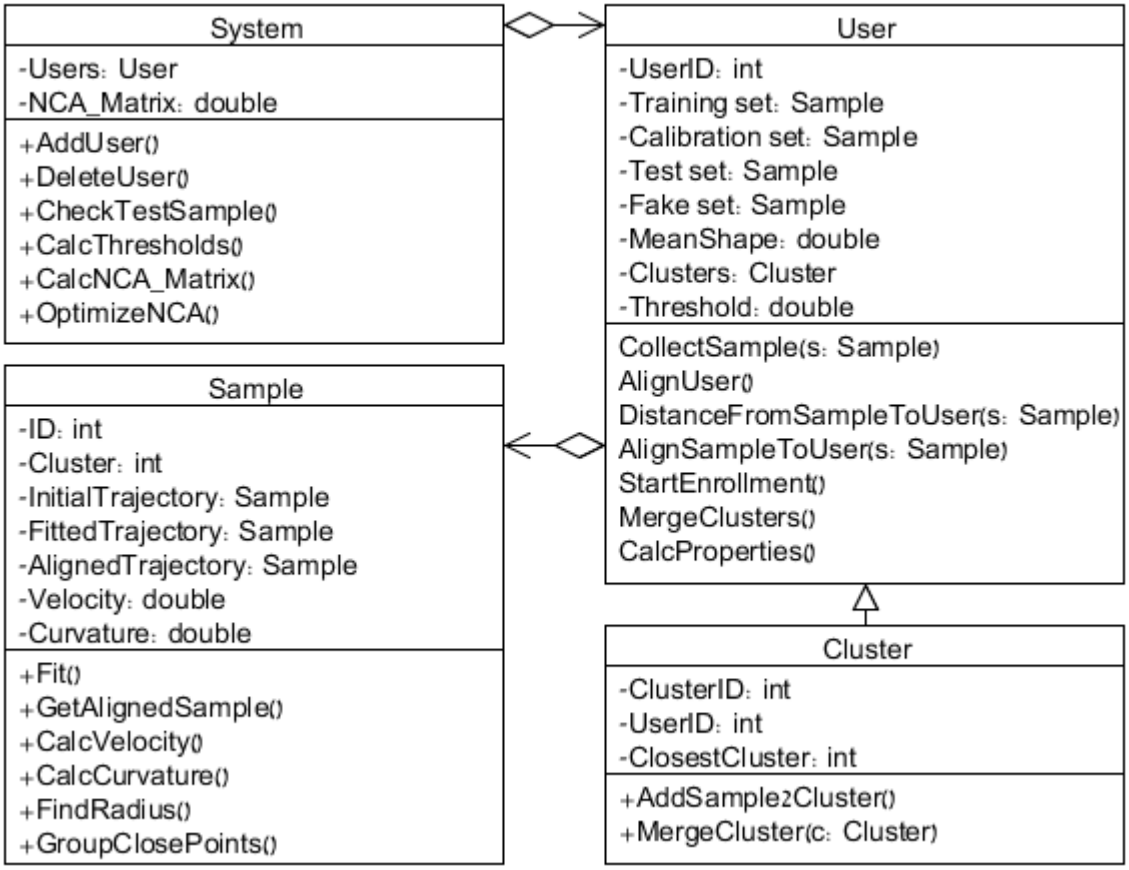

Figure 34 - A class diagram (simplified) of the system.

Refer to the appendix 1 for a full description of the classes, properties and methods.

Class description

Class User

Description

Keeps information of a user enrolled to the system.

Properties

Variable name	**Description**
Aligned	A boolean variable for determining if the samples of this user were already aligned.
AlignedMeanTrainingShape	The mean shape after alignment
AvgCurvature	An average of the curvature across different segments on the trajectory
AvgVelocity	An average of the velocity across different segments on the trajectory.

Clusters	An array of type Cluster containing the clusters of the user
Fake1	5 Fake samples done by a different user
Fake2	5 Fake samples done by a different user
Fake3	5 Fake samples done by a different user
Fake4	5 Fake samples done by a different user
Fake5	5 Fake samples done by a different user
ID	the user's ID
MeanTrainingShape	the mean shape before alignment
NumberOfClusters	The number of clusters that belong to this user
PDM	A struct containing the matrices needed for the PDM analysis (b matrix, X Covariance matrix etc.)
Threshold	The distance for which samples are no longer declared as belonging to this user.
testingSamples1	Reference set of 5 samples, used for optimization and threshold determination
testingSamples2	The test samples that were taken a week later, used for evaluation of the accuracy
trainingSamples	an array of Samples - the training set

Methods

Type	**Function name**	**Description**
	AddSample	Adds a new sample to the set, or creates a set if the set is
	AddSet	type: 1: training, 2: testing1, 3:testing2. 4: fake1, 5:fake2.
static	AlignOneSampletoMean	This function aligns the test sample to the database, in
static	Alignment	This function aligns Training Set shapes to the Mean shape.
static	CalcPDM	Function creates Point Distribution Model Set.
	CopyConstructor	Copies the contents of u into a new user u2
	CreateSampleSet	returns a 3D trajectory set of either the initial or fitted trajectory of this user.
static	ProcAlign	Procrustes analysis - Scales and translates a sample to the mean
	TTSAS	Performs sequential clustering of the samples.
	ThreeDistances	Returns distances from a test sample to three samples of a user: the closest to the sample, the furthest to the sample and the mean shape.
	align2User	Align one sample to the mean of the user
	alignBySegment	same as above, aligns parts of the sample and not the entire sample
	alignTestSets	Aligns the test sets (1 and 2) to the mean shape of the training
	alignUser	aligns all samples to each other and calculates a mean shape accordingly
	calculateTrMean	Calculates the mean of the training set.
	compare2User	Returns the distance from a sample to the mean shape of

		the
	completeSet	Makes sure the user has 20 training samples by taking the missing samples from test1
	copySamples	copies samples from one user to another (similar to copy constructor)
	deleteSample	Delete sample i from sets 1,2,3 = training, test1,test2
	fitSet	Creates a fitted set for the user's samples
	getAlignedSamples	Returns X Y and Z coordinates of the sample set. should be
	getSamples	returns X Y and Z coordinates of the sample set. should be activated only after fitting.
	mergeClusters	Summary of this function goes here
	plotClusters	Plots samples as part of clusters, should be done after
	plotFakeSet	Plots only the fake samples
	plotTestSet	Plots only the test sets (1 and 2)
	plotUser	Plots the user's training set. samples in colors and mean in
	plotUserBeforeAlign	Plots training set before alignment
	setPDM	Calculate the PDM variables, calls CalcPDM - the static function
	startOfflineClustering	Offline clustering of training samples
	startSeqClustering	Sequential clustering of training samples. This function is same

Class Experiment (System)

Description

Experiment Holds information for multiple users, and contains all variables and function that do not belong to a certain user. It is called "Experiment" because it was used to evaluate the program, but could also be called "System" for the finalized version, holding user information and inter-user information such as the NCA matrix.

Properties

Accuracy	The value of the accuracy from 0 to 1 for the users enrolled to the system.
Clustered	A binary value. 1: the users in this experiment were clustered, 0: otherwise.
Confusion	an array of mX2 where m is the total number of test samples that exist for all users enrolled. In column 1 there is the user that performed the sample, in column 2 there is the classification result.
Distances	For each test sample of all users, what are the distances based on the current method (NCA, PCA) from the sample to each user.

NCAMatrix	the projection matrix found by NCA
ShapeType	In case of a shape experiment, which shape (X,O or ->) is currently evaluated in the present experiment.
Users	An array of type User holding the users enrolled to this experiment/system.
number	the serial number of the experiment, in case multiple experiments take place.

Methods

	AddUser	Adds a User to the User array of Experiment.
	AlignUsers	Aligns all users in the experiment, by calling the alignUser function of class User for each user enrolled.
	Calc3Distances	Calculate distances between training sets and testing samples
	CalcDistance	Calculate distances between training sets and testing samples
	CalcFakes	Calculates distances from all fake samples to the user
	CalcThresholds	finds the optimal threshold for each user, based on 5 calibration samples
	CheckSample	Calculate distances between one sample to each user's training set
	CreateClusters	Creates clusters for users, and sends clusters of previously enrolled user to the user currently enrolling, for all-cluster comparison
	InitializePDM	Deletes all information stored for the experiment: PDM, Aligned, mean shape and aligned mean shape, prior to a new calculation.
	NCA	Creates an NCA projection matrix for all users enrolled.
	RecordSample	opens a GRS tracker capture window, then creates the track
	completeSets	Calls the completeSet function of User: makes sure that each user has exactly 20 samples in the training set.
	createTrainSet	Puts all users in the same matrix
	deleteUser	Deletes a user based on the results of the online clustering. currently not active
Static	minimize	A continuous differentiable multivariate function.
Static	ncaExp	NCA Performs NCA on the specified dataset
Static	nca_lin_grad	Computes NCA gradient on the specified dataset
	notify	Notify listeners of event.
	optimizeNCA	Runs NCA 5 times or until a proper accuracy was achieved for the reference samples.
	plotAllUsers	Creates a subplot of all users enrolled for comparison
	reFitEverything	Calls reFitUsers for all sets available - training, reference (test 1), test (test 2) and fake samples
	reFitUsers	Creates a new fit based on new parameters for all samples of a given set (in AdditionalSets).

Class Cluster

Description

This class keeps samples of a certain user that are close to each other. Inherits User

Properties

ClusterNum	The serial number of a cluster
UserID	Serial number of the user holding this cluster
ClosestCluster	Which cluster of UserID is closest to this cluster.

Methods (all non-static)

AddSample	Adds a new sample to the set, or creates a set if the set is empty
calculateTrMean	Calculates the mean shape by calling the super function calcultateTrMean from class User
MergeClusters	Adds the samples in two clusters together into one array and aligns them together. returns the new cluster
compare2Cluster	Calculates the Euclidean distance between this cluster to a test sample

Class Sample

Description

Contains all information needed from a user sample

Properties

Class	Which user it belongs to
Cluster	Which cluster holds this sample
FileName	the txt file from which the sample was taken from.
Number	Sample ID
PCs	The PCs of this sample (previous variables*EigenVectors) for each cluster
RotationRatio	The amount for which the sample was rotated from the initial trajectory
ScalingRatio	The amount for which the sample was scaled from the initial trajectory
TotalLength	The trajectory total length
Xaligned	the fitted trajectory X coordinates, currently aligned to a certain user/cluster.
Xcoor	X coordinates of the initial trajectory
Xfit	X coordinates after curve fitting.
Yaligned	the fitted trajectory Y coordinates, currently aligned to a certain user/cluster.
Ycoor	Y coordinates of the initial trajectory
Yfit	Y coordinates after curve fitting.
Zaligned	the fitted trajectory Z coordinates, currently aligned to a certain user/cluster.

Zcoor	Z coordinates of the initial trajectory
Zfit	Z coordinates after curve fitting.
Zmean	mean value for Z, used for absolute scaling
b	Shape parameters: b=P'(X-Xavg) for each cluster
curvature	the curvature values of different segments across the trajectory
velocity	the velocity values of different segments across the trajectory

Methods

Static	GetTrajectoryCurvature	Calculate the curvature of each point
Static	calcCurvature	for each segment of the fitted trajectory, calculates the average curvature
Static	calcVelocity	for each segment of the fitted trajectory, calculates the average velocity
Static	cleanTrajectory	This function removes points from the original trajectory. If points are grouped at the beginning or the end of the trajectory, and if they are too close to one another (~0), this function removes them.
Static	findRadius	returns the radius of the bounding circle using Euclidean distance
	fit	Calls the fitting function with the above parameters, and adds the trajectory to the sample's variables.
	getAlignedSample	Returns the sample after it has been aligned.
	getFittedSample	Returns a 1X3n vector of the fitted sample of the trajectory, where n is the number of points along the fitted trajectory.
Static	groupClosePoints	This function checks Euclidean distances between points. if the distance is at the bottom 10% (after reducing distances < 0.005), the point is removed
Static	interparc	interpolate points along a curve in 2 or more dimensions, while keeping equi-distant spacing between points
Static	newSegments	This function creates a fitted trajectory with n segments, each segment has the ratio of points that exist on the same segment on the initial trajectory.
Static	newSegmentsCurv	This function creates a fitted trajectory with a number of segments based on the number of points with curvature > threshold.
	plotSample	Plots the sample to the screen

Additional functions that were used and were not a part of a class

Capture1File	Takes a txt file from a predefined folder, and transforms the trajectory in the txt to an instance of Sample
CaptureFilesLab	Transform multiple txt files into Sample instances. Was used with the experiment data
evalAccuracy	Calculates the accuracy of the system in E (Experiment class instance) params are the parameters decided by the optimizing function (e.g. Simulated Annealing)
Optimize	Call a simulated annealing function with different inputs
PlotSet	Plots a set of trajectories to the screen, giving each trajectory a different color
runPS	Calls the capture program and the batch utility (for the creation of a trajectory file)
Start	A simplified tutorial explaining the different methods that were used.
UpdateParameters	Changes a structure of parameters that is being passed to all functions in the system.

APPENDIX II – LIST OF PARAMETERS

Fitting and Cleaning

NoOfPoints=300: *The final number of points on each fitted curve*
NoOfSegments=1: *How many segments should the curve be separated to*
DivideOrNot=0: *1 if the curve should be divided to segments, 0 if not.*
StartAtSample = 1: *if the first samples are to be neglected*
FitType=2: 1= *equal length segments, 2 = curvature based segments*
Aspect = 0.1: *the percentage of the final length that is to be neglected, in start and end points.*
CurveType=2: *1= Hermite 2=Spline, 3 Linear*
FitZ=0: *Whether to use Z coordinates too, or just 2D. 1 = Don't use Z, 2 = use 3D curves.*
GroupOrNot =1: *1: group close points, 0: don't group.*
TemporalSegments = 5: *How many segments should the curve be separated for velocity and curvature calculation*
CurvType = 1: *1: point based, 2: splines based*

Alignment

ScaleOrNot=1: *1 = scaling, 0=no scaling*
RotateOrNot=0: *1 = rotate, 0= no rotation*
AlignmentSegments = 3: *The number of segments to be aligned separately.*
AlignBySegments=0: *0 - normal, 1- by segments*
ScaleOrNot1smp=1: *same as previous, hold for aligning one sample to the mean of a cluster/class*
RotateOrNot1smp=0: *same as previous, hold for aligning one sample to the mean of a cluster/class*

PDM and Classification

CovOrCor=1: 1 = *Covariance matrix PCA, 2 = Correlation matrix PCA*
NoOfPCs=7: *The final number of PCs (if not using clusters)*
Classifier=2; %1: *Mahalanobis PDM, 2: Mahalanobis NCA*
lambda=0.9435: *the regularization parameter of NCA*

Clustering:

ClusterOrNot=0: *0 if no clustering should take place.*
theta1=3087: *The maximum distance for creating a new cluster*
theta2=3373: *If the distance to the closest cluster is below theta2 and above theta1, it is kept aside and not clustered. All samples kept aside will be clustered after all samples have been introduced to the system.*
MaxClusters1=15: *the maximum number of clusters a user can have before merging*
MaxClusters2=10: *the maximum number of clusters a user can have after merging*
Closeness =3241: *The minimum distance for merging two clusters.*
DistanceType =1: *Distance between a sample and a cluster.*

DeleteOrJoin=2: 1 *if a small cluster should be deleted, 2 if should be joined to the closest cluster*

Forgery

*A distance from a samples to a set is calculated by a*Min+b*Mean+c*Max Where max and min are the closest and furthest samples in the set from the testing sample.*
MeanWeight = 1: *The weight given to the mean shape*
MinWeight = 0: *The weight given to the closest shape*
MaxWeight = 0: *The weight given to the furthest shape*
AcceptanceRatio = 1: *whether to accept 4 out of the 5 testing samples, or all of them (1)*

Experiment:

CheckedSet=2: *which set to check. 1-only first set, 2- second set, 3-Fake*
UseThreshold=0: *Whether to check if a smallest distance < threshold*

APPENDIX III: STATISTICAL TESTS OUTPUT

Anova Repeated measures SPSS output

Within-Subjects Factors

Measure:MEASURE_1

Shape	Dependent Variable
1	X
2	O
3	line

Multivariate Tests[c]

Effect		Value	F	Hypothesis df	Error df	Sig.
Shape	Pillai's Trace	.258	14.580[a]	2.000	84.000	.000
	Wilks' Lambda	.742	14.580[a]	2.000	84.000	.000
	Hotelling's Trace	.347	14.580[a]	2.000	84.000	.000
	Roy's Largest Root	.347	14.580[a]	2.000	84.000	.000
Shape * Num	Pillai's Trace	.031	.334	8.000	170.000	.952
	Wilks' Lambda	.969	.330[a]	8.000	168.000	.954
	Hotelling's Trace	.031	.327	8.000	166.000	.955
	Roy's Largest Root	.023	.484[b]	4.000	85.000	.748

a. Exact statistic

b. The statistic is an upper bound on F that yields a lower bound on the significance level.

c. Design: Intercept + Num
Within Subjects Design: Shape

Mauchly's Test of Sphericity[b]

Measure:MEASURE_1

Within Subjects Effect	Mauchly's W	Approx. Chi-Square	df	Sig.	Epsilon[a]		
					Greenhouse-Geisser	Huynh-Feldt	Lower-bound
Shape	.910	7.893	2	.019	.918	.981	.500

Tests the null hypothesis that the error covariance matrix of the orthonormalized transformed dependent variables is proportional to an identity matrix.

a. May be used to adjust the degrees of freedom for the averaged tests of significance. Corrected tests are displayed in the Tests of Within-Subjects Effects table.

b. Design: Intercept + Num
Within Subjects Design: Shape

Tests of Within-Subjects Effects

Source		Type III Sum of Squares	df	Mean Square	F	Sig.
Shape	Sphericity Assumed	.540	2	.270	14.111	.000
	Greenhouse-Geisser	.540	1.835	.294	14.111	.000
	Huynh-Feldt	.540	1.962	.275	14.111	.000
	Lower-bound	.540	1.000	.540	14.111	.000
Shape * Num	Sphericity Assumed	.047	8	.006	.308	.962
	Greenhouse-Geisser	.047	7.342	.006	.308	.954
	Huynh-Feldt	.047	7.849	.006	.308	.961
	Lower-bound	.047	4.000	.012	.308	.872
Error(Shape)	Sphericity Assumed	3.250	170	.019		
	Greenhouse-Geisser	3.250	156.008	.021		
	Huynh-Feldt	3.250	166.787	.019		
	Lower-bound	3.250	85.000	.038		

Tests of Within-Subjects Contrasts

Measure:MEASURE_1

Source	Shape	Type III Sum of Squares	df	Mean Square	F	Sig.
Shape	Linear	.208	1	.208	8.538	.004
	Quadratic	.331	1	.331	23.951	.000
Shape * Num	Linear	.021	4	.005	.215	.929
	Quadratic	.026	4	.007	.472	.756
Error(Shape)	Linear	2.075	85	.024		
	Quadratic	1.175	85	.014		

Tests of Between-Subjects Effects

Measure:MEASURE_1
Transformed Variable:Average

Source	Type III Sum of Squares	Df	Mean Square	F	Sig.
Intercept	159.816	1	159.816	7433.320	.000
Num	.856	4	.214	9.953	.000
Error	1.827	85	.021		

APPENDIX IV – A FLOWCHART FOR INTERACTIVE ENROLLMENT

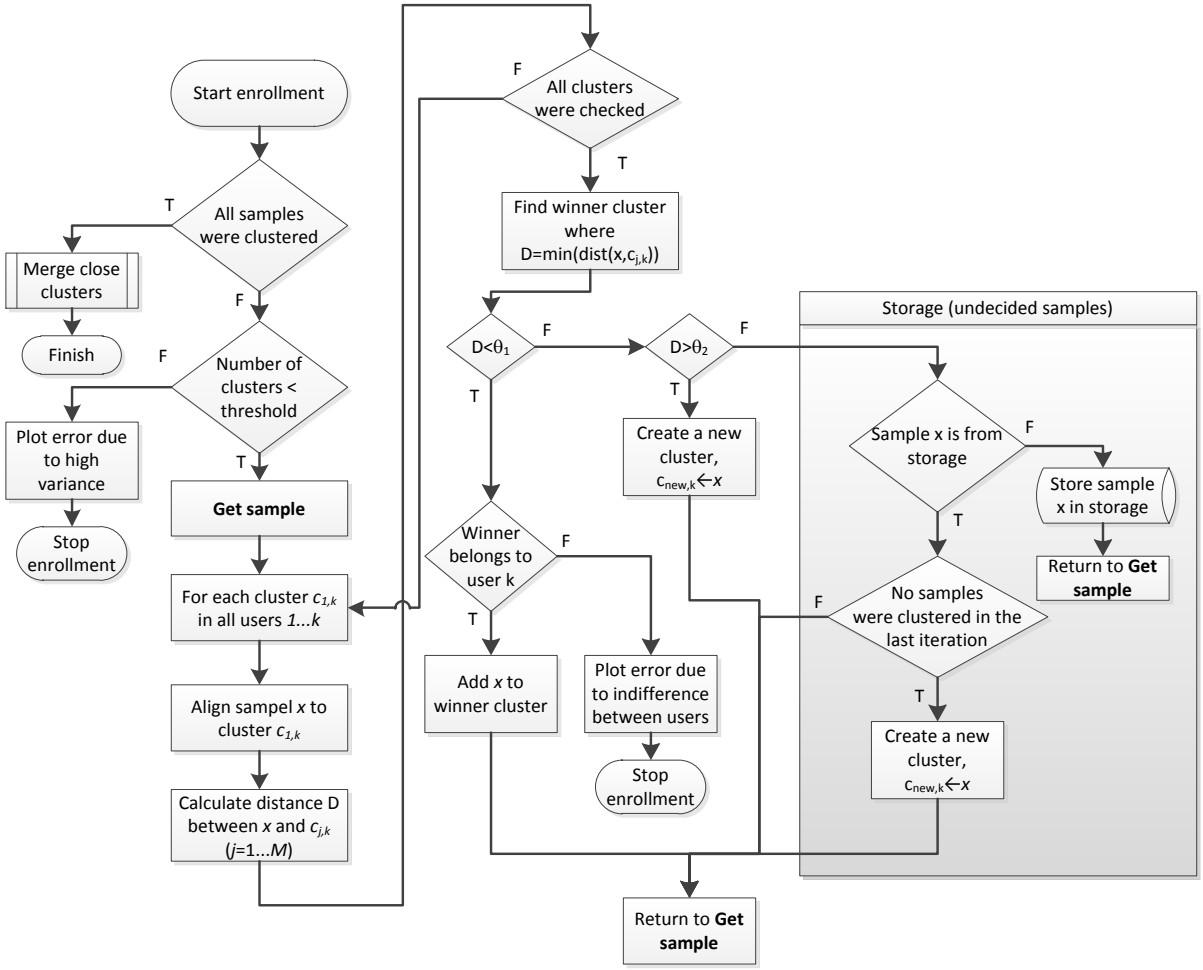

Figure 35 - A flowchart of the interactive enrollment method.

תקציר

מחקר זה עוסק בזיהוי ביומטרי באמצעות מאפייני תנועת יד. לצורך הזיהוי פותחה מערכת הקולטת תנועת יד, ומזהה את משתמש המבצע חתימת תנועה מול המצלמה. המערכת מאפשרת למשתמשים לבחור חתימת תנועה כרצונם, ובכך משפרת את האינטואיטיביות והנוחות בשימוש. מתוך מסלולי היד הנקלטים במערכת נלקחים מאפיינים מרחביים וזמניים. המשתמשים מאמנים את המערכת על ידי הקלטת מספר חתימות תנועה, והמערכת מזהה חתימת תנועה חדשה באמצעות השוואת המרחק מהחתימה החדשה לכל אחד מהמשתמשים הרשומים במערכת. מדד המרחק בין חתימה חדשה לבין המשתמשים הרשומים נלמד באמצעות Neighborhood Components Analysis. בנוסף, מוצעת שיטה לרישום אינטראקטיבי למערכת באמצעות אישכול. שיטה זו מודיעה למשתמש כאשר יש שוני גדול מדי בין דגימות במהלך האימון או כאשר החתימה הנבחרת דומה מדי לחתימה של משתמש רשום אחר. על מנת לבדוק את ביצועי המערכת בוצעו שלושה ניסויים: כל משתמש מבצע מחוות יד קבועה מראש (בלתי-תלוי), כל משתמש מבצע חתימה אישית (תלוי), ובדיקת היכולת של המערכת לזהות התחזות. עבור זיהוי משתמש אחד מתוך קבוצה של 3 עד 7 משתמשים, למערכת הבלתי-תלויה אחוז דיוק של בין 91 ל-77 בהתאמה. למערכת התלויה אחוז דיוק של 98 עד 92. עבור שלושה ניסיונות כניסה, אחוז הדחיה הנכונה ואחוז הקבלה הנכונה הם 81 ו 94 בהתאמה. המערכת המוצעת יכולה להשתלב בתוך ממשק זיהוי מחוות ידיים ולשמש למטרות אבטחה, התאמת תוכן, בקרה הורית ועוד.

מילות מפתח: ביומטריקה, למידת מדד מרחק, זיהוי מחוות ידיים, אשכול.

אוניברסיטת בן-גוריון בנגב

הפקולטה למדעי ההנדסה

המחלקה להנדסת תעשיה וניהול

זיהוי ביומטרי לפי חתימת תנועת היד

חיבור זה מהווה חלק מהדרישות לקבלת תואר מגיסטר בהנדסה

מאת: עמרי מנדלס

מנחים: ד״ר סיגל ברמן

פרופ׳ הלמן שטרן

חתימת המחבר ..... תאריך .....

אישור המנחה/ים ..... תאריך .....

Helman Stern.... תאריך *25.09.2011*

אישור יו״ר ועדת תואר שני מחלקתית ..... תאריך .....

אוניברסיטת בן-גוריון בנגב

הפקולטה למדעי ההנדסה

המחלקה להנדסת תעשיה וניהול

זיהוי ביומטרי לפי חתימת תנועת היד

חיבור זה מהווה חלק מהדרישות לקבלת תואר מגיסטר בהנדסה

מאת: עמרי מנדלס

אלול תשע״א ספטמבר 2011