POLITECNICO DI MILANO
Master of Science in Computer Engineering
Department of Electronics, Information and Bioengineering
(DEIB)

# Design and Development of a Kinect-Based Multi Purpose Gesture Interface for Fragile People

Supervisor: Prof. Sara Comai
Assistant Supervisor: Eng. Fabio Veronese

Master Graduation Thesis by:
Mahsa Teimourikia, Student ID 765565

Academic Year 2013/2014

POLITECNICO DI MILANO
Corso di Laurea Magistrale in Ingegneria Informatica
Dipartimento di Elettronica, Informazione e Bioingegneria
(DEIB)

# Progettazione e Sviluppo di una interfaccia di intepretazione dei gesti ad uso generale basata su Kinect e rivolta a persone fragili

Relatore: Prof. Sara Comai
Correlatore: Ing. Fabio Veronese

Tesi di Laurea di:
Mahsa Teimourikia, matricola 765565

Anno Accademico 2013/2014

*"While any software system introduces some kind of formalization of the world, HCI (like AI) deals with formalizations of human cognition and activity. These are the issues that have lay at the heart of philosophical debate for centuries. In some ways, it would be hard to imagine a more philosophical enterprise."*

Paul Dourish

# Abstract

With the advent of cheap image and depth sensors like Kinect, vision-based pose and gesture recognition have been increasingly used for different purposes. The use of keyboard and mouse and in general traditional input devices have always been an obstacle for older adults to interact with computers. Natural User Interfaces (NUI) such as gestures and hand poses without the need of touching the device seems to be more convenient for these users. However, for elderly or disabled people there are limitations on the movements which may differ from one person to another. In this thesis, the idea is to develop a gesture recognition system using Kinect that can get adapted to a specific person's abilities in performing poses and gestures for Human Computer Interaction.

Gesture recognition systems have been improved greatly in the recent years. Yet, they still have many shortcomings to handle special user needs for elderly or disabled user. The approaches proposed for gesture recognition systems usually ignore the specific hand postures or are able to recognize only a limited number of the postures in combination with the trajectories of the tracked hands. Furthermore, price, responsiveness, user adaptability, learnability, accuracy, and low mental load are the aspects that should be considered in the design of the system.

In this work, an approach for hand pose estimation and static gesture recognition is proposed that is flexible to recognize the user specific poses and gestures with minimum effort for training. The work considers, elderly limitations in performing the gesture and hand poses and proposes an approach to overcome some of these limitations. Skeletons of the hands are extracted as the features needed for hand pose estimation. Two algorithms are adopted and compared for finding the similarity between different hand poses. Last but not least, an HMM-based approach is introduced for static gesture recognition. The tests carried on have shown an average accuracy of 92.29% for pose estimation and 98.4% for gesture recognition.

# Sommario

Grazie alla diffusione di sensori integrati per la ricostruzione di scenari 3D a basso costo, come Kinect, il riconoscimento di gesti, del volto e della postura del corpo basato su visione artificiale trova campi d'utilizzo sempre più ampi. L'utilizzo di mouse, tastiera ed in generale di dispositivi di input tradizionali è da sempre un ostacolo nell'interazione tra persone fragili e computer. Le interfacce utente naturali, implicite, (Natural User Interfaces, NUI) non richiedono il contatto con un dispositivo sembrano essere più adatte per questo tipo di utenti; un esempio è il riconoscimento di un gesto e della posizione della mano per compiere una azione su di un computer. Inoltre persone disabili o anziane possono soffrire di limitazioni nei movimenti, differenti da persona a persona, che richiedono una adattabilità della interfaccia alle loro reali possibilità. Questo lavoro di tesi propone un sistema di riconoscimento di gesti e della posizione della mano basato su Kinect che può essere adattato alle reale capacità ed esigenza dell'utente; il sistema è pensato per essere una interfaccia HCI (Human Computer Interaction).

I sistemi di riconoscimento di gesti si sono molto evoluti negli ultimi anni anche se presentano ancora qualche limitazione nell'adattamento a reali bisogni di utenti disabili o anziani. Inoltre gli approcci proposti per il riconoscimento di gesti solitamente non tengono in conto della specifica posizione della mano, o sono in grado di riconoscere solamente un numero limitato di posizioni combinate, eventualmente, con la traiettoria della mano stessa. Nella fattispecie, di fronte alla progettazione di un sistema di questo tipo e necessario considerare gli aspetti riguardanti costo, reattività, accuratezza, adattabilità all'utente, facilità di apprendimento e spontaneità/intuitività. In questo lavoro viene proposto un approccio per la stima della posizione della mano e il riconoscimento di gesti statici in grado di adattarsi ai bisogni e alle capacità specifiche dell'utente con un impegno di addestramento minimo.

Il lavoro prende in considerazione le limitazioni che caratterizzano i gesti e le posizioni della mano e propone un approccio per affrontarne alcune. Dalle immagini delle mani si estrae un'approssimazione dello scheletro che viene poi utilizzata per estrarre le caratteristiche su cui si basa la stima della posizione. Sono stati presi in esame due algoritmi, confrontandone le prestazioni nel riconoscere la similarità tra diverse posizioni della mano. Infine, un approccio basato su HMM e stato introdotto per implementare il riconoscimento di gesti statici. I test hanno mostrato un'accuratezza del 92.29% per la stima della posizione e del 98.4% per il riconoscimento dei gesti.

# Acknowledgment

I would like to express my deepest appreciation to my supervisor professor: Prof. Sara Comai and also Prof. Fabio Salice, who have continually supported and helped me throughout this thesis and thank them for their patience, motivation, and enthusiasm. Moreover, I would like to express my gratitude to Prof. Matteo Matteucci for his advises for different parts of this thesis. I also want to thank my assistant supervisor Eng. Fabio Veronese. And special thanks for Eng. Hassan Saidinejad for closely following this work. In the end, I would like to thank my parents for the emotional and financial support that they provided me throughout my studies abroad.

# Contents

# List of Figures

# List of Tables

# List of Algorithms

# Chapter 1

# Introduction

Natural User Interfaces (NUI) have attracted considerable amount of research nowadays. With growing usage of computer systems in everyday life and the desire to embed them into the environment, the employment of traditional input devices (like keyboard and mouse) seems to be a bottleneck in efficient and intuitive Human Computer Interaction. NUIs show great promise to facilitate modern interactive computing [70]. Gesture-based interfaces as one of the most prominent NUIs are becoming as ubiquitous as currently dominant paradigms such as GUI and touch-based interfaces are.

More recently, devices like Microsoft Kinect, a low-cost Natural Interaction (NI) device with RGB and Depth sensors, have been used increasingly in Natural User Interfaces and Human Computer Interaction researches. Many approaches have been proposed for hand pose and gesture recognition using vision-based techniques. However, limited research has focused on vision-based Human Computer Interaction adapted to a specific user [78], which is of great importance for the elderly users. They usually have different pathologies for hand movement (e.g., hand shake or vibration), which limit them in performing some gestures. Also, pain when doing some kinds of movements can occur for the elderly. Therefore, an approach that can specifically targets older adults seems necessary to guarantee the effectiveness and ease of use.

## 1.1 Motivations and Problem Statement

A United Nations study [83], as well as European studies [31], convey that the demographics are experiencing an elderly population growth. In an aging world assisting elderly to live independently in their own environment and enhancing their quality of life has been an important concern. Bobeth et al. [16] in a study conclude that older adults acceptance rate toward gesture-based HCI are considerably high and the significant rating of enjoyment conveys that they find the gesture-based approach pleasant to use. Yet, considering the decrease in hand function with aging, especially after the age of 65 years [20], there is the necessity of development of a simple and convenient HCI technique for this kind of limited mobility. Moreover, forgetfulness and difficulty to adopt advanced and complex concepts, which are common in the elderly, introduce a concern to be regarded in an efficient HCI design for the usage of this ever-growing group of society.

Research and studies in geriatrics suggest that in elderly people, both men and women, degenerative changes in musculoskeletal, vascular and nervous systems lead to hand function degradation in terms of handgrip and finger-pinch strength, maintaining pinch force and posture, and dexterity of manual movements [65]. Reduced hand function results in limitations in hand maneuvers needed for a gesture-based human computer interaction. Hand tremor, coarse hand movement, and hand posture are the three limitation characteristics, which are common in the elderly [20]. Hand tremor is involuntary shaking of the hand. If the frequencies of some gestures in the gesture set are comparable to the frequency of the hand tremor, then it is necessary to think of a suitable mechanism to filter out such tremor. Coarse hand movement limitations refer to the coarse movements involving the arm of the person. This limitation could happen, for example, as a result of hand and elbow arthritis, which makes coarse movements painful and unpleasant. Finally, difficulty in moving fingers and making fine hand gestures belong to hand posture limitations. Once categories are identified, it is possible to design corresponding suitable gesture sets dealing with a specific limitation or a mix of them, and therefore an adaptive, effective, and usable approach for each older adult. Furthermore, when designing a gesture set for the elderly, the difficulty in adopting and memorizing complex gestures should also be taken into account. Therefore, simplified or known symbols can be used which can easily correspond to the set of actions in a specific application. This subsection describes the main issues and possible solutions for the three categories of limitations caused by hand function degradation and introduces the gesture sets adopted in our system to provide an easy to use gesture-based HCI for elderly users. These limitations can be categorized as follows:

- *Hand Tremor Limitation*: Involuntary shaking of the hand could be a barrier for a gesture-based interaction system. For example, in Essential Tremor, a progressive neurological disorder, a tremor with a frequency between 4 and 12 Hertz happens when force is exerted on the destination muscle. This could impede the effective interaction of the user if the frequency and range of the tremor is comparable with the timing and range of the defined gesture. The problem of trembling hand can be faced on two fronts: tremor filtering and gesture definition. The former is the low level solution to the problem and involves image processing techniques in order to filter out the tremor in the aforementioned frequency range and thus feeding a tremor-free input to the system. In this case, the higher levels of the system do not need to be aware of the problem. The latter, on the other hand, confronts the problem at a high semantic level. Gestures are defined with the a prior knowledge of the existence of hand tremors. Adopting this approach, high frequency, short-range movements can be removed from the set of possible gestures. For instance, a gesture involving the movement of the fingertip of the index finger back and forth, three times per second with a range of three centimeters for the sake of example, is excluded.

- *Coarse Hand Movement Limitation*: Hand and elbow arthritis and other similar disorders making the arm movement difficult, lead to coarse hand movement limitations. It is painful or unpleasant for the person to perform dynamic gestures in which the trajectory of the hand movement is a meaningful interaction with the system. In such a scenario, a gesture set richer in terms of hand posture gestures is desirable. The user interaction with the system should mainly exploit a static and fixed hand with different postures rather than a moving hand and dynamic trajectories.

- *Hand Posture Limitation*: Giving shape and a specific posture to fingers and the hand could be painful or difficult for some elderly people. Complex hand postures could thus be detrimental to the effective functionality of the system. For this sort of limitation, a gesture set with a few simple hand postures and a richer dynamic hand gestures is more desirable. For instance, the user might consider it easier to form the shape of a number in air using hand movement trajectory rather than showing the number with the fingers.

User interfaces for elderly are one of the topics studied by the Assistive Technology Group (ATG) [1] at Politecnico di Milano. This thesis is a part of these studies and proposes a gesture recognition approach that can get adapted to a specific user which is of great importance for elderly or disabled users with limitations in the hand movements. The proposed system can be trained to work with a gesture set that is the most appropriate for the target user. The training of the system should be minimal: since the system is going to be designed specifically for the special needs of one person, the training is limited to the data acquired from the same person and therefore, there is no need for huge trainings. Furthermore, price, responsiveness, user adaptability, learnability, accuracy, and low mental load are the aspects that are considered in the design of the system.

The research on gesture recognition has focused mainly on dynamic gestures that consider the trajectory of the hands [48, 51, 93]. In these kind of gestures the user is asked to make movements mostly involving the elbows and shoulders that can be tiring or painful or even impossible to do for some elderly or disabled users. Static gestures, that consist in moving the hands between different postures have had less attention during the recent years among researchers. Only a small number of approaches have successfully recognize gestures that include also different hand poses [21, 33]. However, only a limited number of hand poses are understandable for the gesture recognition system in these works, and the posture usually remains the same during the gesturing.

This thesis focuses on recognizing static hand gestures and postures. The approach introduced here is flexible enough to be able to train and use the system with minimal effort for variety of hand poses and gestures that meet the user's specific needs. The proposal of this work is grounded on the results of [16], which show that when a system design takes into account characteristics of special users such as the elderly, a group who are more resistant to the adoption of new technologies, the acceptance rate is high. When the ease of use is guaranteed and the complexity of learning a new technology is not present, older adults have a positive attitude toward using this technology in their everyday life. The proposed approach that adapts to the user's specific movements, aims at increasing the level of flexibility of the applications.

The results of this work show that the feature extraction method, for computing the skeletons of the hand, effectively respond in real-time and can handle the different variations of hand poses. Defining the new poses for hand pose estimation can be done by saving a snapshot of the hand pose in the bank of postures that are used for hand pose estimation. The proposed method for hand pose estimation, is able to recognize the postures in real-time with high accuracy. Also, an improvement on pose estimation is proposed to handle the hand trembling that is very common in the elderly. The proposed system supports also static gesture recognition, so that also postures that do not exist or are not estimated correctly by the pose estimator can be still recognized with high accuracy. In this way, the flexibility and adaptability of the system allow to select hand poses and gestures that are the most convenient for a specific user.

## 1.2 Structure of the Thesis

The structure of the thesis is as follows: in Chapter 2, state of the art and related works are reviewed. Tools, technologies, algorithms and methods used throughout this thesis are introduced and explained in Chapter 3. Chapter 4, details the approach for hand localization, segmentation, feature extraction, hand pose estimation, gesture recognition, and solutions for the elderly. In Chapter 5 the implemented application for testing the introduced approaches will be discussed. Later, in Chapter 6 the evaluation and testing of the proposed methods can be viewed. Finally, conclusion and future works are presented in Chapter 7.

# Chapter 2

# State of the Art

In this chapter state of the art and related works will be reviewed. Hand gesture recognition is used for natural user interactions that are easy to use and learn. Hand gestures can be captured through a variety of approaches such as usage of wearable sensors and/or considering vision-based approaches [51].

## 2.1 Gesture Recognition Using Wearable Sensors

An example of wearable sensors is represented by data gloves, which can provide accurate measurements of hand pose and movements. However, wearable sensors usually require extensive calibrations, are commonly costly while they limit the hand movement and are not very comfortable to be used in everyday life. Another example is the most recent wearable sensor in the time of writing this document called MYO armband which senses myoelectricity (the type of energy presented in the muscle tissue) [54], therefore it is highly responsive and fast, the price is reasonable and it offers an open-source API for developing applications. But on the other hand, as it connects to the devices using bluetooth, it is limited to the bluetooth integrated devices, and also introduces a problem when different devices working with the same gestures. Another problem can be the limitation on the number of the users which can use the armband as it can be used by one user at a time.

Mynatt et al. [52] use a wireless wearable device, called Gesture Pendant, that uses infrared illumination and a CCD (charge-coupled device) camera to recognize a simple set of gestures used for aging in place applications. Some other approaches [82, 47, 36] propose using an accelerometer sensor, markers, colored hand gloves, or wrist bands accompanied with a vision based approach to simplify the hand localization and tracking. Yet, using no intermediate device or marker is desirable for a Natural User Interface. In the following the vision-based approaches for hand pose and gesture recognition will be detailed.

## 2.2 Vision-Based Gesture Recognition

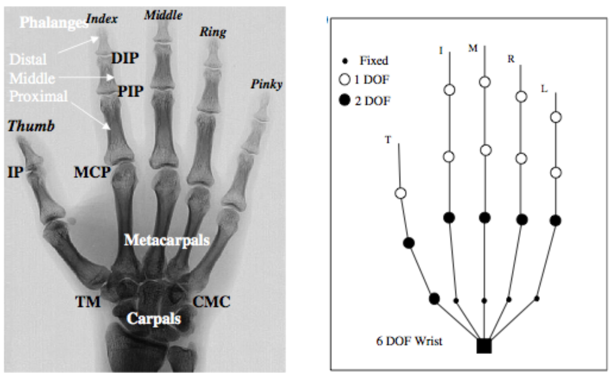
To avoid the use of any wearable or mechanical device, vision-based gesture recognition can be used. The inputs for these approaches are provided by RGB and/or depth images extracted from video cameras, Time of Flight (ToF) cameras, and depth sensors such as Microsoft Kinect and Asus Xtion. In Gesture-based Human Computer Interaction (HCI) real-time video-streaming of the gestures and efficient and real-time responsive algorithms to capture and process gestures [90] is needed. Vision-based gesture recognition techniques can be divided into two broad categories [30]: Model based approaches that focus on taking advantage of a 3D or 2D model of the hand for hand pose and gesture recognition and appearance-based gesture recognition, which extracts the features of the visual data for gesture recognition. first approach provides a continuum of solutions but are computationally costly and depend on the availability of a wealth of visual information, typically provided by a multicamera system. Appearance-based methods are associated with much less computational cost and hardware complexity but they recognize a discrete number of hand poses that correspond typically to the method training set [57].

### 2.2.1 Model-based Approach

According to [25], kinematic hand model that represents the motion of hand skeleton can be introduced as the basis of all hand models. This model is redundant since it does not show the correlation between the joints. Figure 2.1(a) shows the studied anatomy of the hand which is modeled as shown in figure 2.1.

Since the active motion of the hand is not considered in kinematic hand modeling, this model can be complemented by natural hand motion constraints in a closed form [25]. However, because the complex structure of the hands does not allow all the constraints to be expressed as closed form, learning-based techniques might be used to model the natural motions of the hands more accurately.

Moreover, the shape of the hand can be modeled. Hand shape models have both articulated and elastic components, which introduce computa-



(a) (b)

*Figure 2.1: Skeletal hand model (a) Anatomy of hand skeleton, (b) Kinematic model of the hand, (original image in [25])*

tional complexity if the model is profound. This is because in many studies, the model should be projected several times on the input data to being able to extract the features that are needed for hand pose estimation. For this reason less complex models can be adopted to model the hand shape. Figure 2.2 shows three different models proposed by [77, 57, 94, 18]

Furthermore, the accuracy of the model-based approaches can be greatly affected by the user specific shape parameters of the hands like length and width of links. These parameters can vary over a wide range, and a user-specific calibration is needed to overcome this problem in the hand model [25]. This calibration that includes kinematic fitting to estimate the link lengths (location of the joints) is a difficult task.

Pavlovic [60] has done an extensive review on variety of approaches on hand modeling and gesture recognition using the model-based category of techniques. These approaches usually suffer from high complexity in implementation and cannot be used in live applications, which are the essential property of HCIs [90]. Appearance-based approaches, on the other hand, can respond faster in real-time and are less computationally expensive. Yet, they might provide less accuracy compared to model-based approaches [90]. However, the research in this field is promising more accurate and efficient approaches [57]. Appearance-based approaches use RGB or depth data or both as the input.

### 2.2.2 Appearance-based Hand Pose Estimation

For appearance-based gesture recognition with considering the pose of the hands a method to extract the hand pose features is needed. In the studies,

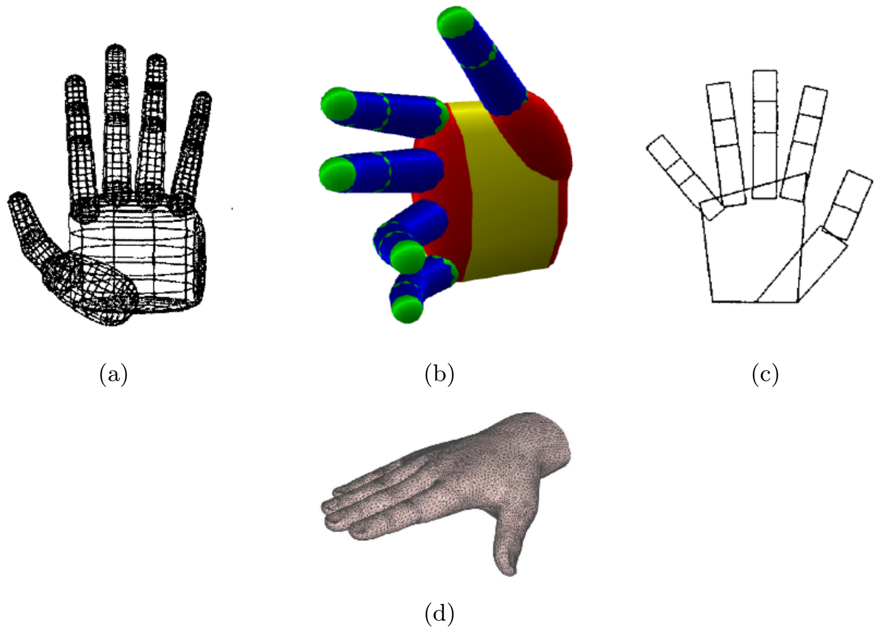


Figure 2.2: Hand shape model (a) Quadrics-based hand model [77], (b) 3D hand model using the RGB and Depth data [57], (c) cardboard hand model [94] (d) Natural hand model polygonal surface[18]

a wide variety of approaches to obtain the features have been adopted such as: Fourier descriptors, moments, optical flow and etc. The interesting features are usually the position and the orientation of the hand, fingertip locations, and shape description of the hand pose. Position of the hand can be described by the position of center of the palm [78]. Center of the palm can be also useful for detecting various types of hand poses. Various methods are proposed for obtaining the center of palm. Sato et al. [68] defines the center of the hand as the point that its distance to the closest region boundary is maximum. Kinect SDK and Nite2 skeletal tracking also position the center of the palm while tracking different body parts.

For obtaining orientation features, [29] proposes using orientation histograms. Nickel and Stiefelhagen [53] identify the orientation based on the largest eigen value of the covariance matrix of the 3D points around the center of the hand. The orientation features are useful especially in the applications in which the orientation is critical such as recognizing pointing gestures and navigation in virtual reality.

Another feature that have been widely used in hand pose recognition and virtual multitouch interfaces is the position of the fingertips. Oka et al. [58] consider a search window on the hand, and use a cylinder with a hemispherical to approximate the finger shape and search for the finger tips based on its geometrical features.

Yang et al. [96] use the ratio of width and height of the hand region for shape description and adopt this information to recognize closed and open hand postures. Triesch et al. [81] employ elastic graph matching method to recognize static hand postures. In this approach the hand postures are described by labeled graphs in which nodes are labeled by local image descriptors based on a wavelet transform with complex Gabor-based kernels, and edges are labeled by a distance vector. Fang et al. [27] calculate image convolution using Gaussian derivatives approximated by integral image for feature detection. Then, to represent the hand gestures multi-scale geometric descriptor at feature points are obtained. And finally, the geometric configuration of the gestures are used in gesture recognition. Chen et al. [21] employ Fourier descriptor (FD) to extract the external boundaries of a hand shape.

### 2.2.3 Usage of Color Images in Vision-based Approaches

Color images of one or more cameras can be used for hand gesture recognition. However color images introduce some problems: First, locating and segmentation of the hand from the noisy and complex background can be challenging, especially when there are other skin colored objects in the scene and in case of occlusions [48]. Also, the hand segmentation can be noisy with illumination changes [25]. Furthermore, with respect to the real-time performance which is crucial for some applications, e.g. HCI, extracting sophisticated features from the image to have a more accurate recognition can greatly increase the processing time [93].

### 2.2.4 Usage of Depth Stream in Vision-Based approaches

Depth images can be either captured from depth cameras such as Microsoft Kinect Depth Sensor or be extracted from stereo video cameras. They can function in several situations in which video cameras cannot, such as low or variable lighting conditions, and in environments with the presence of other skin-colored objects (like face) [78]. As the time of writing this document, with the increasing accuracy in the recent depth-sensor outputs such as the new generation Kinect depth sensor which is going to be introduced in the market next year, can result in more accurate data that is needed for gesture recognition.

## 2.3 Hand Localization

One of the essential components of hand gesture recognition is hand localization. Hand localization pertains to the positioning of the hand and to its segmentation from the background. In other words, understanding which pixels of the input image belong to the hand [78]. There are several methods and approaches proposed and implemented for both. Suarez and Murphy [78] review recent methods and techniques using depth images and/or RGB data to offer an efficient hand gesture recognition in variety of applications. Yet, there are pros and cons in using each approach.

It seems reasonable to use both RGB and depth data to have a more accurate and efficient gesture recognition. Tang [80] takes advantage of a full body tracker to localize the hands using Kinect's RGB and depth inputs. For accurate hand segmentation, they employ a skin color detector on RGB data together with color balancing to compensate for the sensitivity of this method to illumination changes and they accompany this approach with depth thresholding on depth data to exclude the non-hand pixels detected by mistake. However, this approach still suffers from effects of background clutter and illumination changes and the usage of skin-color detector to be performed on the whole image is not efficient and can be time consuming.

## 2.4 Real-time Gesture Recognition and Adaptation

For the use of gesture-based HCI, the accuracy, effectiveness, real-time performance, and adaptation to the user are essential aspects, which should be kept into consideration. HMMs (Hidden Markov Models) are one of the prominent methods used for adaptive gesture recognition as the gestures are continuous motion in sequential time series [49].

Starner et al. [76] adopt HMMs for American Sign Language (ASL) recognition. They have used color cameras in two positions, one on the desk and in front of the user, and second on a cap which is worn by the user. They confirm 92 and 94 percent word accuracy for each approach respectively. For recognizing the sign language they have ignored the detailed shape and pose of the hand and have only considered, coarse hand pose, orientation, and the trajectory of the gesture through time, and used these information as the inputs of HMM. A four-state HMM is used in this approach. However, due to the lack of information on the hand shape, in the evaluation they did not insert the sign language words that are heavily based on the position of the fingers.

In another study by Chen et al. [21] HMM is adopted for continuous gesture recognition. The background is assumed to be stationary to simplify the process of background subtraction. Fourier Descriptor (FD) is employed to extract the spatial features and for getting the temporal features motion analysis is adopted. The feature vector is the combination of spatial and temporal features which is used by HMM for gesture recognition. They have concluded 90% of accuracy in recognition of 20 different gestures.

Zhu and Pun [98] propose a real-time hand gesture recognition system using Kinect depth data and the PrimeSense NITE middleware's tracking algorithm for extracting the trajectory data sequence of the hand movements. They assume the hands are kept relatively static for recognizing the start or end of each gesture. As for real-time gesture recognition, they have adopted Hidden Markov Models (HMMs) for gesture recognition: the system is trained online and at runtime, it gets updated with the user feedback. They claim 92% accuracy in gesture recognition rate with limited updating on a test data for ten hand sign digit gestures. Yet, eliminating the offline training of the gesture recognizer will increase the online adaptation process and will result in inaccuracies at the beginning of the usage of the system. In case of elderly users, it is better for a new technology to perform well when used by them for the first time to encourage the positive attitude to continue the usage in everyday life.

Binh et al. [14] adopt blob analysis and the Kalman filter approach for developing their own hand tracking system on color images. They employ a pseudo two dimension hidden Markov models (P2-DHMMs) approach for gesture recognition using the shape and motion information of the hand. Wilson and Bobic [92] introduce an online adaptive gesture recognition system using appearance-based models and HMMs. They assume that the representation of temporal structure of a specific gesture is present in the system and is able to be combined with the real-time information. Licsár and Szirányi [43] propose an interactive training method for a real-time adaptive gesture recognition for augmented reality applications using a camera-projector system. They have adopted nearest neighbor rule and modified Fourier descriptors (MFD) for hand pose classification. However, they do not store any gesture classes for the users and the gestures will be overwritten by the trained data of another user. In [43] a gesture set is not considered for the users and unsupervised learning have been adopted to classify the gestures, and supervised learning to update the errors in the gestures on the go.

## 2.5 Use of Gesture-based HCI for Elderly

Hand functionality decreases evidently after the age of 65 in both sexes [20]. Therefore, for the purpose of gesture-based HCI for elderly, common approaches may not meet their needs and can introduce difficulties in the usage of the system in everyday life. Developing HCI for the elderly have been the focus of researchers for many years. Rahman et al. [63] employed hand gloves with infra-red emitter and an IR camera to capture user hand gestures and depict its application in a smart home environment. Bobeth et al [16] evaluate the elderly performance and acceptance of a gesture-based approach on "TV Menu Control". Yuan et al. [97] introduced a gesture-based approach on Multi-Touch pads for users with disabilities and limitations in range and function of their hands. Häikiö et al. [32] propose a touch-based user interface for the elderly and examine the suitability of their approach.

Yet, the current gesture-based approaches for the use of elderly do not seem to be sufficient enough and some important aspects still need to be taken into consideration. The adoption of wearable devices for gesture recognition is inconvenient for everyday use. Also, considering a fixed and limited set of gestures that have to be learned by the user, they are not efficient in the case of elderly who might have difficulty learning or performing certain kinds of gestures. In the sequel, we introduce our approach for designing an efficient, adaptive and intuitive geture-based HCI for elderly users.

# Chapter 3

# Tools and Algorithms

In this chapter the tools, technologies and algorithms used throughout this thesis are introduced and explained including the technology that is used to capture the data form the user, and algorithms that are adopted for feature extraction, pose estimation and gesture recognition.

## 3.1 Microsoft Kinect

Microsoft Kinect that is originally made for XBox 360 video games and was released on November 2010 is a motion sensing input device which makes available full skeletal tracking, the RGB and Depth image streams and more. In February 2012 Microsoft released another version of Kinect for windows. Allowing programmers to develop applications using the capabilities offered by Kinect.

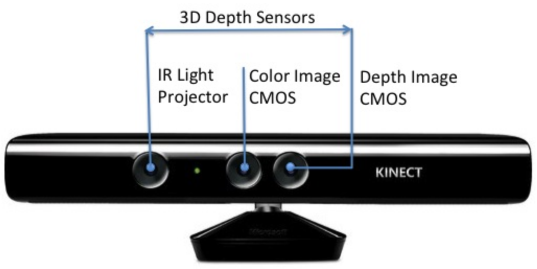


*Figure 3.1: Microsoft Kinect camera sensors*

As it is depicted in Figure 3.1 Microsoft Kinect includes an RGB camera, an infrared (IR) emitter and IR depth sensor. The depth sensor reads the beams that are emitted from the IR emitter and reflected back to the sensor. Using an internal processor by comparing the spacing between the beams the distance between the objects in the scene and the sensor can be calculated, and in this way the depth information will be produced [78]. Moreover, Kinect can capture sound and locate the sound source and the direction of the audio wave using a multi-array microphone. Last but not least, Kinect includes an accelerometer that is used to determine the current orientation of the Kinect.

The specification of the Kinect sensors are summarized in Table 3.1. The practical distance between Kinect and the performer is 800mm to 3000mm. And our approach best results in 800mm to 1500mm distance for more accuracy and efficiency. We employ Kinect RGB images together with 3D depth stream, both with the resolution of 680x480 at 30 frames per second with the depth sensitivity of 2048 levels.

| Kinect Specifications | |
|---|---|
| Resolution of color image | 640x480 pix |
| Resolution of depth image | 640x480 pix |
| Resolution (z axis) | 10mm @ 2000mm |
| Frame rate | 30 fps |
| Horizontal angle | 43 degree |
| Vertical angle | 57 degree |

*Table 3.1: Specification of Kinect Sensor*

## 3.2 OpenNI and NiTE

OpenNI [5] is an open-source SDK used for development of application and libraries for 3D sensors. OpenNI can be also used as an SDK for Microsoft Kinect. In this thesis OpenNI have been used because of its effectiveness and availability of variety of Middlewares and libraries that was needed for development of this work. NiTE [4] is one of these open-source middlewares that can be adopted for body tracking.

## 3.3 Voronoi Diagrams

For solving the geometrical problems in computational geometry many algorithms have been developed. Specifically, computer science areas such as computer graphics and pattern recognition introduce problems that are inherently geometrical. One of the algorithms used in this regard is Voronoi Diagrams [9]. Voronoi Diagrams (VD) are used to extract the regions around the given points (seeds) with respect to the nearest neighbor algorithm upon which the point in the plane are selected if there are closest to the seed point employing a distance measure such as Euclidean Distance. In simpler words, Voronoi Diagrams divide the space in number of regions based on the seed points which are defined beforehand, in a way that all points in a region are the closest to the corresponding seed point. These regions are called Voronoi cells [59].

### 3.3.1 Basic Properties of Voronoi Diagrams

In this section [61, 24, 9, 59] have been used mostly as the resource to introduce the basics of Voronoi Diagram (VD) in geometric problems. For having a general definition of Voronoi Diagrams lets assume that $S$ is a set of $n$ points (seeds or sites) in a plane. If we consider two sites $p, q \in S$, then $p$ is dominant over $q$ if the subset of the plane is at least as close to $p$ as it is to $q$. This can be shown as follows:

$$dom(p, q) = \left\{x \in \mathbb{R}^2 | \delta(x, p) \leqslant \delta(x, q)\right\} \tag{3.1}$$

in which $\delta$ denotes the Euclidean Distance of the two points. The dominance is defined to separate the regions in which the points are closer to a seed point (the dominant one) than the other. Formally we can define a region around $p$ as follows:

$$reg(p) = \bigcap_{q \in S - \{p\}} dom(p, q) \tag{3.2}$$

Which simply means that a region around seed point $p$ is defined to be the point of the space which are closest to $p$. To set an example, figure 3.2 depicts the regions calculated by Voronoi Diagrams considering eight seed points. As it can be seen in the figure 3.2 the regions form a polygonal partitions of the plane. This partition is called *Voronoi Diagram* and is shown as $V(S)$ in which is a finite set of seed points as defined before.

### 3.3.2 The Complexity and Performance of Voronoi Diagrams

Voronoi Diagrams can be considered as a planner graph with minimum vertex degree 3 and n regions. Each edge $e$ is connected to two vertexes and each vertex $v$ has at least three edges. From this we can include that $2e \geq 3v$ and according to the Euler's relation $n + v - e \geq 2$ it can be implied that $e \leq 3n - 6$ and $v \leq 2n - 4$. Therefore, there are less than $3n$ edges and each of them belongs to exactly two of the $n$ regions. This shows that the Voronoi Diagram has linear size complexity and this is one of the reasons

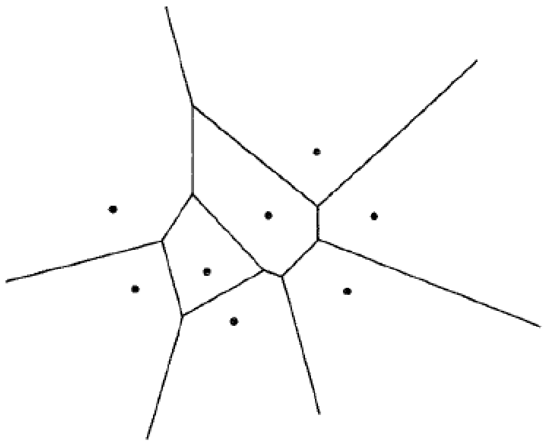


*Figure 3.2: Voronoi Diagram (VD) for eight seed points in the plane, (original figure in [9])*

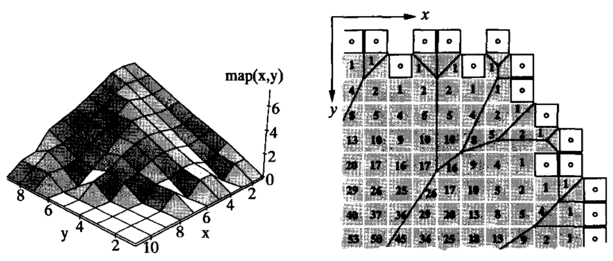
that it is widely used. Another reason is that Voronoi Diagram includes explicitly all the proximity information of $S$ with computation efficiency.

### 3.3.3 Skeletonization

As defined by Lieutier [44], a skeleton is locus of maximal open disks (in 2D) as it is shown in figure 3.4(a) and the process of obtaining the skeleton is called skeletonization. In other words, skeleton is a graph that summarizes the shape of an object and can be used as an efficient shape descriptor for object recognition and image analysis. The concept of skeletonization or thinning is a process in which a 2D object will be transformed into a 1D line representation [55]. The notion of medial axis (MA) or skeleton first was introduced by Blum [15]. An example of MA is depicted in figure 3.4(b). The implementation of the Blum's MA in the discrete world is complicated and inefficient, and the proposed methods that implement Blum's approach usually fail to fulfill the connectivity and the Euclidean metrics [74, 56].

According to [22, 56] skeletonization algorithms can be categorized as follows in four groups:

- *Grassfire simulation*
  The procedure in which simulates the *grassfire* as setting *fire* to the borders of the image. It is possible then to extract the skeleton in the region that the fire meets [41]. The algorithms rarely have implemented the actual *prairie fire* [56], instead, algorithms such as the one proposed in [41] are implemented relying on the distance surfaces and are based on active contour model.



(a) (b)

*Figure 3.3: (a) 3D view of the discrete distance map, (b) Voronoi Diagram on discrete distance map of a set of boundary points, with the Euclidean distance to the next boundary point. (original figure in [56])*

- *Analytical computation of the medial axis*
  These algorithms attempt to directly calculate the skeleton of an object using its polygonal approximation. However, the choice and the quality of the method used for approximating the polygonal shape of the object greatly influence the accuracy of the extracted skeleton [56].

- *Topological thinning*
  In *Topological Thinning*, the algorithm tests the object pixels and deletes them subsequently until their removal does not change the topology of the thinned shape. Topological Thinning guarantees the skeletons to be connected. However, there is loss of Euclidean metrics [22].

- *Medial axis extraction from a distance map*
  This approach relies on computing the distance map (DM) as it is shown in figure 3.3. The computation of the exact DM based on Euclidean distance is complex and inefficient [56]. Therefore, many algorithms suggest using other metrics such as *chessboard* which introduce great deviation from the actual Euclidean metric. Yet, the calculation of the distance map is closely related to the computation of the Voronoi polygons [23]. And mapping Voronoi Diagram into the rectangular grid as is shown in figure 3.3(b) is equivalent of calculating the distance map.

### The Voronoi Medial Axis

Using the Voronoi Diagrams, it is possible to extract characterizing elements from a given site pattern [9]. As studied in [8] finding the exact skeleton of an object which can be very complex, is solved for few objects. In [39] Lee shows that the exact skeleton of an polygon can be found by the use of generalized Voronoi Diagrams on its boundary. Also, in continuous methods in which the object is defined by set of points that are sampled on its boundary, Voronoi graph can be employed to approximate the object's skeleton with reasonable computation costs [17].

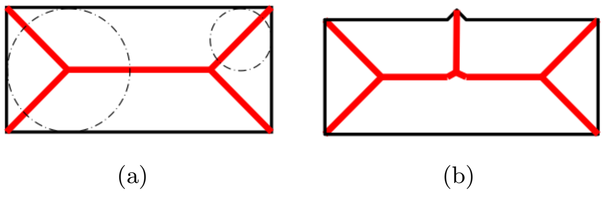


*Figure 3.4: (a) Maximal balls and the corresponding skeleton, (b) Medial axes of a polygonal shape (original figure in [22])*

By employing VD of the boundary line segments, it is possible to obtain the medial axis of polygonal shapes [50, 39] as shown in figure 3.5. Voronoi Medial Axes (VMA) of polygonal shapes include segments of straight lines and parabola. According to [26] all VD points are also Medial axis points and hence it is a center of a maximal ball.

A major disadvantage of using this method to extract the medial axes as a representation of an object is its high sensitivity to the small changes to the boundary [22]. Figure 3.4(b) represents how a tiny change in the boundary can affect the extracted skeleton. This is because, according to the definition of the medial axes any convex locus on the boundary is represented by an additional skeleton branch. And commonly, the polygonal approximation approaches are not robust and do not fulfill the invariance requirements. Also, with considering the requirement to produce a smooth skeleton to represent the shape, numerous line segments are needed which do not necessarily carry useful information for shape representation. This problem can be solved by pruning the obtained skeleton graphs to extract only the useful information.

A pruning method should have following characteristics to be acceptable [71, 10]:

- It should preserve the topology
- It should be a continuous method
- It should preserve the significant information
- It should be rotation and translation invariant

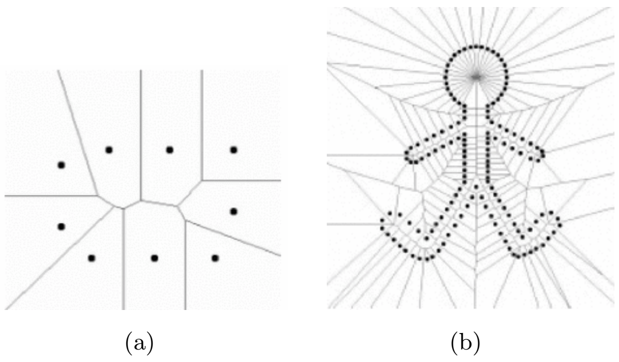


Figure 3.5: WMA obtained by calculating Voronoi diagram on the points sampled from the boundary of the shape (original figure in [26])

- it should include the centers of maximal disks which can be employed for reconstructing the object

Figure 3.6 depicts the original Voronoi medial axes extracted from boundary points, and then the results of a unacceptable pruning method which can cause incontinuities in the skeleton and change the topology compared with the result of a good pruning algorithm that pertains all the requirements mentioned above.

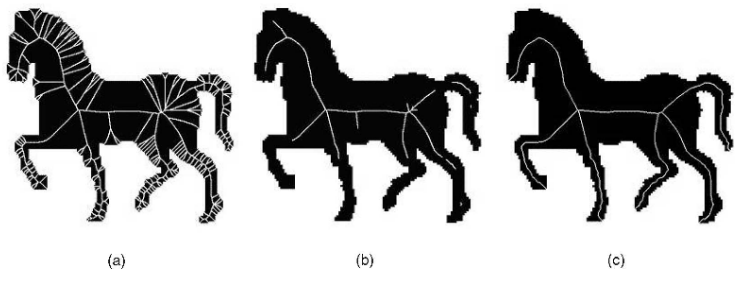


Figure 3.6: (a) Voronoi skeleton containing lots of redundant branches (b) The result of unacceptable pruning method (c) The result of a good pruning method (original figure in [10])

After applying the pruning to the obtained Voronoi skeleton, a good representation of the specified object can be extracted and used for further analysis.

The approach to extract the shape skeleton can be summarized as follows:

- Obtaining the planner shape

- Polygonal approximation
- Obtaining the Voronoi diagram on boundary points
- Pruning the VMA

## 3.4 Hand Pose Estimation

For hand pose estimation using the skeleton graphs of the hands an algorithm should be used for comparing different skeletons. Two algorithms have been used for this purpose in this thesis that are explained in the following sections.

### 3.4.1 Dynamic Time Warping

Dynamic Time Warping (DTW) [13] is a algorithms for comparing time series. By sorting the points in the skeleton this algorithm can be used to compare the two skeletons. This algorithm is explained as follows:

$B$ as similarity coefficient of two skeletons is adopted. Total distance routes are used, that have to pass the vertices of one skeleton at its transformation to an accurate hit on the vertices of the other skeleton. The matched vertices are determined by the DTW algorithm. $U$ and $V$ are denoted as the set of vertices, respectively, of the first and second skeleton, where $|V| = m, |U| = n$. $A \in R^{m \times n}$ is the distance matrix of between vertices $V$ and $U$ in the Euclidean metric:

$$a_{i,j} = d(u_i(x, y), v_i(x, y)) = (u_{ix} - v_{jx})^2 + (u_{iy} - vjy)^2 \tag{3.3}$$

The next step is to find paths in the matrix $A \in R^{m \times n}$, starting from element $a_{1 \times 1}$ and reaching the element $a_{m \times n}$, for which the sum of the elements in this path is minimal. To solve the problem in polynomial time dynamic programming algorithm can be adopted. For the matrix $A \in R^{m \times n}$ a new matrix $B \in R^{m+1 \times n+1}$ is calculated. Element $b_{1,1}$ is set to 0, and the other elements of the first row and first column of the matrix $B$ will be set to the value $\infty$. The remaining elements of the matrix is calculated as follows:

$$b_{i,j} = a_{i,j} + min \{b_{i-1,j}, b_{i,j-1}, b_{i-1,j-1}\} \tag{3.4}$$

$B$ as the coefficient of similarity of two skeletons gets the value of $b_{m+1,n+1}$ element. Note that how the order of the skeleton vertices used for the construction of the matrix affects the similarity factor. For this reason, the vertices of the skeleton are initially sorted. The vertices of the skeleton are the 2D points. For sorting the points they are sorted first upon values in X coordinates and then with values of Y coordinates.

### 3.4.2 Optimal Subsequence Bijection

Optimal Subsequence Bijection (OSB) is an algorithm defined by Latecki et al. [38] for elastic matching of sequence of real numbers. Since the query and the target sequences can contain outliers and be noisy, this method have been shown to have a more robust matching performance in many real-life data compared with some similar algorithms such as DTW (Dynamic Time Warping) due to their sensitivity to outliers [38]. OSB algorithm is defined as follows: consider two sequences $a = (a_1, a_2, ..., a_m)$ and $b = (b_1, b_2, ..., b_n)$ while $m$ and $n$ can be unequal. The goal of this algorithm is to find subsequence $a'$ from $a$ and subcequence $b'$ from $b$ in a way that $a'$ best matches $b'$. OSB will output a one-to-one correspondence (bijection) between the remaining elements after eliminating the outliers.

OSB skips the outliers and considers a penalty for doing so. This is for the reason that skipping too many outliers can reduce the accuracy as dissimilar sequences can be matched. It is assumed that the distance function $d(a_i, b_j)$ in which $i \in \{1, 2, ...m\}$ and $j \in \{1, 2, ..., n\}$ is defined to compute the dissimilarity between the elements $a_i$ and $b_j$ in before mentioned sequences. OSB does not make any assumptions on the distance function and any method can be used to define the dissimilarity or distance of two elements. The optimal correspondence between two sequences $a$ and $b$ can be found with a shortest path algorithm on the Directed Acyclic Graph (DAG). The nodes in DAG are defined as pairs $(i, j) \in \{1, 2, ...m\} \times \{1, 2, ..., n\}$ and the edges are weighted by $w$ as shown below:

$$w((i,j),(k,l)) = \begin{cases} \sqrt{(k-i-1)^2 + (l-j-1)^2}.C + d(a_k, b_l) & \text{if } i < k \wedge j < l \\ \infty & \text{otherwise} \end{cases} \tag{3.5}$$

Equation 3.5 computes the Euclidean distance between two nodes $(i, j)$ and $(k, l)$ multiplied by jump cost $C$ which is the penalty for skipping an outlier plus the dissimilarity measure of the elements $a_k$ and $b_l$. As mentioned before, OSB allows penalized jumps by considering a penalty. Assuming that $b \in B$ in which $B$ is a set of all target elements that are going to be compared with $a$. First, the query sequence $a$ is compared with $b \in B$ and we define:

$$C(a,b) = \underset{i}{mean}(\underset{j}{min}(d(a_i, b_j))) + \underset{i}{std}(\underset{j}{min}(d(a_i, b_j))) \quad (3.6)$$

In equation 3.6 the closest element $b_j$ to $a_i$ is found. And the mean plus standard deviation of the distance to the closest element is considered. In this way if we assume that the two sequences $a$ and $b$ are similar except for one outlier $a_k$ the $C(a,b)$ will be smaller than the distance between the closest element $b_j$ to $a_k$ and therefore, $a_k$ will be excluded from the matching sequence with a relatively small penalty. To have a fair comparison of $a$ with all sequences $b \in B$ we define a fix jump cost as follows:

$$C(a) = mean\{C(a,b) : b \in B\} \quad (3.7)$$

## 3.5 Hidden Markov Models

The observations from real-world processes can be characterized as signals which can be discrete or continuous in nature, it can be pure or contain noise, and the signal source can be stationary or non-stationary. The goal is to characterize such real-world signals as signal models which can be adopted very efficiently in prediction, recognition and identification systems. Deterministic or statistical models can be used to provide the signal model. Hidden Markov Models (HMMs) are statistical models which are related to Finite State Machines (FSMs) and are increasingly used in applications such as speech recognition, hand writing and also gesture recognition because of their adaptability and versatility [42]. They are also referred to as Markov sources or probabilistic functions of Markov chains. Statistical models like HMMs assume that the signal can be well characterized as a parametric random process and the stochastic process can be estimated efficiently and precisely [62].

This section will describe basic Markov chains and then the extension to HMMs. Through out this section [62] is used as a main resource.

### 3.5.1 Markov Chains

A Markov Chain, also called Discrete Markov Process can be described as set of $N$ distinct states, $S_1, S_2, ..., S_N$, with state transitions to other states or back to the same state, according to a set of probabilities associated with the state at regular intervals. The time instances in which the transitions occur will be denoted as $t = 1, 2, ...$, whereas $q_t$ will refer to the actual state at time $t$. In the stochastic process provided by Markov Chains each output of the process is a set of states at each instance of time and each state corresponds to an observable event. This can cause an important restriction on Markov Chains while the probability of a transition to an specific state depends only on the current state and not its predecessor states. The state transition probabilities matrix $A$ is defined as

$$a_{ij} = P\left[q_{t+1} = S_j | q_t = S_i\right], 1 \leqslant i, j \leqslant N \tag{3.8}$$

While conditions below should be satisfied

$$a_{ij} \geqslant 0 \tag{3.9}$$

$$\sum_{j=1}^{N} a_{ij} = 1. \tag{3.10}$$

Relying on observable events is one of the strong limitations of Markov chains. In order to be applied to problems such as recognition of speech, handwriting or gestures a less restrictive model is needed.

### 3.5.2 Extending Markov Chains

In HMMs, instead of producing the same fixed event each time, one out of a set of observable events will be outputted according to a probabilistic function associated with that state. Thus, the current state is only hinted by an observation which makes the process of state transition hidden. When modeling an HMM, the $N$ observation events will correspond to $N$ states. The occurrence of initial observation event marks the initial state probability $\pi_i$ which is defined as follows

$$\pi_i = P\left[q_1 = S_i\right], 1 \leqslant i \leqslant N \tag{3.11}$$

Occurrence of a new observation corresponds to the state transition probability $a_{ij}$ as defined in equation (3.8). While in HMMs, unlike the Markov chains, the $M$ observation symbols are not strictly linked to a specific state, they are considered separately and are shown as $V = v_1, v_2, ..., v_M$ where $M$ denotes the number of observation symbols. Additionally, observation symbol probability distribution in each state is defined as $B = b_j(k)$, where

$$b_j(k) = P[v_k \text{at} t | q_t = S_j], 1 \leqslant i \leqslant N and 1 \leqslant k \leqslant M \tag{3.12}$$

Finally an observation sequence can be defined as

$$O = O_1 O_2 ... O_T \tag{3.13}$$

where each $O_t (t = 1, ..., T)$ is a symbol from $V$. The observation sequences are generated as follows

1. Choose the initial state $q_1 = S_i$ using the initial state distribution $\pi$ defined at equation 3.11.
2. Set t = 1.
3. Choose the current observation symbol $O_t = v_k$ using the symbol probability distribution $b_i(k)$ in state $S_i$.
4. Use the state transition probability distribution $a_{ij}$ of state $q_t = S_i$ to get the new state $q_{t+1} = S_j$
5. Set $t = t + 1$ and return to step 3 if $t < T$, otherwise, terminate.

For convenience, a HMM can be denoted as a compact notation

$$\lambda = (A, B, \pi) \tag{3.14}$$

In which $A, B, and\pi$ are specification of three probability measures that are required in the specification of HMM together with $N$ and $M$ model parameters and specification of the observation symbols.

### 3.5.3 The Three Basic Problems for HMMs

Considering the notation we defined in equation 3.14 we can specify three interest points that should be solved for real world applications.

#### Problem 1: Computing the probability of observation sequence of the given model

This can be considered as an evaluation problem where we should compute the probability that the observed sequence of observation was produced by the model $P(O|\lambda)$, while we have the model $\lambda$ and the observation sequence $O = O_1O_2...O_T$. This problem can be viewed as an scoring problem in which we want to choose one of the several models that best matches the observation. An efficient way to solve this problem is to use *Forward-Backward Procedure* [42] where it defines the forward variable $\alpha_t(j)$ as the probability of a partial observation sequence $O_1O_2...O_t$ up to $t < T$ with the model being in state $S_i$ at time $t$,

$$\alpha_t(i) = P(O_1O_2...O_t, q_t = S_i|\lambda). \tag{3.15}$$

The problem now can be solved inductively through

1. Initialization,
$$\alpha_t(i) = \pi_i b_i(O_1), 1 \leqslant i \leqslant N \tag{3.16}$$

2. Induction

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i).a_{ij}\right].b_j(O_{t+1}),\ 1 \leqslant t \leqslant T-1,\ 1 \leqslant j \leqslant N \quad (3.17)$$

3. Termination,

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i). \quad (3.18)$$

Figure 3.7 depicts the induction step. During each step associated state transition probability is multiplied with the probability of each previous partial observation sequence. Then the products are summed up and multiplied by the symbol probability of the current observation symbol. Since the process of choosing the next step only depends on the current state and not any predecessor steps, all the previous partial observation probabilities can be merged at each step. Finally, termination sums up the results of the last step.

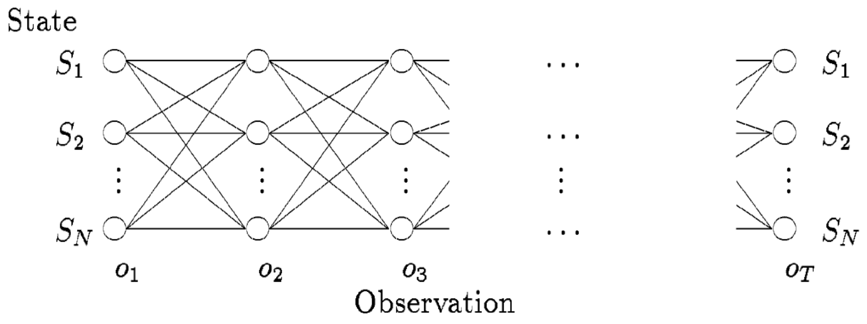


*Figure 3.7: Illustration of forward-backward procedure, (original figure in [42])*

Similarly, with the model being in state $S_i$ at time $t$, the probability of a partial observation sequence $O_{t+1}O_{t+2}...O_T$ from $t+1$ to $T$ can be defined with a backward variable $\beta_t(i)$ as follows

$$\beta_t(i) = P(O_{t+1}O_{t+2}...O_T|q_t = S_i, \lambda) \quad (3.19)$$

The problem can be solved by the backward variable, similar to the forward variable:

1. Initialization,

$$\beta_t(i) = 1, 1 \leqslant i \leqslant N \quad (3.20)$$

2. Induction

$$\beta_t(j) = \sum_{j=1}^{N} a_{ij}.b_j(O_{t+1}).\beta_{t+1}(j),$$
$$t = T-1, T-2, ..., 1, \quad 1 \leqslant i \leqslant N \tag{3.21}$$

3. Termination,

$$P(O|\lambda) = \sum_{j=1}^{N} \pi_i.b_j(O_1).\beta_1(j). \tag{3.22}$$

It is important to note that, for calculation of $P(O|\lambda)$ one of these methods will be sufficient.

**Problem 2: Choosing the a state sequence that best explains the observations**

This problem attempts to decode the hidden part of the HMM which is finding the *correct* state sequence. Considering the given observation sequence $O = O_1O_2...O_T$, and a model $\lambda$ we want to choose a corresponding state sequence $Q = q_1q_2...q_T$, which is optimal in some meaningful sense. Since, in real-world applications the absolute correct state sequence cannot be found we try to solve the problem as good as possible using *Viterbi Algorithm* [28].

The Viterbi Algorithm (VA) was introduced in 1967, by Viterbi [86] for decoding convolutional codes. It has been widely used in variety of digital estimation problems. VA considers the problem of estimating the state sequence of a discrete time finite-state Markov process and provides a recursive optimal solution [28]. For finding the best state sequence $Q$ we need to define $\delta_t(i)$ which is the highest probability along a single path up to time $t \leq T$ as follows:

$$\delta_t(i) = \underset{q_1,q_2,\cdots,q_{t-1}}{max} P[q_1q_2...q_t = i, O_1O_2...O_t|\lambda] \tag{3.23}$$

Also, we should define $\psi_t(j)$ as a variable that keeps track of the argument that is maximized. The recursive procedure can be defined as follows:

- Initialization:

$$\delta_1(i) = \pi_i.b_i(O_1) \qquad 1 \leq i \leq N \tag{3.24}$$

$$\psi_1(i) = 0 \tag{3.25}$$

- Recursion:

$$\delta_t(j) = \max_{1\leq i\leq N}[\delta_{t-1}(i).a_{ij}].b_j(O_t) \qquad 2 \leq t \leq T,\ 1 \leq j \leq N \tag{3.26}$$

$$\psi_t(j) = \underset{1\leq i\leq N}{argmax}[\delta_{t-1}(i).a_{ij}] \qquad 2 \leq t \leq T,\ 1 \leq j \leq N \tag{3.27}$$

- Termination:

$$P^* = \max_{1\leq i\leq N}[\delta_T(i)] \tag{3.28}$$

$$q^* = \underset{1\leq i\leq N}{argmax}[\delta_T(i)] \tag{3.29}$$

- State Sequence Backtracking:

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \qquad t = T-1, T-2, ..., 1 \tag{3.30}$$

As it is obvious from the introduced algorithm, VA is very similar to Forward variable calculation in the Forward-Backward procedure. The only difference is that instead of the sum of previous probabilities, the maximum probability of the previous iteration is used. $\psi_t(j)$ which is calculated using $\delta_{t-1}(i)$ is used to extract the actual state sequence.

**Problem 3: Training the HMMs to recognize the previously unknown data**

This problem corresponds to adjusting parameter $\lambda$ in order to maximize $P(O|\lambda)$. There is no optimal way of estimating the model parameters, given the training training data as finite observation sequence. Locally iterative algorithms such as Baum-Welch method [62, 87] can solve this problem. For iterative update and improvement of HMM parameters, assuming that we have the model and the observation sequence, it is possible to define the probability of being in state $S_i$ at time $t$ and being in state $S_j$ at time $t+1$ as follows:

$$\xi_t(i,j) = P(q_t = S_i, q_{t+1} = S_j|O, \lambda) \tag{3.31}$$

Figure 3.8 depicts the procedure and the sequence of operations by which $\xi_t(i,j)$ is calculated.

Using the forward and backward variables from 3.17 and 3.21, we can write $\xi_t(i,j)$ as follows:

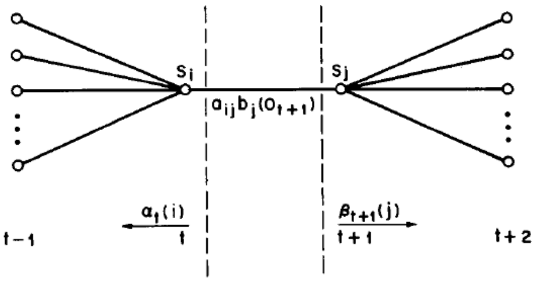


*Figure 3.8: Illustration of* $\xi_t(i,j)$, *(original figure in [62])*

$$\xi_t(i,j) = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1(j)}}{P(O|\lambda)} = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1(j)}}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1(j)}} \tag{3.32}$$

Next, it is possible to define $\gamma_t(i)$ using the calculated $\xi_t(i,j)$, as the probability of being in state $S_i$ at time $t$, given the observation sequence and the model.

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j) \tag{3.33}$$

Moreover, the expected number of transitions from state $S_i$ can be calculated by summing $\gamma_t(i)$ over time $t$. And similarly, the expected number of transitions from state $S_i$ to state $S_j$ is calculated by summing $\xi_t(i,j)$ over time $t$.

$$\sum_{t=1}^{T-1} \gamma_t(i) = \quad \text{expected number of transitions from } S_i \tag{3.34}$$

$$\sum_{t=1}^{T-1} \xi_t(i,j) = \quad \text{expected number of transitions from } S_i \text{ to } S_j \tag{3.35}$$

Now, it is possible to reestimate the parameters of HMM.

$$\bar{\pi}_i \quad \text{expected frequency in state} S_i \text{ at time } (t=1) = \gamma_1(i) \tag{3.36}$$

$$\begin{aligned}\bar{a_{ij}} &= \frac{\text{expected number of transitions from state} S_i \text{to state } S_j}{\text{expected number of transitions from state} S_i} \\ &= \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}\end{aligned} \tag{3.37}$$

$$\begin{aligned}\overline{b_{ij}} &= \frac{\text{expected number of times in state j and observing symbol } v_k}{\text{expected number of times in state j}} \\ &= \frac{\underset{s.t. O_t = v_k}{\sum_{t=1}^{T} \gamma_t(j)}}{\sum_{t=1}^{T} \gamma_t(j)}\end{aligned} \tag{3.38}$$

The reestimated model is defined as $\overline{\lambda} = (\overline{A}, \overline{B}, \overline{\pi})$ as computed using the equations above. It is proven by Baum et al. [12] that one of these two cases can happen:

- The initial model $\lambda$ defines a critical point in which $\lambda = \overline{\lambda}$ in the likelihood function.
- Since $P(O|\overline{\lambda}) \geq P(O|\lambda)$, we have created a model $\overline{\lambda}$ from which the observation sequence is more likely to have been produced.

### 3.5.4 Types of Models

Until now we have considered a special case of HMMs in which every state is reachable by every other state in a finite number of states. This is called ergodic or fully connected HMM. As it is shown in the figure 3.9(a) in a 4-state ergodic model every coefficient $a_{ij}$ is positive. Therefore, for figure 3.9(a) we have:

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$$

This model is not always the best choice. For some applications such as speech recognition in which the states proceed with time another form of model will be more efficient. Left-right or Bakis model [11] can model the signals whose property change during time. With simpler words, the cases in which the state indexes increase with time so it can be said that they proceed from *left to right.* An example of this model with four states is shown in figure 3.9(b). The transition matrix of this example can be defined as follows:

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & 0 \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{pmatrix}$$

These properties can be formulated as:

$$a_{ij} = 0, \quad j < i \tag{3.39}$$

$$a_{ij} = 0, \quad j > i + R \tag{3.40}$$

In which R is defined as the *reach* of the model that is the number of states that are reachable from state $S_i$, ( excluding $S_i$ itself). In the example of figure 3.9(b) the reach will be equal to 2.

Also, further constraints are applied on the initial properties:

$$\pi_i = \begin{cases} 1 & , i = 1 \\ 0 & , i > 1 \end{cases} \tag{3.41}$$

There are more models which are not included here, as in this thesis they are not considered.

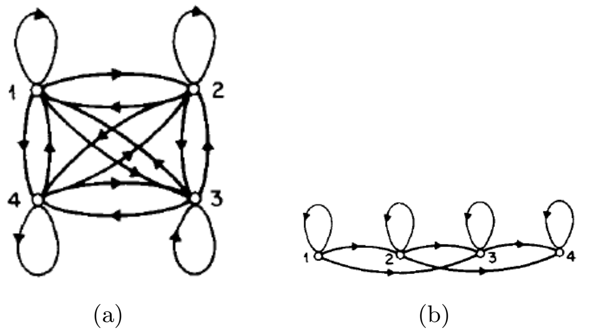


*Figure 3.9: From left to right: (a) A 4-state ergodic model, (b) A 4-state left-right model (original figure in [62]*

# Chapter 4

# Methodology

This chapter will detail the approach for designing a vision-based hand gesture recognition system. The proposed approach tries to meet the requirements of hand-gesture interfaces described by Wachs et al. [90]. In particular, price, responsiveness, user adaptability, learnability, accuracy, and low mental load are the aspects that have been considered in the design of the system. The Microsoft Kinect is used as an input device, provided one RGB camera, infrared emitter structured and a light receiver which receives light reflected from the surfaces of volume projects. As a result, the sensor returns a color image and depth data expression with a resolution of 640x480 at 30 frames per second, which is quite appropriate for developing applications in real time.

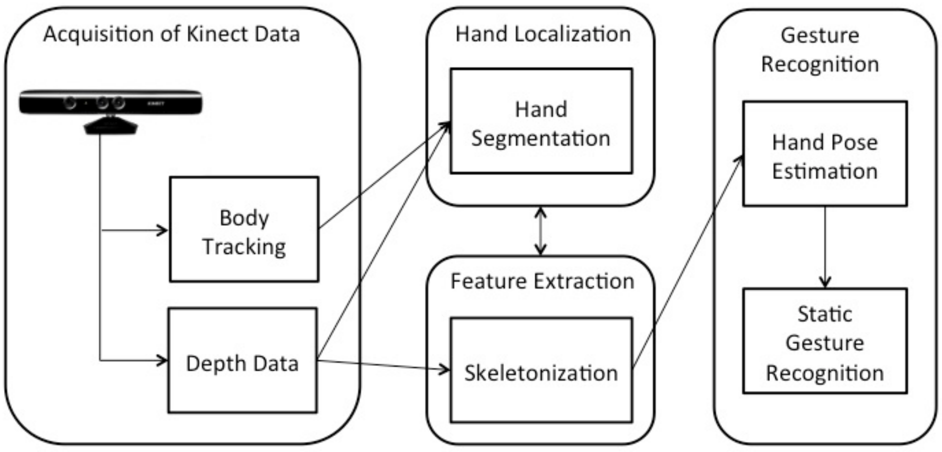


*Figure 4.1: The Gesture Recognition Scheme*

Our approach for gesture recognition is broken down into the steps depicted in Figure 4.1. Depth information are extracted from Kinect sensors to locate the hand and to recognize static hand postures. Regarding the hand tracking, the NiTE2 [4] skeletal tracking implementation has been used, which gives us the position of the wrist and the center of the palm used in hand localization, segmentation and background elimination. Later, skeletonization is done on the segmented hand to extract the required features for hand pose estimation. And eventually, a static gesture performed by the user is broken down into a sequence of hand poses that are recognized by the hand pose estimator and will be used as the input for the gesture recognition system.

In the following the methodologies and approaches for hand pose estimation and gesture recognition will be detailed. Furthermore, solutions and approaches to handle elderly limitations in using the system will be discussed.

## 4.1 Hand Localization and Segmentation

One of the essential components of hand gesture recognition is hand localization and tracking. Hand localization pertains to the positioning of the hand and to its segmentation from the background. In other words, understanding which pixels of the input image belong to the hand [78]. To understand the gesture the trajectory of the hand should be tracked. The accurate skeletal tracking in NiTE2 proposed by Shotton et al. [73], which is able to track 20 skeleton joints of the whole body in real time, in this thesis has been used for two purposes: first, to identify the center of the palm and of the wrist to position and segment the hand. Second, to track the trajectory of the (center of the) hand for gesture recognition. We use the seated mode of NiTE2 skeletal tracking which will only provide the skeleton tracking data for the upper body part, since, for our purpose the user is assumed to be commonly seated as it is shown in figure 4.2 and also for hand gesture recognition we do not need the tracking data from other body parts.

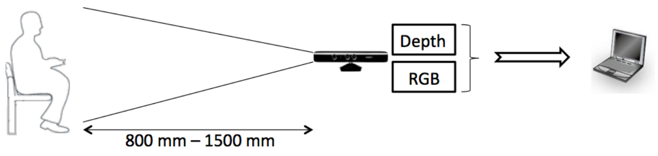


*Figure 4.2: The environmental setting*

Depth thresholding is an easy and quick way for real-time hand segmentation by assuming that hands are the closest objects to the scene. Still, this approach can be inaccurate in uncontrolled environments, which is the case in real-life practice. In the depth image (or depth map) each pixel is characterized by the distance to the depth sensor. Distinguishing palm of the hand in the image can be done in many ways. As it is common while making the gestures, hand should be on the shoulder level, slightly brought forward. Considering this factor, a natural limit can be put on human gesturing: hand should be the closest object to the sensor. In this case, the pixels of the depth map image which convey the minimum distance will be the pixels of hands. Entire palm and fingers of the hand can be selected using segmentation algorithms on the depth image [7]. In this thesis, a different method is used. Before beginning the gestures, a person makes a certain hand gesture for initialization (shaking his hand, holding hand in a certain position, etc.). In this way the hands will be recognized and using the NiTE2 tracking data the position of center of the palms will be available which is shown in Figure 4.3(c). Employing the centroid of the palm and the position of the wrist with the corresponding distance from the sensor, we are able to crop the hand region. With this approach, usage of wrist bands or dark closing to assist the hand segmentation, is not needed. Also with this approach, because of the usage of depth data instead of color images problems such as detecting other skin-colored parts such as face, and the sensitivity to the different illuminations will be solved. The segmented hand using this approach is shown in Figure 4.4.

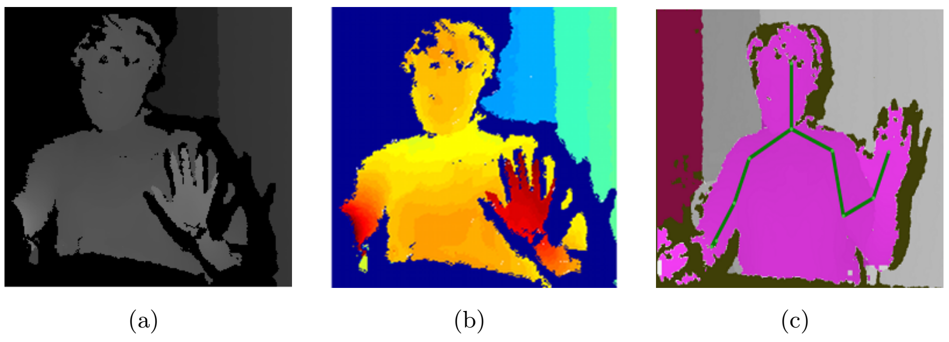


*Figure 4.3: From left to right: (a) depth data extracted from Kinect, (b) view of the depth image depicting the different levels of depth with different colors, (c) Seated-mode Skeletal Tracking Output of NiTE2*

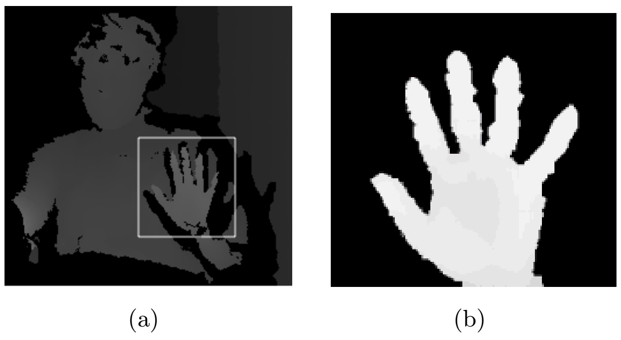


Figure 4.4: (a) Hand localization from the depth image (b) The segmented hand

## 4.2 Feature Extraction

In this section the approaches used to extract features form the segmented hands are explained. These features are then used in the hand pose estimation process.

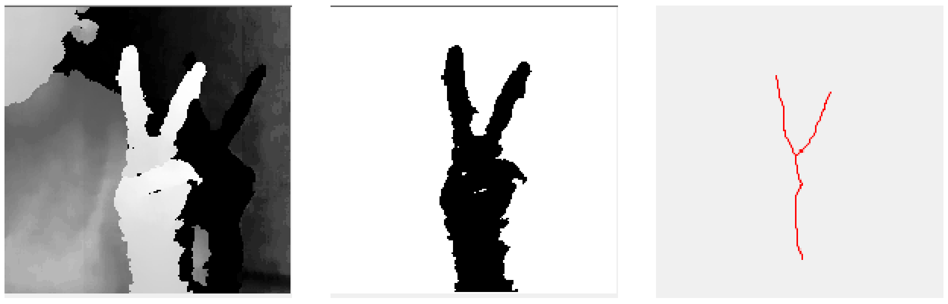
### 4.2.1 Extracting the Geometric Skeletons of the Hand

First of all we need to extract the skeleton of the hand to be able to estimate the hand poses. To obtain the geometric skeleton of the hand, the Voronoi diagram on boundary points is adopted. The geometric skeleton is one of the descriptors of the object, which is widely used in computer vision for recognition of two-dimensional and three-dimensional objects [79, 19, 84]. As mentioned in Chapter 3, to compute the Voronoi Medial Axis (VMA) we need to find the boundaries of the segmented hand first and use the sample points on the boundary to calculate the Voronoi Diagrams.

At a later stage, the extra arc of the skeleton are removed by *pruning* as explained in Chapter 3; an example of the main phases for extracting the geometric skeleton of the hand is shown in the Figure 4.5.

### 4.2.2 Extracting the Fingertips

Fingertips are important features in recognition of the hand pose. Also, knowing the position of the fingers can be adopted in applications like virtual multitouch surfaces. Using the skeleton graph as the representation of the hand pose, it is fairly straightforward to extract the number of fingertips and the position of each finger. First, some definitions are introduced. If we consider the skeleton to be a graph $G$ with $V = \{v_1, v_2, ...v_n\}$ as vertices (nodes) and $E$ the edges between two vertices (nodes) we will have:



(a) (b) (c)

*Figure 4.5: (a) Depth image of the hand, (b) Monochrome segmented hand, and (c) The geometric skeleton of the hands after pruning*

*Definition 1*: An *end node* is the node that has only one edge and is connected to only one other node.

*Definition 2*: A *joint* is a node that has more than two edges and is connection to at least three other nodes.

*Definition 3*: An *intermediate node* has exactly two edges and is connected to exactly two other nodes.

With these definitions it is possible to count the number of fingertips and also find their positions. Using simply this method will cause the position of the wrist to be also counted as the fingertips, since the graph is not considered to be directed. As for this application only the number of visible fingertips is important and the position of the fingers are ignored, the number of fingers is simply the number of end nodes reduced by 1. However, even if it is necessary to rule out the wrist from the end nodes, either it is possible to define a directed graph and consider fingertips to be the end nodes with only one incoming edge. Or it is possible to rule out the wrist by considering the fact that it is usually in a lower position than the fingertips. And therefore, it is possible to ignore the end node with the minimum $y$ in the $y-axis$.

## 4.3 Hand Pose Estimation

Hand pose estimation pertains recognition of the postures of the hand. This process uses the data provided by feature extraction approach, and the output will be the recognized hand pose for the performed posture. The method presented here takes into consideration the limitations of the elderly users.

### 4.3.1 Normalizing the Skeleton

The geometric skeleton of the hand should be invariant to translation and scaling. Invariance to rotation have not been considered since the orientation of the hand can be a feature to differentiate different hand poses. For a fair comparison of the skeletons, first they should be normalized. Normalization involves changing the scale and location of the hand skeleton. The scale is normalized based on the distance from the center point of the palm to the sensor. The normalized skeleton is considered to be at the distance of 1 meter $Dist_{norm}$ from the sensor as shown in figure 4.6. $Size_{real}$ is the real size of the user's hand pose. $Dist_{real}$ is the real distance of the center of the palm from the user. $Size_{observed}$ is the size of the hand pose that is seen by the sensor and can be different considering the distance from the sensor. And the $Size_{norm}$ is the normalized scale that we want to calculate.

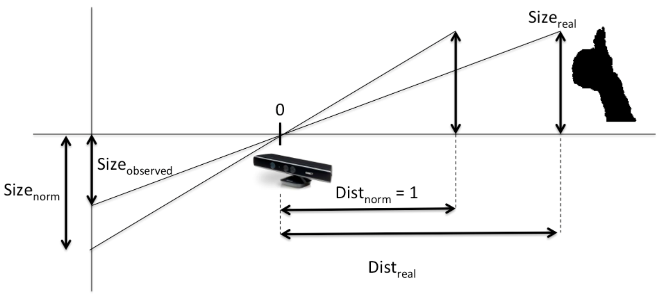


Figure 4.6: The parameters used for normalizing the scale of the skeleton

As depicted in Figure 4.6, normalizing the scale of the skeleton can be done as follows for each point:

$$Size_{norm} = Dist_{real} \times Size_{observed} \tag{4.1}$$

Later, skeleton shifting is adopted to match its geometric center to the origin of the coordinate axes as shown in figure 4.7. and the other points will be reestimated accordingly.

### 4.3.2 Comparing the Skeletons

Despite the large number of object recognition algorithms, the direct use of classical algorithms for comparing skeletons to recognize hand gestures do not give satisfactory results. This is due to the fact that with the geometric skeletons the same gestures may differ from each other for the number of

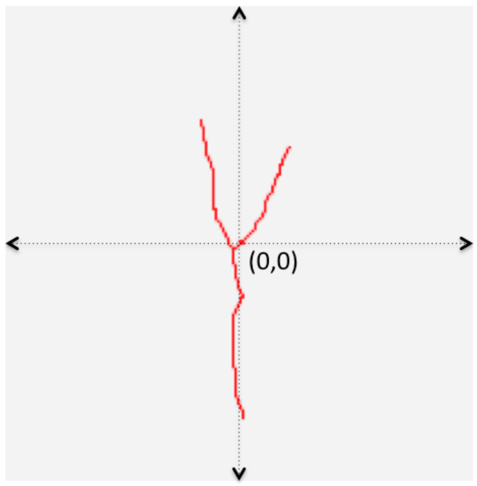


*Figure 4.7: Normalized skeleton regarding the transformation*

arcs and the lengths and angles of the individual arcs of the skeletons, Figure 4.8.

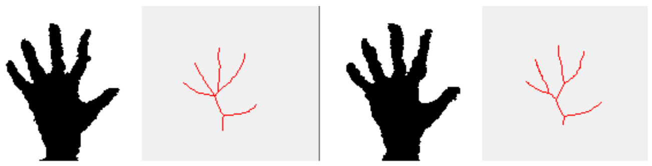


*Figure 4.8: The difference of skeletons with regard to the same hand pose*

It is possible to compare each observed skeleton with all the targeted hand postures that have been previously defined and saved in a database, and select the hand pose with the minimum distance from the observed hand pose. However, this method can be time consuming. Since, in HCI applications, the responsiveness of the system is of great importance, in this thesis, a method has been proposed to extract the spatial features of the hand pose to be further used in the gesture recognition system. A reference hand skeleton, which is the skeleton of the open hand with five visible fingers has been considered. All the observed hand poses are compared with this reference to obtain a level of similarity of each hand pose with the reference one.

Two methods have been adopted and compared for estimating the similarity between two skeletons. Dynamic Time Warping (DTW) - explained in section 3.4.1 - and Optimal Subsequence Bijection (OSB) - detailed in section 3.4.2. Both have been used successfully to compare time sequences. OSB is able to jump some of the data in the sequence by considering a penalty and to find the optimal subsequence of subject and target sequences that best match each other. On the other hand, DTW matches all the points in both sequences to the closest corresponding point. This can make DTW inefficient when there are many differences between the compared sequences. For a fair comparison the skeletons are first normalized as described before. Also, the points are sorted in both the target and the subject as both algorithms are implemented to work with time sequences and therefore are sensitive to the order in the sequences. Euclidean metrics have been adopted in both algorithms as the distance metric. The used algorithms are explained in details in Chapter 3.

## 4.4 Gestures and Hand Poses For The Use of Elderly

As mentioned in Chapter 1, research and studies in geriatrics suggest that in elderly people, both men and women, degenerative changes in musculoskeletal, vascular and nervous systems lead to hand function degradation in terms of handgrip and finger-pinch strength, maintaining pinch force and posture, and dexterity of manual movements [65]. Reduced hand function results in limitations in hand maneuvers needed for a gesture-based human computer interaction. This problem is tackled in this work, first, by characterizing the hand movement limitations, and second, by introducing an appropriate set of gestures for each category. In the following, a gesture set considering the before mentioned hand motor limitations and limit of memory capacity in elderly, have been designed. This gesture set will be used later for training the system.

### 4.4.1 Gesture Set

As mentioned in Chapter 1, the limitations for elderly hand movements can be categorized in three main groups:

- *Hand Tremor Limitations* which include the involuntary shaking of the hands.
- *Coarse Hand Movement Limitations* in which the user has problems or feels pain in moving the arms.
- *Hand Posture Limitations* that unable the user to perform complex hand postures.

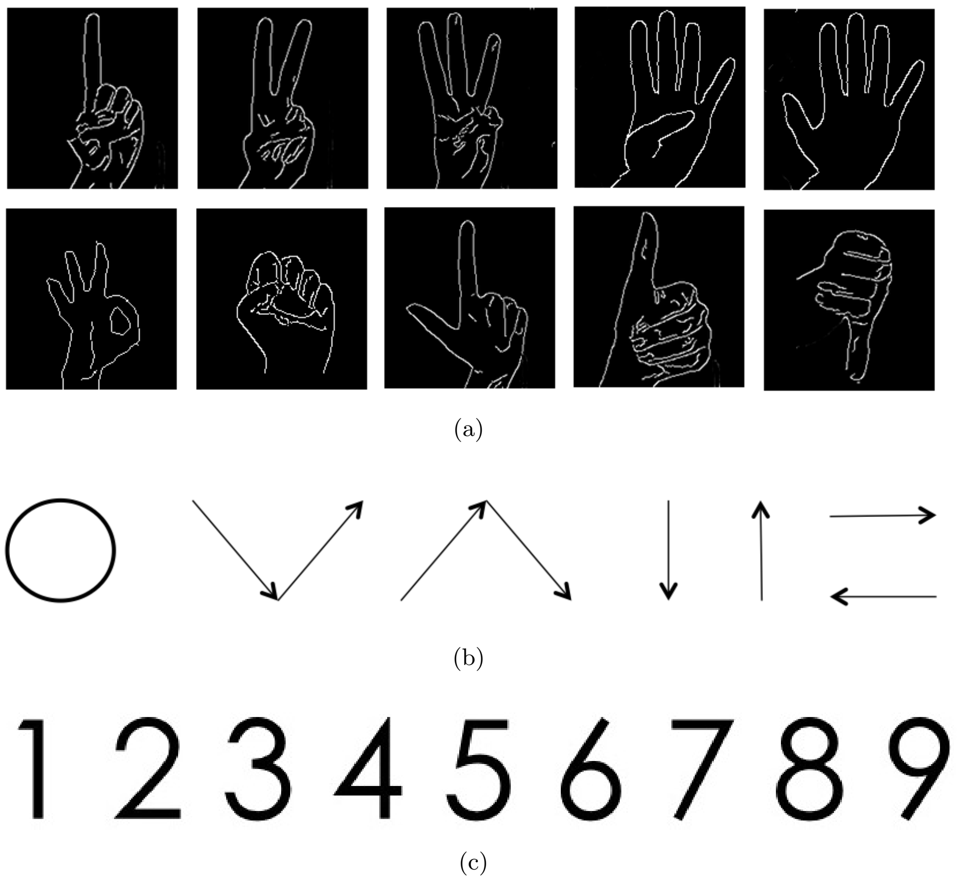


*Figure 4.9: (a) Set of hand posture samples, (b) Set of simplified dynamic gesture samples that can be used in combination with hand postures, (c) Set of dynamic number gestures*

Elderly people, mainly put in these categories on the basis of their age, can have different capability levels. An older adult might have none of the limitations mentioned above, while another could have all of them, while the limitations can be present in each individual with different levels of intensity. Considering the characteristics of the limitations and their level of intensity a mix of gestures or/and hand poses can be used. A person with hand posture limitation could opt to use mostly dynamic gestures with hand movement whereas another person with painful arm movements could choose to use mainly the static gestures forming different hand postures.

To adapt the system to the specific user needs, the system is able to recognize wide variety of hand poses and gestures that can be chosen and defined by the user or the specialist or physician to guarantee the ease of use and the performance and usability of the system. Also, when the system is configurable upon the needs and abilities of the user, the user, especially the elderly, is more encouraged to employ the system. Figure 4.9 depicts some of the possible poses and gestures that can be defined for elderly users. If the user suffers from a single kind of hand movement limitation, a specified gesture set could be used, or the system should be flexible enough to allow the user to select in each set a combination of static hand poses and dynamic gestures to define a personalized gesture set.

Dynamic gestures pertain to the gestures that consider the trajectory of the hand. Figure 4.9(b) and figure 4.9(c) are examples of these group of gestures. Dynamic gestures can be used for the group of elderly with hand posture limitation with the condition that they do not suffer from course hand movements. In any case, a physician can recommend a set of gestures that will be appropriate for the use of this group of elderly. There have been intensive research in dynamic gesture recognition and considerable approaches have been proposed in this area [21, 27, 30, 34, 36, 40, 43, 45, 46, 47, 48, 49, 51, 64, 66, 72, 80, 82, 85, 90, 92, 91, 93, 98]. Also, using Microsoft Kinect with the Kinect SDK or Nite2 tracking, it is possible to easily obtain the trajectory of the hand which can be used in dynamic gesture recognition. Therefore, these group of gestures have not been considered in this work.

On the other hand, vision based approaches for the gestures including the hand postures have not been very evolved, and there have been very limited research in this area [21, 33] that only consider a limited number of postures combined with the trajectory of the hand. Even in the case of sign language recognition the majority of approaches ignore the posture of the hand and consider only the trajectory, velocity, orientation and etc. as the parameters for recognition [89, 35, 88, 75]. Therefore, in this thesis the focus is on the *static* gestures in which the user changes his/her hand pose without considering the trajectory of the hands.

Figure 4.9(a) shows examples of static poses that can be used for users with coarse hand movement limitation. The set of hand poses are finger counts and some simple postures like thumb up and thumb down, which can be easily performed and can be associated with a set of desired actions. For elderly with hand tremor problem, the hand pose will be sampled several times while the user holds the hand pose for some seconds and the most probable posture will be recognized as the estimated hand pose. The time to hold the posture by the user can be defined experimentally and according to the heaviness of hand shake. Usually 1 to 2 seconds will be enough for an accurate pose estimation.

In this work, the static hand poses, a sequence of hand poses, and gestures that move between poses can be defined upon the needs of a specific user. The system will be defined to be used by a single user and to meet his/her specific needs. Hand poses and sequence of hand poses can be estimated by the approaches introduced in the previous Chapter. In this Chapter, however, an approach is introduced for gesture recognition considering the limitations of elderly users. The gestures are not considering the trajectory of the hand as mentioned before and only include the movement between different postures.

## 4.5 Static Gesture Recognition

Due to the stochastic nature of human including immeasurable and hidden mental states along with the measurable and observable human actions, HMMs can be used to model these processes. HMMs have been successfully used in many applications and studies for speech recognition [62] and hand writing recognition [37]. Also, it have been proven effective in sign language recognition and other complex hand gesture recognition processes [95, 69]. In this thesis HMMs are used for gesture recognition because of their simplicity and reliability.

The hand pose estimator based on DTW is able to recognize the saved postures with a reliable accuracy, however, if we consider the case in which the hand pose does not exist in the bank, or when the hand is moving between the poses (e.g., during a gesture) the ability of the pose estimator will degrade and will not be able to recognize the correct pose. Also, there are variations in features even when the same gestures are being performed by the same user. In these cases HMMs are useful to handle the variations and also to recognize the gestures that the pose estimator cannot understand.

Our approach for gesture recognition involves three major processes, namely, the feature extraction, which is done as skeletonization using Voronoi diagrams on boundaries. The next step will be hand pose estimation, in which we use DTW to compare the performed hand pose with a bank of saved poses and select the most similar as the performed pose. The last step is the recognition of the gesture which is constructed from a mix of hand poses. The process of recognition relies on HMMs. Figure 4.10 shows our approach of gesture recognition including the hand pose estimation, HMM training, and their use for recognizing the performed gestures.

### 4.5.1 Assumptions for Input Data Acquisition

We make some assumptions to acquire the input data (i.e., gestures) for gesture recognition. The user who performs the gestures is seated in front

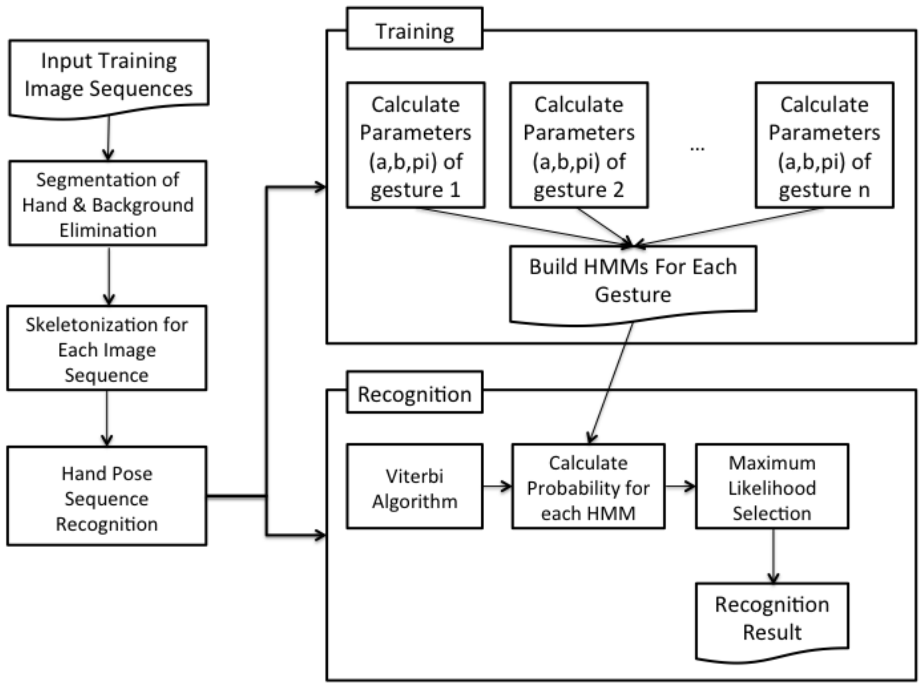

Input Training Image Sequences
Segmentation of Hand & Background Elimination
Skeletonization for Each Image Sequence
Hand Pose Sequence Recognition
Training
Calculate Parameters (a,b,pi) of gesture 1
Calculate Parameters (a,b,pi) of gesture 2
...
Calculate Parameters (a,b,pi) of gesture n
Build HMMs For Each Gesture
Recognition
Viterbi Algorithm
Calculate Probability for each HMM
Maximum Likelihood Selection
Recognition Result

*Figure 4.10: Gesture Recognition Using HMMs*

of the Kinect at a distance between 800mm to 1500mm. We assume that the user needs to wave, for the system to start looking for meaningful gestures. While user's hands are in the rest mode (e.g. hands are fixed on the arms of the armchair) the gesture recognizer also goes into the rest mode. The gesture duration is considered to be fixed and we sample the data every 200 milliseconds up to 2 seconds and the hand pose for each acquired frame will be estimated. In this way we will have 10 sequences of estimated hand poses which will be used as the input for training the HMMs and for gesture recognition.

### 4.5.2 Training the HMMs

Machine learning methods such as HMMs which treat the gesture as an output of a stochastic process have garnered significant importance for dynamic gesture recognition [51]. We also employ HMM to recognize gestures based on their pose and temporal information because of its reliability, simplicity and effectiveness. The HMM is a collection of states connected by transitions, and it can be used to represent the statistical behavior of observable symbol sequences in terms of network states. Therefore, each observation symbol is one of the states in the HMM and the HMM can stay in one state or move to another state due to probability of transition associated with each state [21].

Three parameters are considered for HMMs: initial state probability vector, state-transition probability matrix, and the observable symbol probability matrix. And the left-to-right HMM model is considered as it best fits the time-line of the gesture. For training the HMMs, the maximum number of states, number of symbols and the gesture length should be defined. By experimenting it is possible to define these as the ones who result in a better recognition. An HMM will be trained for each gesture.

As it was detailed in Chapter 3, to train the HMMs the *Baum-Welch* method [62, 87] is adopted. Using this method and having the observation sequence $O = \{O_1, O_2, ..., O_n\}$ it is possible to re-estimate HMMs parameters $(a, b, \pi)$. The training procedure can be continued until the desired accuracy in recognition of the gestures is acquired.

As this system is going to be adopted for an elderly user with their special needs, the training of the system should not use many input data, since recording these data by the elderly can be tiring and discouraging. On the other hand the gesture recognition system should be accurate and responsive even from the first usage to encourage the user to continue using the system. Therefore, a trade-off between the number of data used for training and the accuracy of the system should be adopted. Also, since we want to make the system adapted to a specific person's movements and designed for the use of that person only, the input data for training the gestures will be acquired only from the single target user.

### 4.5.3 Recognizing the Gestures

After training several HMMs, each corresponding to a specific gesture, it is possible to recognize the gestures that are not used as training data. As input, a sequence of hand poses that make the gesture will be provided. This input can be acquired as the user performs a gesture, during which some frames will be sampled, and by applying the pose estimation for each sample the recognized hand pose will be estimated.

By acquiring the hand pose sequences of a gesture observation sequence $O = \{O_1, O_2, ..., O_n\}$ will be made in which $O_i$, $i \in [1..n]$ represents a hand pose. Employing the *Viterbi Algorithm* [62] it is possible to determine the state sequence $Q = q_1 q_2 ... q_T$ given the model and the observation sequence. Later, the probability of the observation to belong to each model will be calculated using forward or backward procedures [62] and the model with maximum likelihood will be selected as the recognized gesture. If the maximum likelihood is smaller than a threshold the gesture will not be recognized.

Chapter 5

# Implementation

In this chapter the implementation of the introduced approaches in chapter 4 will be discussed. Implementation and evaluation of the approaches introduced in this thesis have been done in two levels. First, an application prototype written in C# in .NET framework, using OpenNI [5] and Nite2 [4] to acquire the data from Kinect. This prototype is used for the online testing and recording the data for offline evaluations. The data saved in this application are used in MATLAB environment for test and evaluation of the proposed approaches. Hand poses and gestures are considered to be isolated for evaluation and testing. In the following the application developed for testing the approaches will be briefed.

## 5.1 The Main Functionalities of the Application Prototype

The application prototype is the mean to extract the required data for test and analysis from the Kinect depth sensor. Also the algorithms and the approaches explained throughout this thesis are implemented as well to be able to evaluate the functionalities and performance in real-time. In this section the details about the main functionalities of the application prototype will be explained. The application main functionalities is depicted in figure 5.1.

### 5.1.1 The User Interface

A user interface is designed in C# using OpenNI [5] and NiTE2 [4]. OpenNI is an open-source SDK designed for 3D sensing application and middleware development. Together with drivers available for Kinect it can be used also with Kinect. NiTE on the other hand, is an open-source middleware

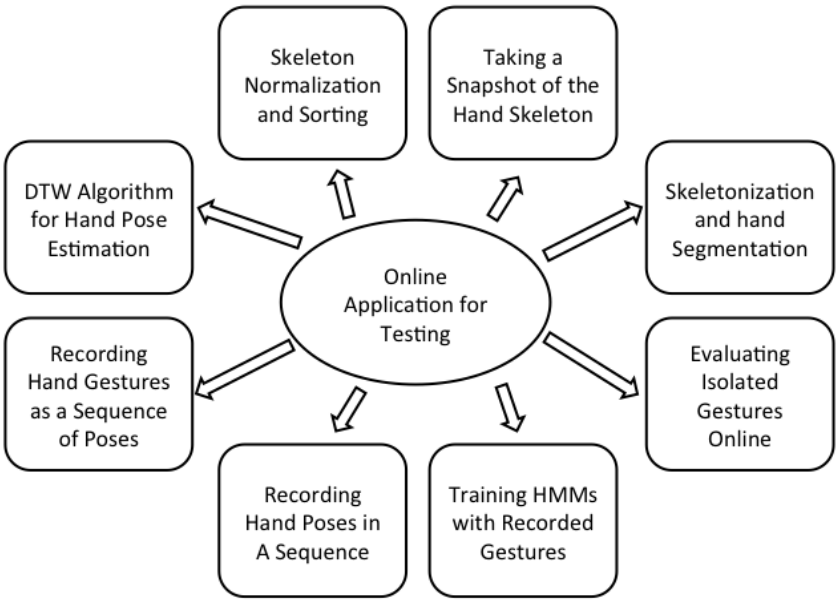

Skeleton Normalization and Sorting
Taking a Snapshot of the Hand Skeleton
DTW Algorithm for Hand Pose Estimation
Skeletonization and hand Segmentation
Online Application for Testing
Recording Hand Gestures as a Sequence of Poses
Evaluating Isolated Gestures Online
Recording Hand Poses in A Sequence
Training HMMs with Recorded Gestures

*Figure 5.1: The application prototype for testing main functionalities*

library designed by PrimeSence for *body tracking*. In this interface the user is able to see the depth images of him/herself. By waving to the Kinect the skeletonization will be started and the user is able to see the skeleton of the hand with the bounding box that shows the segmented hand area that will be calculated. From this point the user will be able to record the data needed for testing. Figure 5.2 shows a snapshot of the user interface of the application prototype for testing.

### 5.1.2 Hand Segmentation and Skeletonization

As mentioned before the application is able to segment and compute the skeleton of the hand in real-time. As explained in chapter 4 skeletonization is done using the Voronoi Diagrams on boundary points. In this application, TipTep Skeletonizer [6], a good open-source implementation of the VD on boundary points is used for generating the skeleton of the hand, which is a middleware library for OpenNI. Using this library the user is able to see the skeleton of the hand in real-time after waving to the Kinect. Waving to the Kinect is needed for the NiTE body tracking to start tracking the body parts. NiTE body tracking is used to find the center of the palm and also

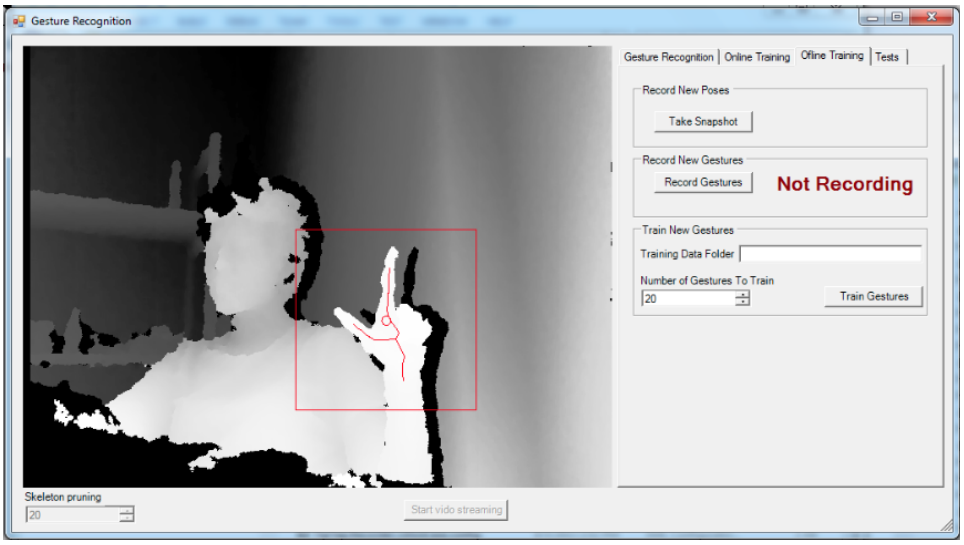


*Figure 5.2: The application prototype for testing the gesture recognition system*

to use the depth information and the position of the palm center for hand accurate hand segmentation, background elimination and normalization of the acquired skeleton.

The user is able to set the skeleton pruning level. In the experiments 20 results in an accurate skeleton. The application lets the user to take an snapshot of the skeleton information in XML format, and the depth image and the segmented hand image in bmp format. These data can be used for an offline analysis of the hand skeletons. Later in this chapter in Section 6.1 the recorded snapshots of hand skeletons will be employed to categorize errors and calculate the error rate in skeletonization and segmentation algorithm.

The procedure used in this section can be briefed in algorithm 1.

### 5.1.3 Skeleton Normalization and Sorting

The normalization of the skeletons are needed for a fair comparison. The skeletons are normalized regarding scaling and translation. The normalization due to rotation is ignored since the orientation of the hand pose is important in gesture recognition. And the slight changes in orientation would not effect the hand pose estimation in a great extent.

Furthermore, for the use of skeleton data for hand pose estimation using OSB or DTW the skeleton points should be sorted as these algorithms are actually for comparing two sequences that vary in time or frequency. And therefore, the order of the compared data can have effects in the result of the algorithms. The user is able to record the normalized and sorted

**Algorithm 1** Hand segmentation and skeletonization using the depth data and body tracking

```
Connect to the Kinect depth sensor and show the depth frames;
Wait for the user to wave to the Kinect;
if User waved to the Kinect then
    Start NiTE body tracking;
    while User hand is not in the rest mode do
        Update body tracking data;
        Use position and distance of palm center for hand segmentation;
        Eliminate the background;
        Apply Voronoi Diagram on boundaries to get the skeleton graph;
        Apply pruning using the pruning parameter provided by the user;
        if skeleton data is ready then
            Show the skeleton, palm center in real-time;
        end if
    end while
end if
```

skeleton for each hand pose in CSV format. Or to normalize and sort the previously saved skeleton data that explained in the previous section to avoid the need of recording the data twice for skeletonization analysis and for pose recognition purposes.

In figure 5.3, it is possible to view the class diagram regarding the skeletonization procedure. The class *VoronoiDiagram* is implemented to handle to normalization procedures of skeletons, palm center, and the bounding window. This class makes use of the TipTepSkeletonizer packages to get the skeleton data.

### 5.1.4 DTW Algorithm for Hand Pose Estimation

The DTW algorithm is implemented in the application prototype as detailed in algorithm 2. Figure 5.4 shows the class diagram for DTW implementation. This algorithm is used in hand pose estimation. The user is able to use this algorithm for analysis in several ways. It is possible to select a single source hand pose and compare all the targeted hand poses with it and save the data for further analysis. Furthermore, this algorithm is used in recording the gestures and hand poses. During the selected duration for recording the gesture each sample will be compared with a bank of previously saved hand poses using DTW and the most similar hand pose will be selected.

**Algorithm 2** DTW algorithm to compare two skeletons

```
Skeleton $source[0..n]$ ← Source Skeleton Data
Skeleton $target[0..n]$ ← Target Skeleton Data
$DistanceMatrix[0..n, 0..m] \leftarrow 0$
$Cost \leftarrow 0$
for $i \leftarrow 1..m$ do $DistanceMatrix[0, i] \leftarrow \infty$
end for
for $i \leftarrow 1..n$ do $DistanceMatrix[i, 0] \leftarrow \infty$
end for
$DistanceMatrix[0, 0] \leftarrow 0$
for $i \leftarrow 1..m$ do
    for $j \leftarrow 1..n$ do
        $Cost \leftarrow EuclideanDistance(source[i], target[j])$
        $DistanceMatrix[i, j] \leftarrow cost + minimum(DistanceMatrix[i-1, j],$
                                    $DistanceMatrix[i, j-1],$
                                    $DistanceMatrix[i-1, j-1])$
    end for
end for
return $DistanceMatrix[m, n]$
```

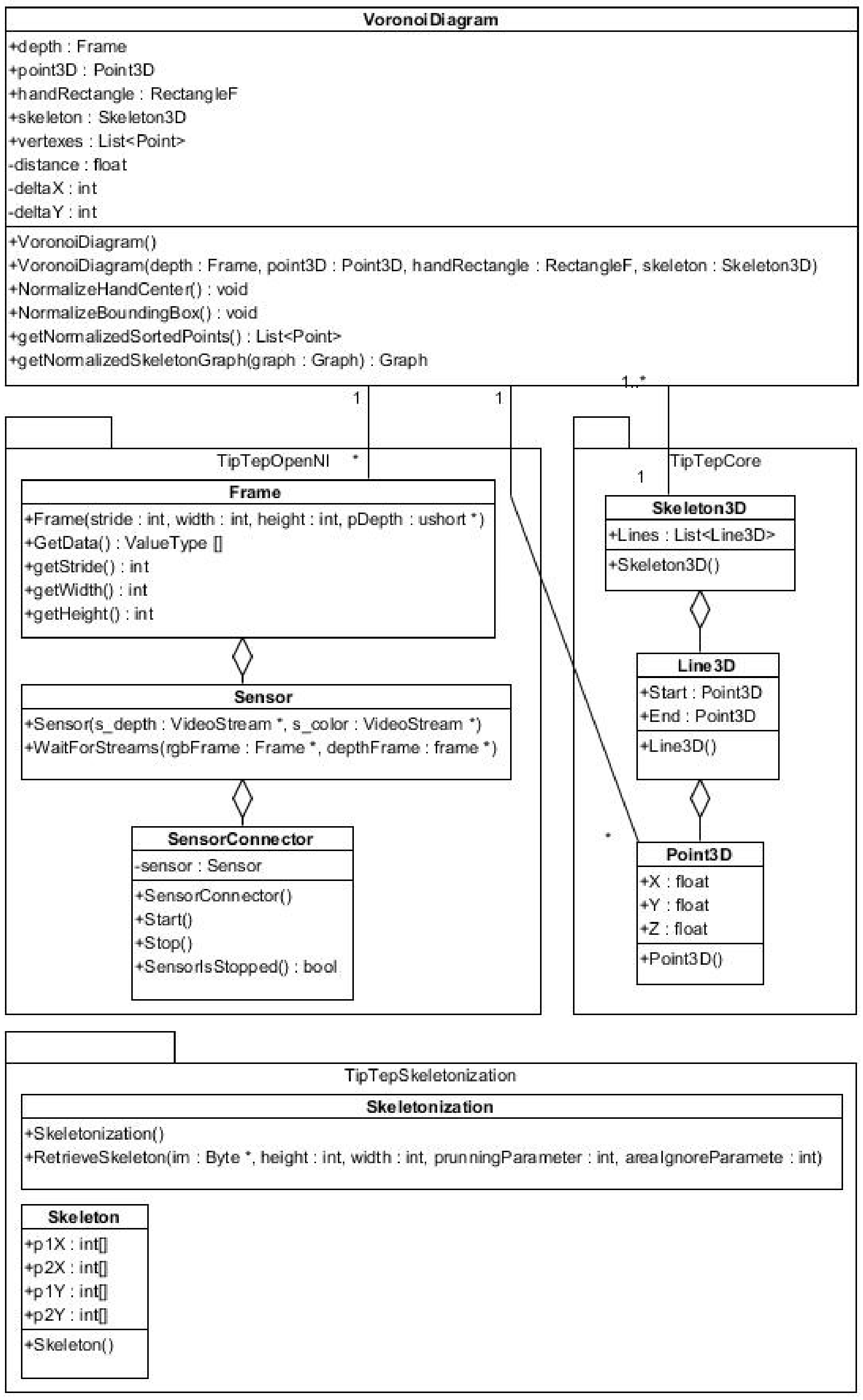

VoronoiDiagram
+depth : Frame
+point3D : Point3D
+handRectangle : RectangleF
+skeleton : Skeleton3D
+vertexes : List<Point>
-distance : float
-deltaX : int
-deltaY : int
+VoronoiDiagram()
+VoronoiDiagram(depth : Frame, point3D : Point3D, handRectangle : RectangleF, skeleton : Skeleton3D)
+NormalizeHandCenter() : void
+NormalizeBoundingBox() : void
+getNormalizedSortedPoints() : List<Point>
+getNormalizedSkeletonGraph(graph : Graph) : Graph
1
1
1..*
TipTepOpenNI
*
Frame
+Frame(stride : int, width : int, height : int, pDepth : ushort *)
+GetData() : ValueType []
+getStride() : int
+getWidth() : int
+getHeight() : int
Sensor
+Sensor(s_depth : VideoStream *, s_color : VideoStream *)
+WaitForStreams(rgbFrame : Frame *, depthFrame : frame *)
SensorConnector
-sensor : Sensor
+SensorConnector()
+Start()
+Stop()
+SensorIsStopped() : bool
TipTepCore
1
Skeleton3D
+Lines : List<Line3D>
+Skeleton3D()
Line3D
+Start : Point3D
+End : Point3D
+Line3D()
*
Point3D
+X : float
+Y : float
+Z : float
+Point3D()
TipTepSkeletonization
Skeletonization
+Skeletonization()
+RetrieveSkeleton(im : Byte *, height : int, width : int, prunningParameter : int, areaIgnoreParamete : int)
Skeleton
+p1X : int[]
+p2X : int[]
+p1Y : int[]
+p2Y : int[]
+Skeleton()

*Figure 5.3: Class diagram of the skeletonization and the packages of TipTep middleware library*

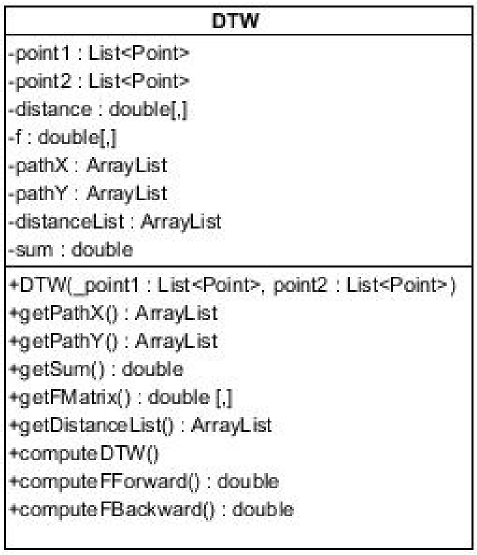


*Figure 5.4: Class diagram for Dynamic Time Warping Implementation*

### 5.1.5 Recording Hand Pose and Gestures

It is possible to record hand poses and gestures for the use in evaluation or offline training of HMMs. five cases are considered:

1. Recording a sequence of static hand poses
2. Recording a single hand pose in the presence of hand trembling
3. Saving the fingertip count and their locations
4. Saving the height and width of the hand pose
5. Recording a static gesture

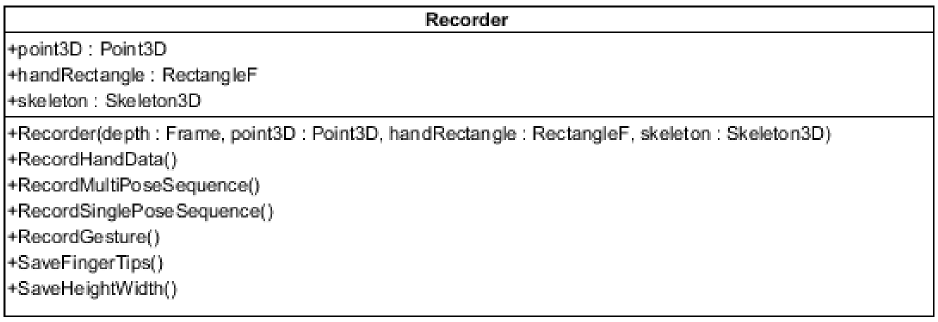

Recorder
+point3D : Point3D
+handRectangle : RectangleF
+skeleton : Skeleton3D
+Recorder(depth : Frame, point3D : Point3D, handRectangle : RectangleF, skeleton : Skeleton3D)
+RecordHandData()
+RecordMultiPoseSequence()
+RecordSinglePoseSequence()
+RecordGesture()
+SaveFingerTips()
+SaveHeightWidth()

*Figure 5.5: Class diagram of the Recorder class*

Figure 5.5 shows the class diagram of the recorder class. The first case considers number of hand poses that are estimated using DTW in a sequence. This sequence can be used together to convey an action. The second case, is used for testing the approach robustness in the presence of hand trembling or shakes which is very common in elderly user. As stated in previous chapters, to overcome the problem of hand shakes in pose estimation, the hand pose is sampled several times in a short period like in 1 to 2 seconds depending on the heaviness of the hand shakes. Moreover, it is possible to calculate the fingertip count and locations together with the width and the height of the hand pose to be used as extra features in the analysis. Last but not the least, the static gestures can be recorded for a later use in offline training of HMMs for gesture recognition.

### 5.1.6 Training HMMs Offline

In the application Accord.NET framework is used that includes the HiddenMarkovModel class. The application is able to train the HMMs using the previously recorded gestures. The number of training data and the number of states and symbols can be set by experience to get optimum results. Also, it is possible to set the parameters of an already trained HMM model to avoid training again. In figure 5.6 it is possible to view the class diagram of this library.

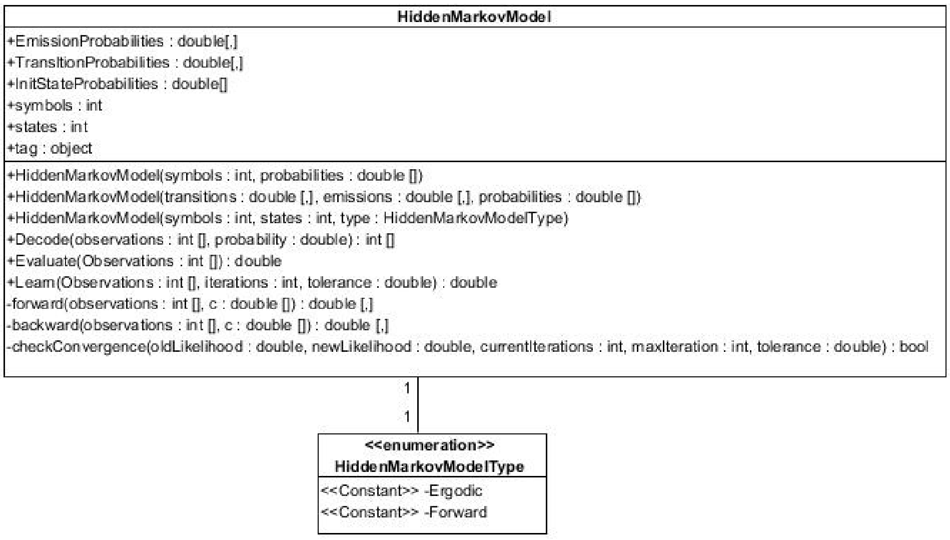

HiddenMarkovModel
+EmissionProbabilities : double[,]
+TransitionProbabilities : double[,]
+InitStateProbabilities : double[]
+symbols : int
+states : int
+tag : object
+HiddenMarkovModel(symbols : int, probabilities : double [])
+HiddenMarkovModel(transitions : double [,], emissions : double [,], probabilities : double [])
+HiddenMarkovModel(symbols : int, states : int, type : HiddenMarkovModelType)
+Decode(observations : int [], probability : double) : int []
+Evaluate(Observations : int []) : double
+Learn(Observations : int [], iterations : int, tolerance : double) : double
-forward(observations : int [], c : double []) : double [,]
-backward(observations : int [], c : double []) : double [,]
-checkConvergence(oldLikelihood : double, newLikelihood : double, currentIterations : int, maxIteration : int, tolerance : double) : bool
1
1
<<enumeration>>
HiddenMarkovModelType
<<Constant>> -Ergodic
<<Constant>> -Forward

*Figure 5.6: Class diagram of the Accord.NET Hidden Markov Model*

For training the HMMs, the procedure described bellow will be followed:

1. The recorded static gestures are divided into training and testing sets. Two-third of the data is used for training and one-third for testing or validation.
2. Initialize three HMM parameters: initial state probability vector, state-transition probability matrix, and the observable symbol probability matrix.
3. Choose the number of states, symbols and number of required HMMs (one for each gesture).
4. Re-estimate the HMM parameters for each gesture using Baum-Welch method until the convergence threshold of 0.0001 is acquired.
5. Repeat 4 for each gesture.

### 5.1.7 Evaluating Online Isolated Gestures

Similar to the recording the gestures, gestures can also be evaluated online using the trained model. In this case, the user will perform the isolated gesture in the requested time and the system will find the most probable gesture by finding the maximum likelihood of the trained HMMs. The evaluation of the gesture is done using the *Evaluate* method of *HiddenMarkovModel* class by providing the observation list which is the sequence of recognized symbols. This procedure is explained in algorithm 3

**Algorithm 3** Evaluating the hand gestures

```
Observation ← list of symbols;
HMMs ← list of trained Hidden Markov Models;
evalValue ← array of 0 with the size of HMMs list;
MaxLikelihood ← $\infty$;
Gesture ← null;
$i \leftarrow 0$;
for each hmm in HMMs do
    $evalValue[i] \leftarrow hmm.Evaluate(Observations)$;
    $i \leftarrow i + 1$;
end for
$MaxLikelihood \leftarrow evalValue.FindMaximum()$;
$Gesture \leftarrow hmm[evalValue.FindMaximumIndex()]$;
```

# Chapter 6

# Evaluation and Testing

MATLAB have been chosen for evaluation and testing because of the simplicity and efficiency and the presence of many useful libraries that were needed for this thesis. Also the visualization of the data and the results can be easily done in this environment.

The OSB code in MATLAB and C++ was provided by the writers of [38]. DTW and OSB algorithms are tested in MATLAB for hand pose estimation. After the evaluations DTW proved to be more efficient than OSB in most cases and therefore, it was used for hand pose and gesture recognition procedures. For this reason, OSB have not been implemented in the *application prototype* that was introduced in chapter 5. For HMMs the Hidden Markov Model Toolbox for MATLAB [3] is used which is written by Kevin Murthy, 1998 and later updated in 2005. The complete MATLAB codes are provided in the Appendix A.

The procedures of Skeletonization, pose estimation and gesture recognition are tested using the application prototype that was introduced in chapter 5 and experiments showed that all of them respond in real-time.

## 6.1 Extracted Hand Skeletons

Figure 6.1 shows the results of hand skeleton recognition on 21 different hand poses. The recognition works in real time and it is stable to hand trembling, different lightings, uncontrolled environments and changing and dynamic backgrounds. However, there are some problems for computing the correct skeleton of the hand, which can be categorized as follows:

- *Problem 1 - Self Occlusion*: This problem can be caused by hand self occlusions in which some parts of the hands would not be visible regarding the camera view angle. Figure 6.2 depicts an example of self

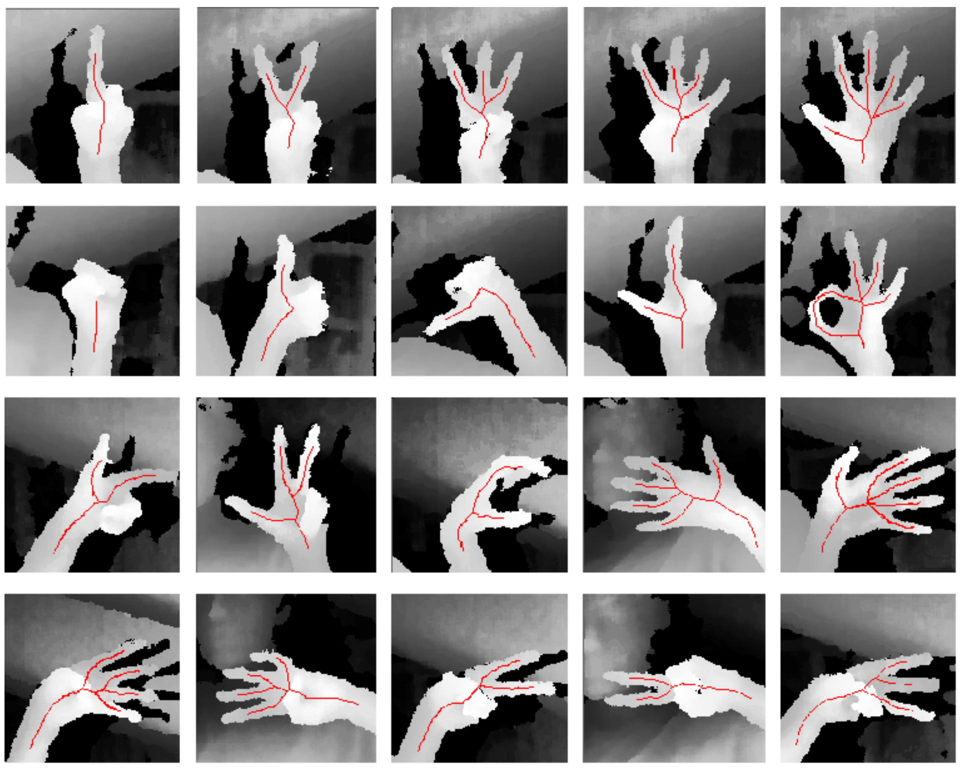
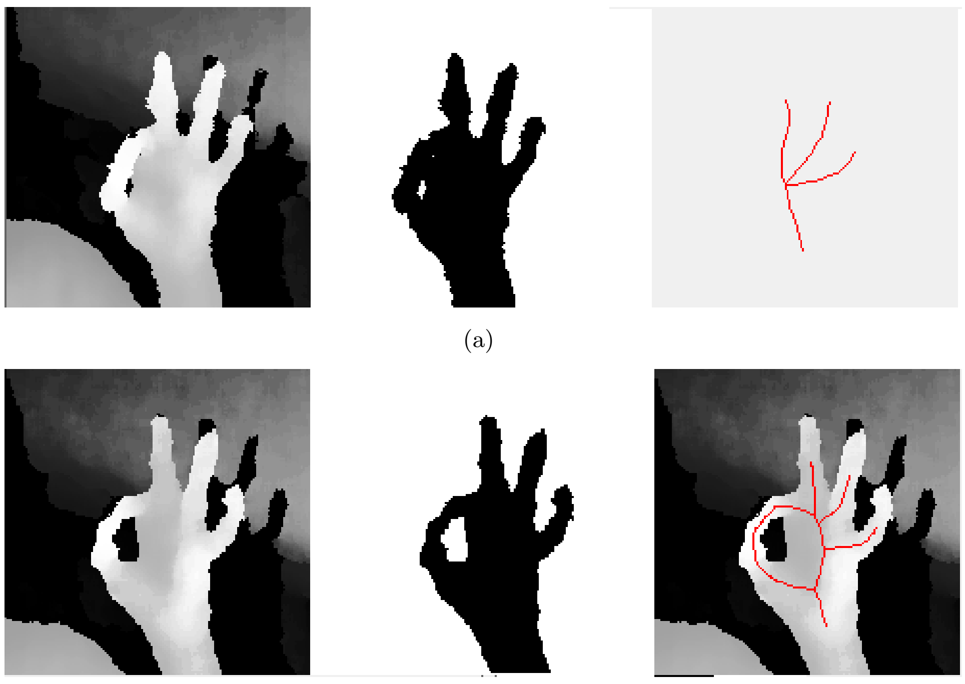


Figure 6.1: Extracted Hand Skeletons Using VD on Boundaries

occlusion and how it can affect the skeleton computation. For avoiding self-occlusion problems the poses should be performed in front of the camera parallel to the viewing angle.

- *Problem 2 - Missing parts due to errors in background thresholding*: As explained before the position of the palm's center is considered to be used for hand segmentation and excluding the background. A threshold is considered around the distance of the palm's center to recognize the hands pixels. This threshold must be chosen carefully for considering all hand pixels in the range and in the same time to exclude all other objects around and also the background. In some cases if the hand is posed in a way that some parts (usually the fingers) are bent backwards, or in other words, the closest and the furthest points of the hand to the camera have a distant more than the considered threshold, some parts of the hand might be considered as the background and therefore be canceled out by the algorithm. Figure 6.3 shows two examples of how some parts of the hand can get excluded by the hand segmentation algorithm. For avoiding these problems hands should



(a)

(b)

*Figure 6.2: Example of effects of self occlusion in skeleton computation (a) shows the skeleton of a hand pose when self occlusion occurs while (b) shows the correct skeleton in the absence of self occlusion*

stand parallel to the camera.

- *Problem 3 - Addition of some parts due to errors in background thresholding*: In the contrast with the previous case, sometimes the hand can be very close to an object in the same depth or in the background but very close to the hand. In these cases the segmentation algorithm might mistake the close object as the part of the hand as they are in the same depth range of the hand and also inside the segmentation window that is considered around the hand. This can greatly affect the skeleton computation. Figure 6.4 shows examples that some parts of the background are considered to be parts of the hand by the segmentation algorithm. As it is also visible in the images these errors occur in the poses that are performed very close to the body. Therefore, they can be simply avoided by keeping the hands apart from the body or other objects that can be around.

- *Problem 4 - Bad resolution*: As it is stated before the depth sensor of the Kinect has a very low resolution that is some cases might cause

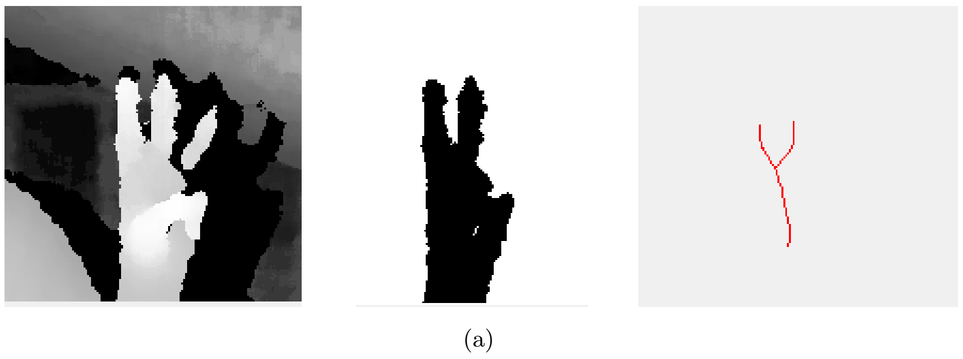
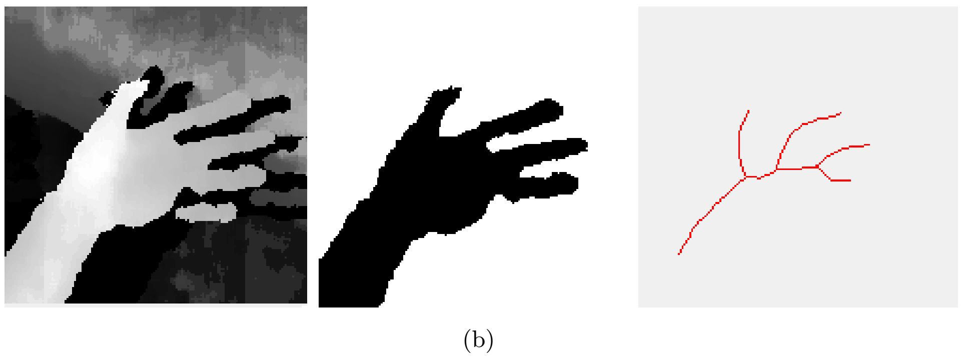
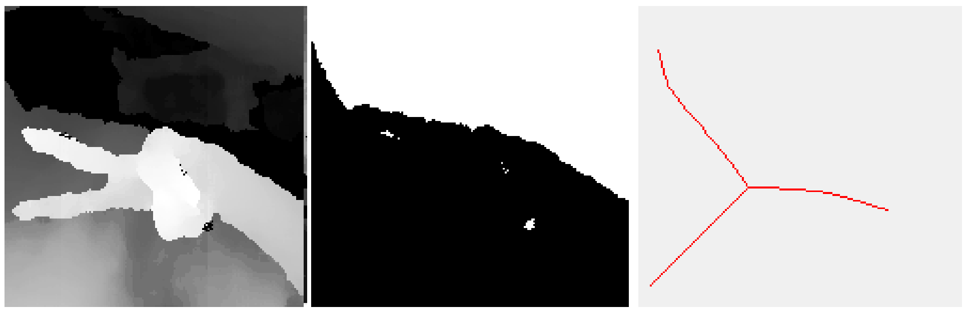



*Figure 6.3: Example of effects of hand segmentation in excluding some parts of the hand (a) a gesture with three fingers that one finger is excluded by segmentation algorithm (b) a gesture with five fingers that one finger is excluded by segmentation algorithm*

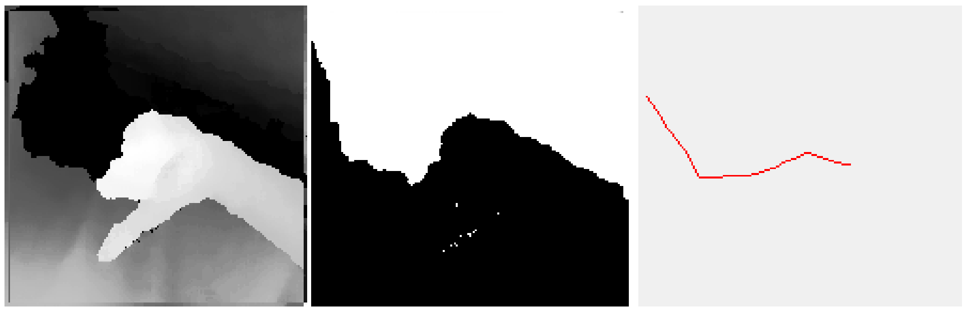
errors in the extracted skeleton. Most common problems in this category the linkage between two fingers that can fool the algorithm to consider them as one as it is shown in figure 6.5(a) or add an extra branch to represent the linkage as it is visible in figure 6.5(b).

- *Problem 5 - White image*: Another problem that can occur is that in the time of sampling the image the body tracking of NiTE2 might not have been ready and the information of the position of the hand remains unknown. This will cause segmentation output to be an empty rectangle and will result in having no skeleton information on the hand. These data will be simple excluded as errors and the data will be sampled again.

- *Problem 6 - Shadows*: In the depth information the shadows are considered to be black (as depth images are color coded with white to be the closest pixels, black the furthest pixels, and different levels of gray for the pixels in between). This may cause some irregularities in the produced skeleton since the shadow part will be eliminated by the seg-



(a)

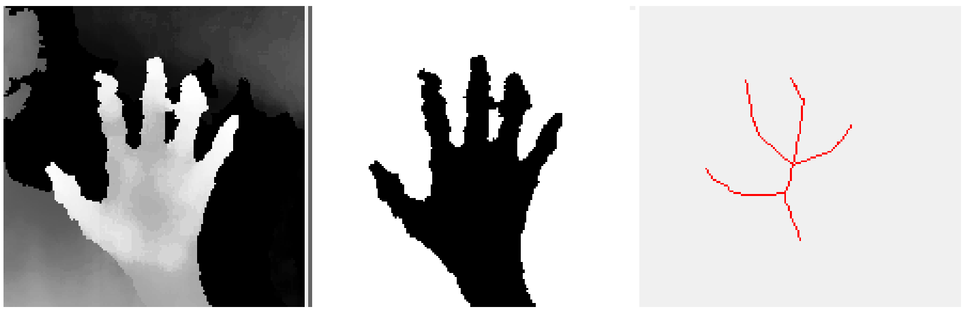


(b)

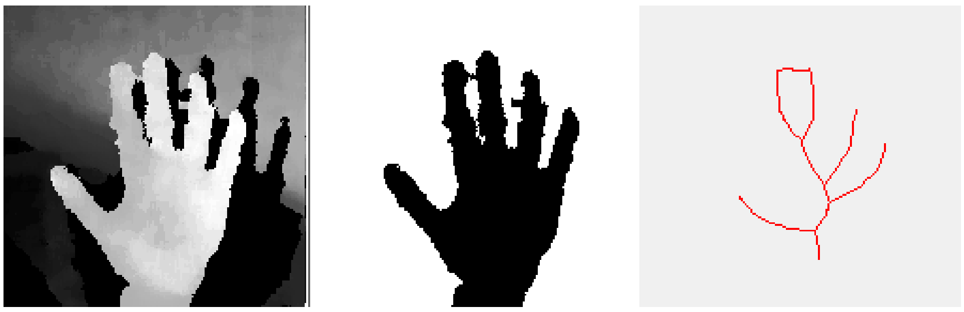
*Figure 6.4: Example of effects of hand segmentation in including some external parts in the segmented hand. Both (a) and (b) poses have been performed very close to the body*

mentation algorithm as it is considered to be part of the background. If the shadow is in between the part that is being processed the algorithm might consider the part as two separate parts and therefore, produce 2 branches to represent each. Figure 6.6 shows some examples of this problem.

Table 6.1 shows the error percentage overall and for each before mentioned problems per hand pose. The error rate for each before mentioned problem groups can be shown as follows in table 6.2. These data are concluded from 21 different hand poses and 100 samples per each, recorded by a single user. As it is possible to view in the table 6.2 the overall error rate for the introduced problems is very low. Also, for compensating for the errors usually more samples are used for pose estimation instead of one.



(a)



(b)

*Figure 6.5: Example of effects of bad resolution in skeleton computation. (a) shows how two fingers might be considered as one, (b) shows the linkage between two fingers*

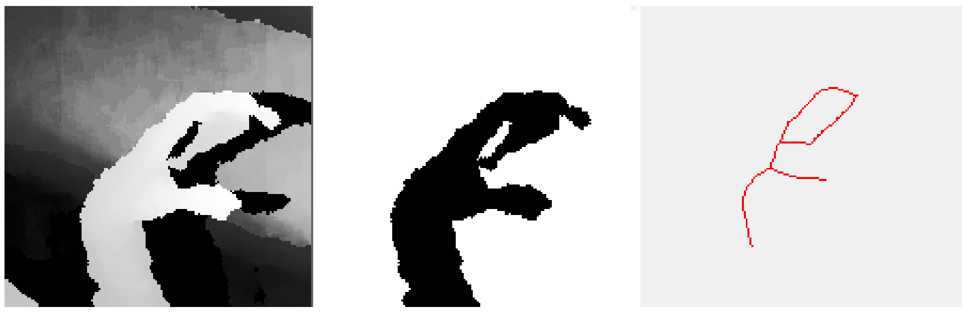
## 6.2 Skeleton Similarity

For the purpose of hand pose estimation we should somehow compare the extracted features of the hand postures. DTW and OSB algorithms can be used for comparing the skeletons of different hand postures. Also, other features such as number of fingertips, the width and the height of the posture can be helpful for discriminating the hand postures from each other. Several approaches have been tested for comparing the skeletons and hand pose recognition:

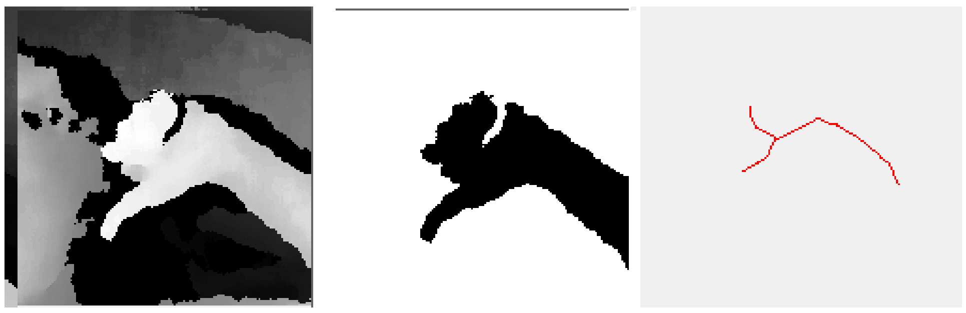
1. Comparing the specified pose skeletons with a single source (e.g. the skeleton of an open hand) and use the results of DTW or OSB algorithms together with other extracted features such as number of counted finger tips, height and width of the posture.

2. Comparing the specified pose skeletons with a bank of previously saved skeletons for desired hand poses, using DTW or OSB algorithms and select the most similar hand pose as the estimated posture.

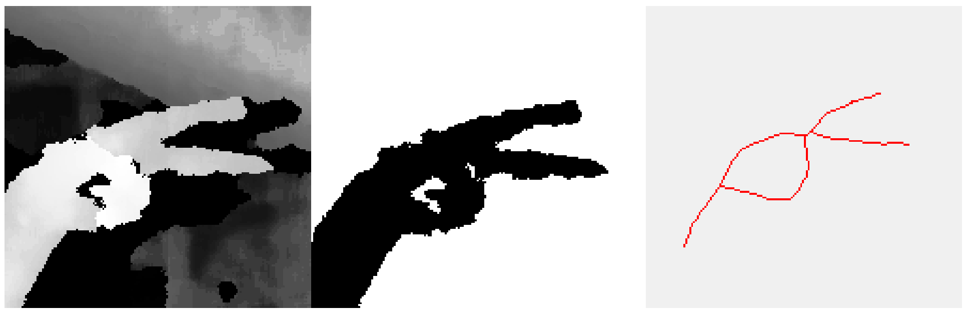
These approaches will be discussed in more details in the following.



(a)



(b)



(c)

*Figure 6.6: Example of effects of shadow in skeleton computation. (a), (b) and (c) show the extra branches that are added because of the presence of shadows*

### 6.2.1 Skeleton Comparison Results

The difference between the performances of DTW and OSB is visible when we compare different hand poses. For better understanding of these methods some visualization of the data is done. First two similar hand gestures are compared. As the results are shown in figure 6.7, OSB have skipped many nodes in the skeleton while DTW corresponds all the points together.

Now, two different gestures are compared and the results are visible in figure 6.8. Now it can be seen that OSB performs better when two very

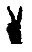
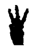
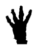
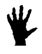
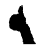
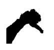
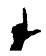
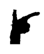
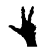
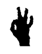
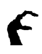
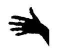
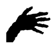
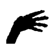
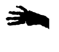
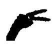
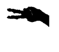
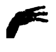
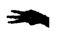
| Pose | Prob. 1 | Prob. 2 | Prob. 3 | Prob. 4 | Prob. 5 | Prob. 6 | Overall |
|---|---|---|---|---|---|---|---|
| | 0% | 0% | 0% | 0% | 3% | 0% | 3% |
| | 0% | 2% | 0% | 0% | 0% | 0% | 2% |
|  | 0% | 0% | 0% | 0% | 3% | 0% | 3% |
|  | 0% | 3% | 0% | 0% | 4% | 0% | 7% |
|  | 0% | 3% | 0% | 0% | 0% | 0% | 3% |
|  | 0% | 1% | 0% | 3% | 1% | 0% | 5% |
|  | 0% | 0% | 0% | 0% | 1% | 0% | 1% |
|  | 0% | 1% | 2% | 0% | 2% | 5% | 10% |
|  | 0% | 0% | 0% | 0% | 0% | 0% | 0% |
|  | 0% | 0% | 0% | 0% | 0% | 0% | 0% |
|  | 0% | 0% | 0% | 0% | 0% | 0% | 0% |
|  | 3% | 0% | 0% | 4% | 0% | 0% | 7% |
|  | 1% | 0% | 0% | 2% | 0% | 1% | 4% |
|  | 0% | 0% | 0% | 0% | 0% | 1% | 1% |
|  | 0% | 2% | 0% | 0% | 0% | 0% | 2% |
|  | 0% | 0% | 0% | 1% | 4% | 0% | 5% |
|  | 0% | 2% | 0% | 2% | 0% | 0% | 4% |
|  | 0% | 1% | 0% | 1% | 1% | 5% | 8% |
|  | 0% | 0% | 2% | 1% | 2% | 0% | 5% |
|  | 0% | 2% | 0% | 2% | 0% | 0% | 4% |
|  | 0% | 4% | 7% | 2% | 0% | 1% | 14% |

*Table 6.1: Error rate of skeleton calculation for each problem category and in overall per hand pose*

different sequences are compared, and it will result a better correspondence between the nodes in the subject and the targeted skeletons. While DTW, tried to find a correspondence between every 2 nodes.

| Problem 1 | Problem 2 | Problem 3 | Problem 4 | Problem 5 | Problem 6 |
|---|---|---|---|---|---|
| 0.19% | 1% | 0.52% | 0.86% | 1% | 0.62% |

*Table 6.2: Error rate of skeleton calculation for each problem category*

The results of both algorithms are shown in table 6.3 and table 6.4, for a fair comparison the results are normalized to be between 0 and 1. In table 6.3 path cost is the value of $B$ which is the coefficient of similarity between two skeletons computed using DTW algorithm. In table 6.4, path cost indicates the cost of the optimal path found using OSB algorithm between the two skeleton vertices. JP indicates the Jump Penalty and D is the Euclidean distance between the corresponding elements without considering the Jump Penalty.

| Data | Path Cost | No. of Points Compared |
|---|---|---|
| Figure 6.7 | 0.0127 | 122 |
| Figure 6.7 | 0.3200 | 111 |

*Table 6.3: Results of comparing two skeletons using DTW algorithm*

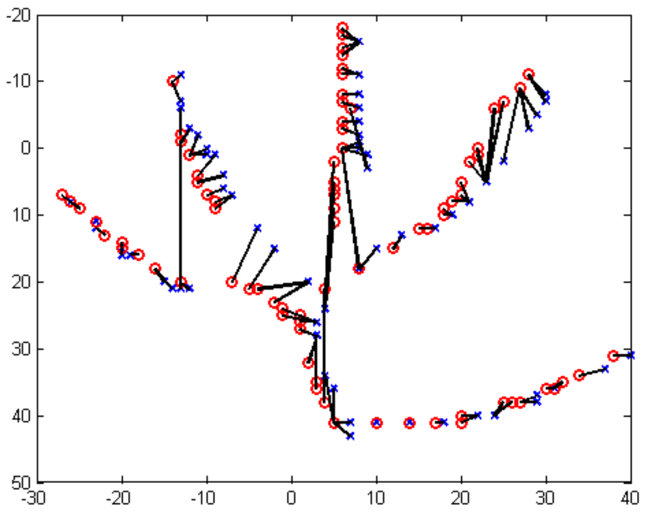
| Data | Path Cost | No. of Points Compared | JP | D |
|---|---|---|---|---|
| Figure 6.8 | 0.0043 | 37 | 0.0445 | 0 |
| Figure 6.8 | 0.0502 | 19 | 0.2264 | 0.0024 |

*Table 6.4: Results of comparing two skeletons using OSB algorithm*

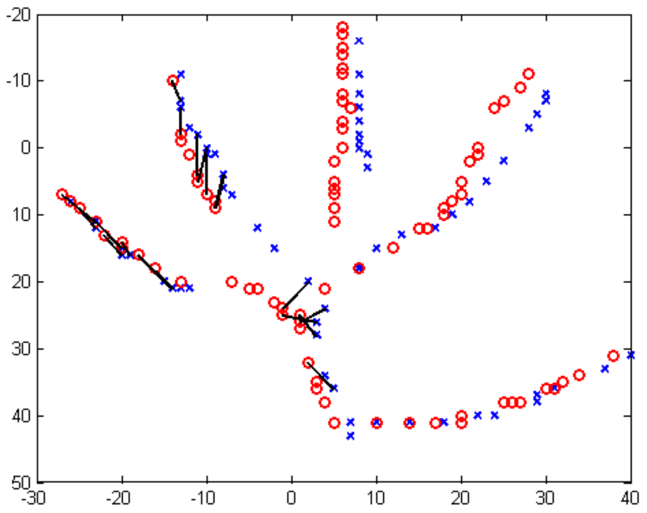
### 6.2.2 Skeleton Similarity Using a Single Source for Comparison

As stated before, one of the approaches tested to acquire classifiable data for hand pose estimation (classification), is to compare the specified hand poses with one source pose that is saved in the system. Using DTW or OSB, the level of similarity can be acquired, and this data can be used together with number of fingertips and/or width and height of the posture. To analyze the results of such approach some visualization have been done. First, a set of 10 different hand poses have been considered as shown in the guide figure 6.9.

The visualization have been done using the mixture of several extracted features. First the results of DTW and OSB algorithms are acquired for comparing the hand pose samples in the guide with the source sample which



(a)

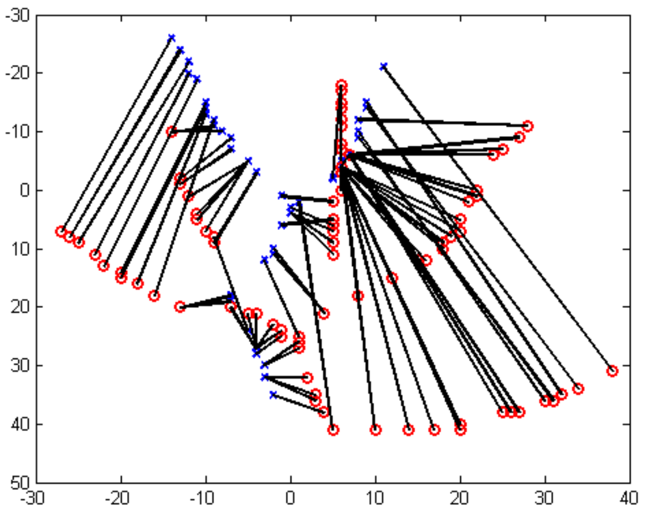


(b)

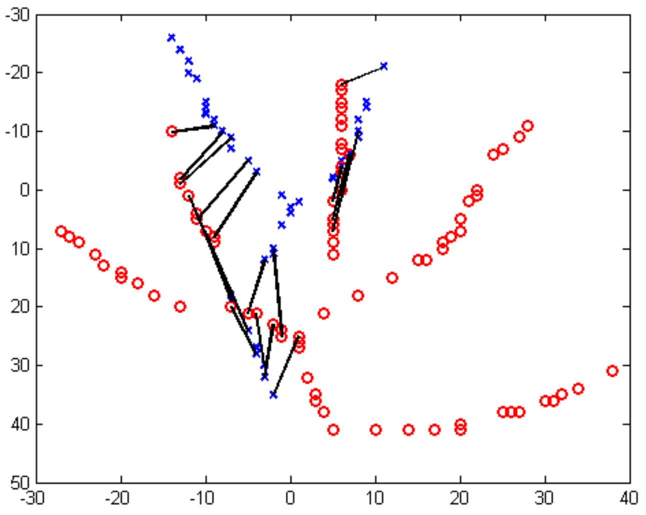
*Figure 6.7: (a) finding the correspondence between similar hand poses using DTW , (b) finding the correspondence between similar hand poses using OSB*

is the open hand. This results together with the number of finger tips are visualized in figure 6.10.

Figure 6.10 does not show the difference between the DTW and OSB approaches for finding the similarity of the hand poses to the source. However,



(a)



(b)

*Figure 6.8: (a) finding the correspondence between different hand poses using DTW , (b) finding the correspondence between different hand poses using OSB*

it is possible to view that the number of the fingertips can be used as one of the features to classify the hand poses as it makes a good separation between hand poses with different number of fingers that are visible. Fingertips are recognized by the application prototype in real-time and with the average

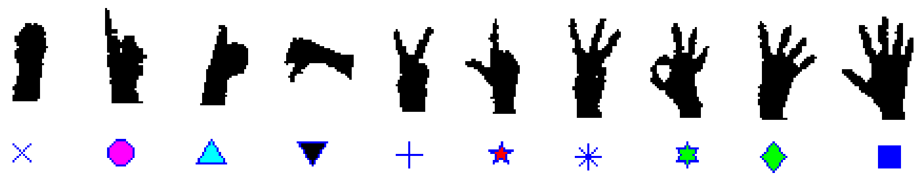
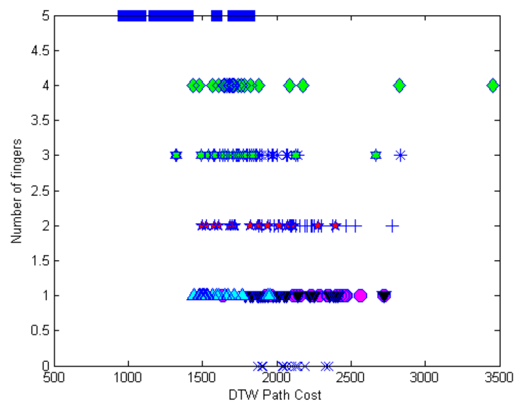
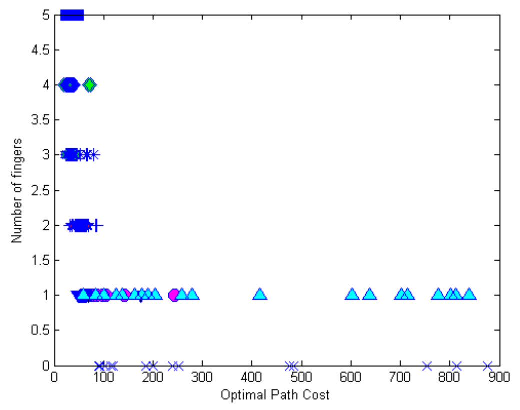


Figure 6.9: The guide for the postures used in the visualization



(a)



(b)

Figure 6.10: (a) Data visualization using DTW similarity and number of fingers, (b) Data visualization using optimal path costs and number of fingers

accuracy of 98%.

The number of subsequent points that can be ignored by the OSB algorithm and to be considered as outliers in both source and target skeletons is defined to be 100. In the experiments considering the number of allowed jumps between [0..150], 100 jumps proved to provide the best results considering the structure and the size of data that are compared.

Three cases have been analyzed:

1. The combination of OSB and DTW distance together with the number of visible fingertips shown in figure 6.11(a).

2. The combination of OSB path cost, and the squared euclidean distance of corresponding skeletons without considering the jump penalty, together with the number of visible fingertips depicted in figure 6.12(a).

3. The combination of OSB path cost, and the jump penalty provided by the OSB algorithm together with the number of visible fingertips that is shown in figure 6.13(a).

For better visibility these data are shown in separate plots for each number of finger tips, in figures 6.11, 6.12 and 6.13 accordingly. As it is possible to view in the visualization results, using OSB path cost and the jump penalty together makes a good separation between the hand poses that have the same number of fingertips. Jump penalty, as explained in Chapter 4, is considered for penalizing the outlier eliminations that are done by the algorithm. The more points that are ignored by the algorithm the higher will be the jump penalty. Therefore, it makes sense to use it as a measure of similarity together with the path cost obtained by the OSB algorithm.

The results of this visualization shows that the results obtained by comparing the specified skeletons with only one source posture, used together with other features can be discriminating enough to let the classification of hand poses to be done accurately. In the experiments OSB algorithm have been concluded to provide more useful and accurate results for classification. This is for two reasons: first, the ability of the algorithm to eliminate the outliers makes it more proper to compare less similar data (e.g. the skeleton of different hand poses). In comparison DTW is more sensitive to outliers since the algorithm compares every two points in the skeletons. Another reason is that OSB can provide more features with discriminating values such as, optimal path cost, jump penalty, euclidean distance without considering the jump penalty, and the number of ignored (jumped) points.

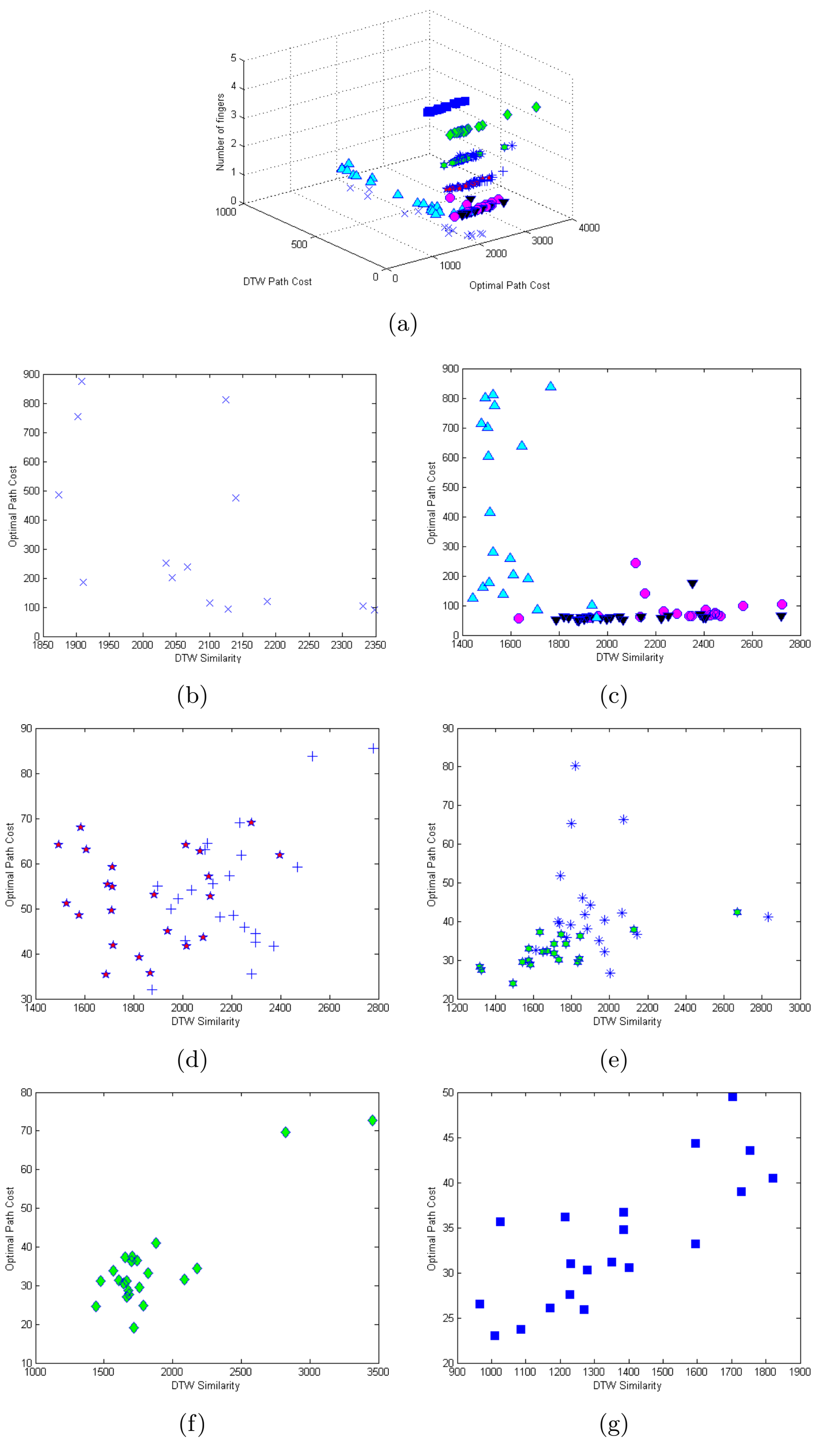
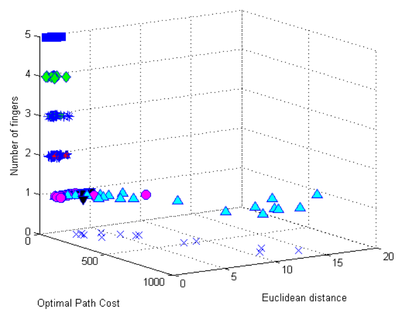
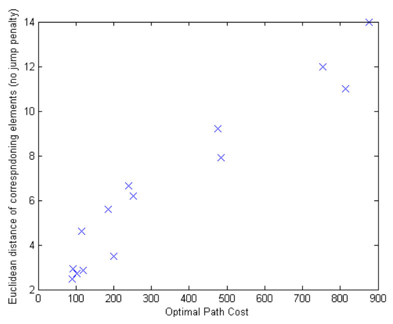


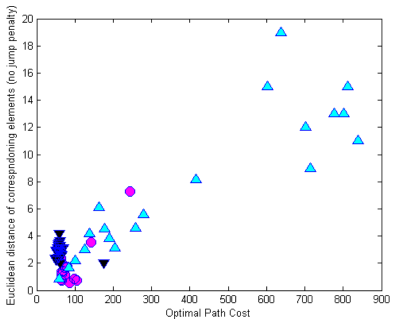
*Figure 6.11: Visualization of results of comparing hand poses using optimal path cost, dtw similarity and number of fingertips (a) is a 3D representation, results separated for each number of visible fingertips from zero to five are shown in (b), (c), (d), (d), (e), (f) and (g) respectively*



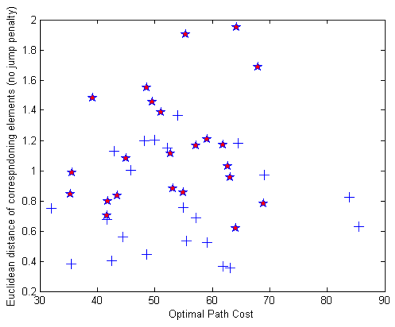
(a)



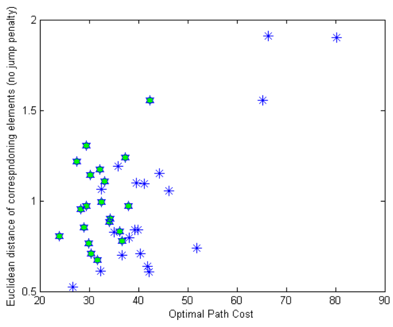
(b)



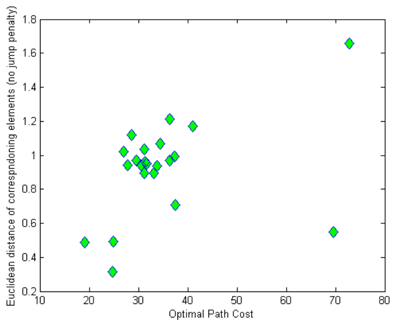
(c)



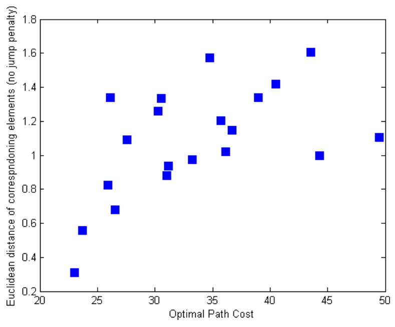
(d)



(e)

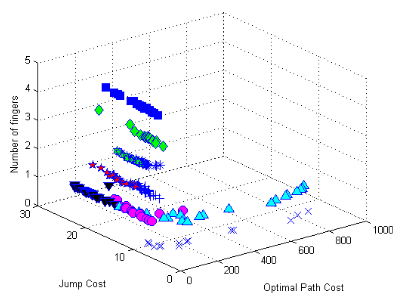


(f)

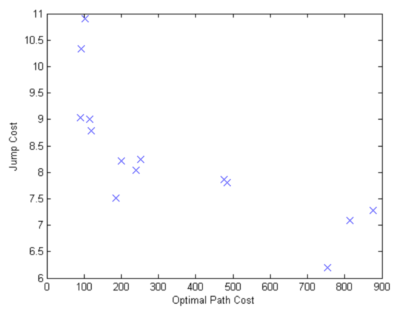


(g)

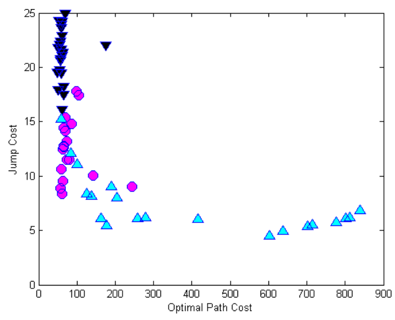
*Figure 6.12: Visualization of results of comparing hand poses using optimal path cost, squared euclidean distance of corresponding elements (no jump penalty) and number of visible fingertips. (a) is a 3D representation, and results separated for each number of visible fingertips from zero to five are shown in (b), (c), (d), (d), (e), (f) and (g) respectively*



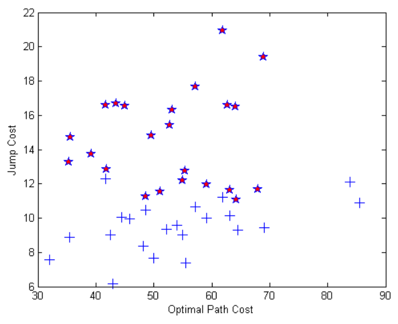
(a)



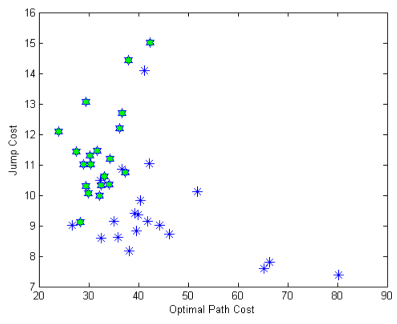
(b)



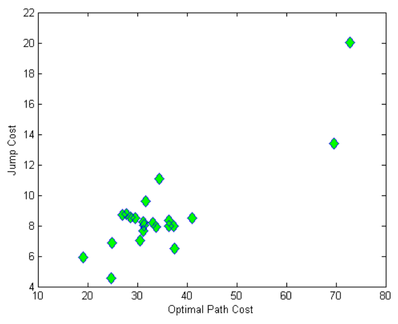
(c)



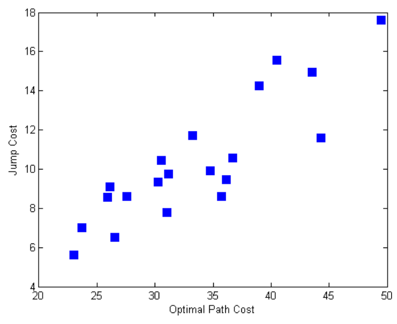
(d)



(e)



(f)



(g)

*Figure 6.13: Visualization of results of comparing hand poses using optimal path cost, jump cost, and number of fingers. (a) is a 3D representation, and results separated for each number of visible fingertips from zero to five are shown in (b), (c), (d), (d), (e), (f) and (g) respectively*

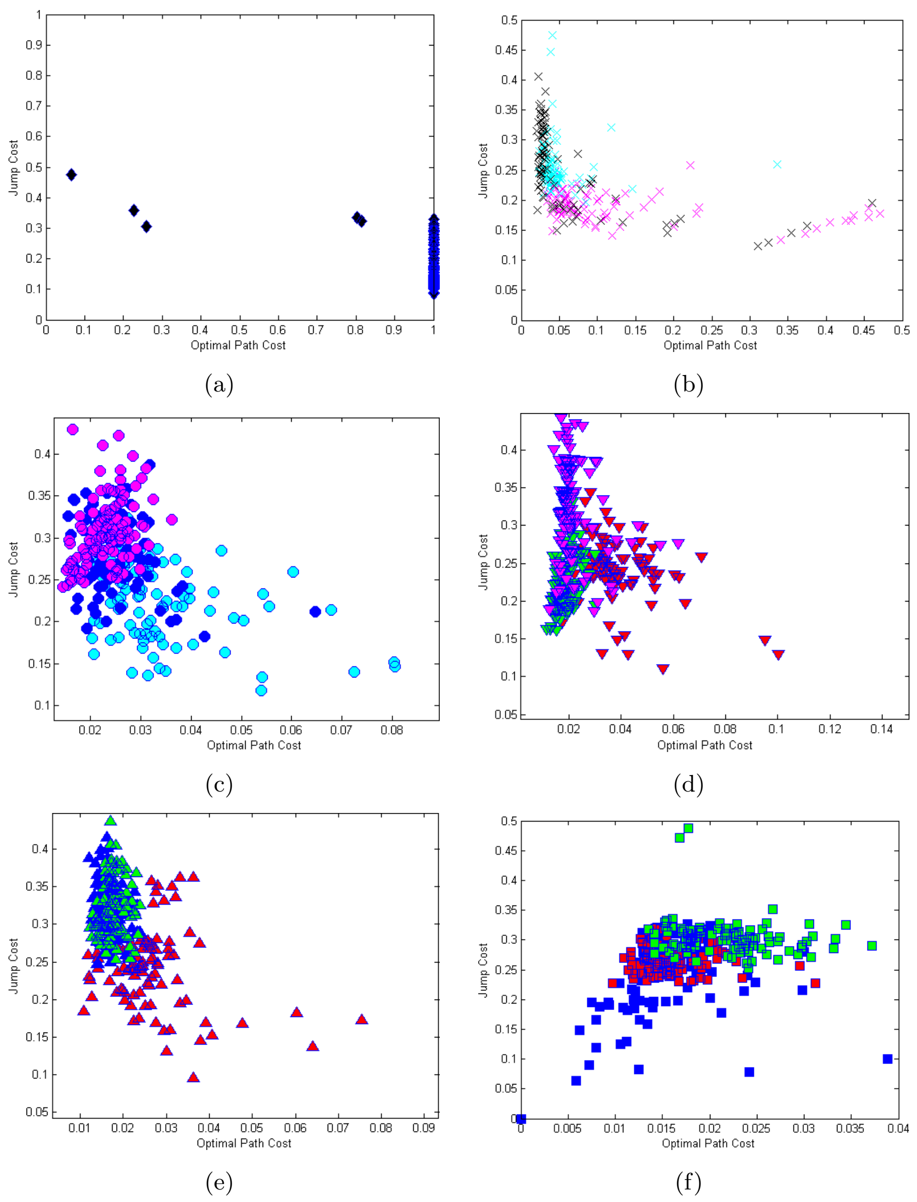


*Figure 6.14: Visualization of results of comparing hand poses using optimal path cost, jump penalty and number of visible fingertips for a set of 21 gestures. The number are fingertips are: (a) zero, (b) one, (c) two, (d) three, (e) four, (f) five*

These visualizations are repeated for a bigger set of hand poses. When the number of hand poses are increased the separation between the poses using only one source pose for comparison will be more difficult. For a good separation, more features will be required, and therefore with a big dataset of hand poses this approach might be costly regarding the complexity and time. Figure 6.14 shows the results for using OSB optimal path cost, jump penalty and the number of fingers for a set of 21 hand poses. Comparing this to the same results obtained for a smaller set of gestures in figure 6.13, it is possible to see that the poses are less classifiable using this approach for a medium sized to big datasets.

### 6.2.3 Skeleton Similarity Using a Bank of Desired Hand Poses as a Source for Comparison

Another approach for hand pose estimation is to define the poses and save a sample from each to be used as the source for comparison. Later, any specified postures can be compared to this bank using DTW or OSB algorithms and the most similar posture will be selected to represent the specified hand pose.

For these tests 21 different poses are considered and 100 samples from each gesture is recorded by a single user. For each sample the Voronoi skeleton is calculated. From each pose sample collection a sample pose is selected as the reference pose. All other samples are compared with these 21 reference poses using DTW and OSB algorithms and the best matched is selected which is the reference pose that is most similar to the sample pose according to DTW and OSB outputs. Three parameters are used and compared in this approach. DTW distance, OSB optimal path cost, and the multiplication of OSB optimal path cost and jump penalty. The accuracy of results results acquired from each of these approaches are compared in table 6.5.

As it was expected, DTW shows an overall better accuracy in this approach in comparison to OSB algorithm. The reason is that OSB ignores many points as outliers that are corresponding to each other. This happens when two similar skeletons are compared. However, in some cases, that because of the nature of the posture there are many differences between the same poses that have been performed, OSB will perform more accurately. This is observable in the row 15th of the table 6.5 in which DTW depicted very low accuracy of 40% and OSB optimal path cost and the multiplication of OSB path cost and jump penalty result in better accuracy of 67% and 76% accordingly. The two approaches that have been used for OSB algorithm the usage of OSB optimal path cost multiplied by jump penalty instead of employing only the optimal path cost have improved the accuracy slightly in some cases, but these improvements are not much, and in some cases it even results in a lower accuracy in comparison to the other approaches.

The results of the DTW algorithm for hand pose estimation are shown

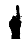
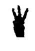
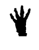
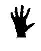
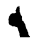
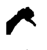
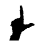
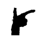
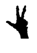
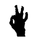
| Hand Pose | DTW | OSB PC | OSB PC * JP |
|---|---|---|---|
| | 100% | 98% | 99% |
|  | 98% | 96% | 77% |
| | 97% | 99% | 84% |
|  | 85% | 50% | 75% |
|  | 89% | 51% | 71% |
|  | 96% | 67% | 80% |
|  | 81% | 41% | 60% |
|  | 92% | 98% | 97% |
|  | 95% | 95% | 91% |
|  | 59% | 55% | 55% |
|  | 91% | 57% | 59% |
|  | 83% | 66% | 77% |
| | 92% | 52% | 84% |
| | 93% | 98% | 83% |
| | 40% | 67% | 76% |
| | 65% | 64% | 55% |
| | 52% | 51% | 46% |
| | 98% | 45% | 43% |
| | 89% | 84% | 85% |
| | 96% | 89% | 89% |
| | 63% | 42% | 36% |

*Table 6.5: Comparison of precision of DTW distance, OSB optimal path cost and OSB optimal path cost multiplied by jump penalty for hand pose estimation*

in a confusion matrix which depicts the percentage that a hand pose is recognized most similar to any of the reference poses.

As the accuracy of the results of DTW shows in table 6.5 and the provided confusion matrix in tables 6.6 and 6.7, DTW performs with good to fair accuracy in most of the tested hand poses. Also, the good performance of this algorithm makes it a proper candidate for real time applications. The standard DTW algorithm that have been used in this thesis has $O(N^2)$ time and memory complexity, with N being the number of points being

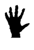
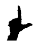
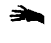
| Pose | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 100 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 2 | 98 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 1 | 97 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| | 0 | 0 | 3 | 85 | 10 | 0 | 1 | 1 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 3 | 89 | 5 | 3 | 0 | 0 | 0 | 0 |
|  | 0 | 0 | 1 | 1 | 1 | 96 | 0 | 0 | 0 | 0 | 1 |
| | 2 | 3 | 0 | 0 | 0 | 0 | 81 | 4 | 4 | 6 | 0 |
| | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 92 | 0 | 0 | 0 |
|  | 0 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 95 | 0 | 0 |
| | 0 | 0 | 13 | 0 | 0 | 0 | 0 | 28 | 0 | 59 | 0 |
| | 0 | 0 | 3 | 2 | 0 | 0 | 0 | 0 | 4 | 0 | 91 |
| | 0 | 0 | 2 | 4 | 0 | 0 | 1 | 2 | 0 | 0 | 4 |
| | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 5 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 5 | 0 | 0 |

*Table 6.6: Confusion matrix in determining 21 different hand poses using DWT (part 1)*

compared. As the skeletons contain 15 to 25 points in average the time complexity would not be high. This algorithm can be optimized to perform with $O(N)$ time and memory complexity [67]. The performance of the pose estimation in this approach using the DTW algorithm have been tested in the application prototype for 21 poses as the comparison sources for each posture that is performed, and the real-time response of the algorithm is confirmed.

Even if DTW has a high accuracy for most of the proposed hand poses,

| Pose | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 6 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 83 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 |
| | 0 | 92 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| | 0 | 0 | 93 | 0 | 0 | 0 | 0 | 6 | 0 | 1 |
| | 0 | 0 | 0 | 40 | 18 | 0 | 0 | 0 | 40 | 2 |
| | 0 | 0 | 0 | 0 | 65 | 0 | 2 | 0 | 33 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 52 | 0 | 38 | 0 | 7 |
| | 0 | 0 | 0 | 0 | 6 | 0 | 98 | 0 | 2 | 0 |
| | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 89 | 0 | 8 |
| | 0 | 0 | 0 | 0 | 2 | 0 | 2 | 0 | 96 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 8 | 0 | 23 | 0 | 63 |

*Table 6.7: Confusion matrix in determining 21 different hand poses using DWT (part 2)*

yet there are some poses that have been mistaken with different poses and the recognition accuracy for them is low. This low accuracy in some cases, is due to the similarity of skeletons to different poses, errors in skeletonization, and also, they highly relate to the nature of the hand pose. As some of the poses that are chosen as the test set are difficult to hold, and the rotation in respect to the camera view and self occlusions occur as the result. Also, another problem that can cause this is again depending on the nature of the hand pose, that makes the same pose skeletons look very different in the performing time. For the usage of elderly, the poses can be selected in a way that they will be easy to perform and memorize. The set of hand poses that was shown as an example in figure 4.9(a) can be a good selection of static, and as it is possible to see in table 6.5 the average accuracy for this set would be 92.29% .

**Improving The Performance of Hand Pose Estimation for Hand Trembling**

As stated before, hand trembling is a common problem in elderly users. It is necessary to note that, here hand trembling pertains to the common hand shakes in old people, and the heavy hand trembling caused by disease such as Parkinson's disease are not covered in this research. Trembling of the hands can cause the increase of the error rate in skeletonization process and therefore in pose estimation. For a better accuracy of pose estimation using DTW algorithm and reducing the error rate in skeletonization process, instead of using one sample of the hand pose, the user have been asked to keep the pose for one second and the hand pose skeleton have been sampled 10 times. The sampling numbers can be calibrated according to heaviness of hand shakes. Pose estimation is done for these samples and the estimated pose that has been repeated the most in the estimation process is chosen. This process is tested with the application prototype for the set of hand postures shown in figure 4.9(a) and the average accuracy of 89.3% have been obtained by simulating the hand shakes by a normal user.

## 6.3 Static Gesture Recognition

Static gestures pertain to the gestures that move from one posture to another. The trajectory of the hands are not considered in these kinds of gestures. Static gestures are proper for the elderly users with course hand movement problem that feel pain or cannot perform the gestures that need movements of elbows and shoulders. Also, it can be used in the presence of hand trembling. However, if the trembling is heavy, and the course hand movement is not present, it is more advised to use dynamic gestures with a good difference between the pathology of gestures in the set. As it was evaluated in section 6.2.3, it was seen that DTW algorithm might show inaccuracies in estimating some of the postures. Also, because of the hand shakes, and the movements during performing of the gestures the error rate of DTW algorithm for pose estimation can be increased. All of the problems mentioned above can be solved by using HMMs for gesture recognition as

HMMs can be trained to handle some range of errors caused by inaccuracies in DTW pose estimation.

Five static gestures shown in figure 6.15 have been chosen for test and evaluation of the HMM-based gesture recognition system. The gestures are chosen in a way that:

1. Very similar poses regarding their skeletons exist in the gesture.

2. Some of the poses in the gestures might not exist in the hand pose bank used for pose estimation.

3. The poses with low estimation accuracy by the DTW exists in the gesture

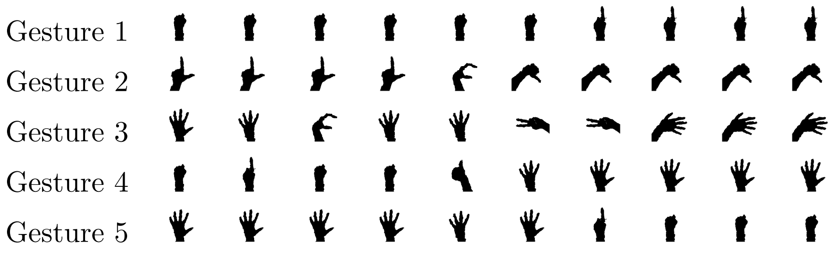
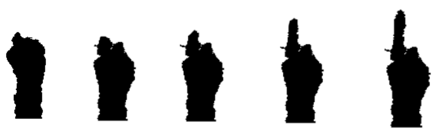
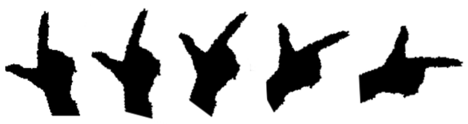


*Table 6.8: Observation sequence of 10 hand poses sampled during performing of gestures by the hand pose estimator*

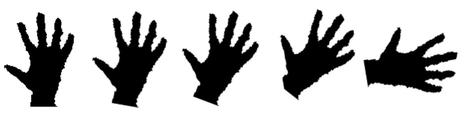
These gestures have been recorded by a single user as isolated gestures and then used for training and testing of HMMs. The gestures are considered to have fixed lengths and are sampled 10 times during 1.5 seconds. A sample of the observation sequence for each gesture that is obtained by the pose estimator is shown is table 6.8 The number of sampling and the duration of the gestures can be defined due to the nature of the gestures and the heaviness of the user's hand trembling. The experiments showed that for these set of gestures 3-state HMMs performs well. Number of symbols are set to 21 as the input of the HMMs is a sequence of 10 hand poses that are sampled and estimated by the pose estimation system using DTW algorithm, and the defined poses for the hand pose estimator consist of 21 hand poses. 5 left-right HMMs are trained, each for one gesture. Using 20 training data for each of the gestures the recognition accuracy represented in table 6.9 is acquired. Later, number of training data is increased to 30 to improve the recognition accuracy.



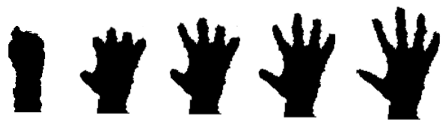
(a)



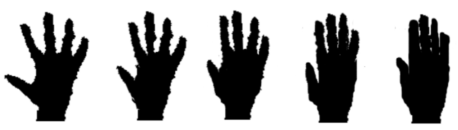
(b)



(c)



(d)



(e)

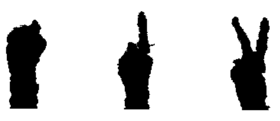
*Figure 6.15: Set of gestures for test: (a) gesture 1, (b) gesture 2, (c) gesture 3, (d) gesture 4, (e) gesture 5*

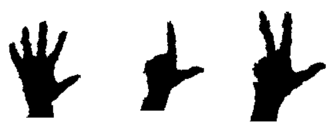
| Gesture | Recognition accuracy: with 20 training data | with 30 training data |
|---|---|---|
| Gesture 1 | 100% | 100% |
| Gesture 2 | 90% | 99% |
| Gesture 3 | 90% | 95% |
| Gesture 4 | 88% | 98% |
| Gesture 5 | 100% | 100% |

*Table 6.9: Recognition accuracy of trained HMMs for the test gestures*

### 6.3.1 Recognizing a Sequence of Static Hand Postures

A sequence of static hand postures can be trained to be recognized as a gesture. Since the hand pose estimator has high accuracy in recognizing the static hand poses may be training the HMMs for this purpose would not be even necessary. However, with very little number of training very high accuracy can be resulted by the HMM-based gesture recognizer and the errors presented by the pose estimator can be compensated. Three sequences of hand poses that are shown in figure 6.16 have been defined and trained with 10 training data for each gesture.


(a)


(b)

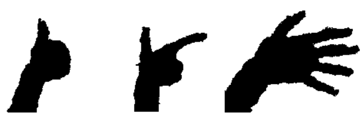

(c)

*Figure 6.16: Set of gestures as the sequence of hand poses: (a) Sequence 1, (b) Sequence 2, (c) Sequence 3*

The recognition accuracy for the pose sequences in figure 6.16 are listed in table 6.10.

| Gesture | Recognition accuracy with 10 training data |
|---|---|
| Sequence 1 | 100% |
| Sequence 2 | 98% |
| Sequence 3 | 97% |

*Table 6.10: Recognition accuracy of trained HMMs for the test gestures*

Chapter 7

# Conclusions and Future Work

This work has presented an approach for a flexible hand pose and static gesture recognition system that can be adapted to the special needs of specific users, like the elderly. At this aim, a classification for elderly hand posture and movement limitations has been first identified, by reviewing the studies in this area. The approaches offered and employed in this work take note of these limitations and try to overcome them to implement a HCI system based on hand pose and static gesture recognition that is usable and convenient for this group of users.

The advantages of this work can be summarized as follows: The system introduced in this work responds in real-time which is of great importance in HCI applications. Microsoft Kinect depth sensor is employed because of the affordable price, easiness of setting up and its efficiency in extracting the tracking information of the user body parts. Because of the usage of depth sensor that employs infrared light the system can perform also in low lighting situations. For background elimination, the depth data and the position of the center palm are used. This approach makes the system to be independent of background changes and even to respond well in case of dynamic backgrounds. Another advantage of this approach is that the user would not need to wear any colored gloves, wrist bands or long sleeves for correct hand segmentation.

An application prototype has been designed and developed for testing and experimenting the gesture recognition and pose estimation algorithms in real-time. This application also enables the recording of the data needed for further tests and evaluations offline. The hand pose estimation system is flexible to define the desired poses with minimal effort, only by recording a sample of each hand pose in the source posture bank for comparison and recognition of later performed poses. To improve the accuracy of pose recognition in presence of hand trembling, which is common in the elderly, several samples are taken in one second, while the user holds the posture. Another advantage of the proposed approach that distinguishes it from the other works in the literature, is the usage of the hand poses in gesture recognition. Static gestures are adopted, which consist in changing the hand posture between different poses. These kind of gestures do not include movements of the elbow and the shoulders which are difficult or painful to perform by elderly, specially by the persons with coarse hand movement limitation. The gesture recognition system is configurable for the definition of different static gestures upon the user's convenience. The use of HMMs for gesture recognition makes up for the shortcomings of the pose estimator in recognizing some of the hand poses. It can also recognize the gestures including the poses that do not exist in the posture bank or is estimated incorrectly by the pose estimator. In average the accuracy of 98.4% is presented by the proposed static gesture recognition system by using only 30 training data for 5 test gestures that include the poses that are challenging for the pose estimator to recognize correctly.

For feature extraction, Voronoi Diagrams on boundary points are used to extract of the skeleton of hand. To compare the skeletons two algorithms have been tested and compared: OSB (Optimal Subsequence Bijection) and DTW (Dynamic Time Warping). In both cases, two approaches have been tested for hand pose estimation. The first approach is to compare each performed pose with a single source pose (an open hand) and then for better separation of the pose classes employing other features like number of fingertips, width and height of the pose which are provided by the application. The other approach was to compare the skeletons of performed poses with a bank of defined hand poses and choose the most similar as the estimated pose. OSB showed better results in the former approach, and resulted in better separation of the pose classes. However, the results showed that the first approach would not be accurate with a larger set consisting of 21 hand poses as it was evaluated in this work. DTW was proved to perform with better accuracy in the latter approach and was eventually used for hand pose estimation in this work. The average accuracy of 92.29% is showed by the algorithm on a set of seven hand poses including the fist, finger counting from one to five and the thumb up postures. However, by adding some challenging hand poses, the average accuracy of 83.52% is obtained for a set of 21 poses.

As future work, the shortcomings of the hand pose estimator for some poses will be improved by considering other discriminating features in the pose recognition. Moreover, the proposed prototype will be tested with the actual target users (the elderly) to improve the system to better meet their needs. The static gesture recognition can be extended to consider also hand trajectories, to be able to recognize a larger variety of gestures. Further extension of this system may include also facial expression recognition and vocal orders to support an even wider disabilities and limitations that are present in elderly or disabled people.

# Bibliography

[1] Assistive technologies group (atg), politecnico di milano, http://atg.deib.polimi.it/. Accessed: 2013.09.08.

[2] Dynamic time warping for matlab, http://www.mathworks.com/matlabcentral/fileexchange/6516-dynamic-time-warping/content/dtw.m. Accessed: 2013-08-30.

[3] Hidden markov model toolbox for matlab, http://www.cs.ubc.ca/ murphyk/software/hmm/hmm.html. Accessed: 2013-08-30.

[4] Nite official website, http://www.openni.org/files/nite/. Accessed: 2013-08-30.

[5] Openni official website, http://www.openni.org/. Accessed: 2013-08-30.

[6] Tiptep skeletonizer official website, http://www.openni.org/files/tiptep-skeletonizer/. Accessed: 2013-08-30.

[7] Alexey Abramov, Karl Pauwels, Jeremie Papon, Florentin Worgotter, and Babette Dellen. Depth-supported real-time video segmentation with the kinect. In *Applications of Computer Vision (WACV), 2012 IEEE Workshop on*, pages 457–464. IEEE, 2012.

[8] Dominique Attali and Annick Montanvert. Computing and simplifying 2d and 3d continuous skeletons. *Computer Vision and Image Understanding*, 67(3):261–273, 1997.

[9] Franz Aurenhammer. Voronoi diagramsÑa survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, 23(3):345–405, 1991.

[10] Xiang Bai, Longin Jan Latecki, and Wen-Yu Liu. Skeleton pruning by contour partitioning with discrete curve evolution. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(3):449–462, 2007.

[11] Raimo Bakis. Continuous speech recognition via centisecond acoustic states. *The Journal of the Acoustical Society of America*, 59:S97, 1976.

[12] Leonard E Baum and George R Sell. Growth transformations for functions on manifolds. *Pacific J. Math*, 27(2):211–227, 1968.

[13] Donald J Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, 1994.

[14] Nguyen Dang Binh, Enokida Shuichi, and Toshiaki Ejima. Real-time hand tracking and gesture recognition system. *Proc. GVIP*, pages 19–21, 2005.

[15] Harry Blum et al. A transformation for extracting new descriptors of shape. *Models for the perception of speech and visual form*, 19(5):362–380, 1967.

[16] Jan Bobeth, Susanne Schmehl, Ernst Kruijff, Stephanie Deutsch, and Manfred Tscheligi. Evaluating performance and acceptance of older adults using freehand gestures for tv menu control. In *Proc. of the 10th European conference on Interactive tv and video*, pages 35–44. ACM, 2012.

[17] Jonathan W Brandt and V Ralph Algazi. Continuous skeleton computation by voronoi diagram. *CVGIP: Image Understanding*, 55(3):329–338, 1992.

[18] Matthieu Bray, Esther Koller-Meier, and Luc Van Gool. Smart particle filtering for 3d hand tracking. In *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*, pages 675–680. IEEE, 2004.

[19] Angela Brennecke and Tobias Isenberg. 3d shape matching using skeleton graphs. In *SimVis*, pages 299–310. Citeseer, 2004.

[20] Eli Carmeli, Hagar Patish, and Raymond Coleman. The aging hand. *The Journals of Gerontology Series A: Biological Sciences and Medical Sciences*, 58(2):M146–M152, 2003.

[21] Feng-Sheng Chen, Chih-Ming Fu, and Chung-Lin Huang. Hand gesture recognition using a real-time tracking method and hidden markov models. *Image and Vision Computing*, 21(8):745–758, 2003.

[22] Nicu D Cornea, Deborah Silver, and Patrick Min. Curve-skeleton properties, applications, and algorithms. *Visualization and Computer Graphics, IEEE Transactions on*, 13(3):530–548, 2007.

[23] Mark De Berg, Marc Van Kreveld, Mark Overmars, and Otfried Cheong Schwarzkopf. *Computational geometry*. Springer, 2000.

[24] Herbert Edelsbrunner. *Algorithms in combinatorial geometry*, volume 10. Springer, 1987.

[25] Ali Erol, George Bebis, Mircea Nicolescu, Richard D Boyle, and Xander Twombly. Vision-based hand pose estimation: A review. *Computer Vision and Image Understanding*, 108(1):52–73, 2007.

[26] Ricardo Fabbri, Leandro Farias Estrozi, and L Da F Costa. On voronoi diagrams and medial axes. *Journal of Mathematical Imaging and Vision*, 17(1):27–40, 2002.

[27] Yikai Fang, Jian Cheng, Kongqiao Wang, and Hanqing Lu. Hand gesture recognition using fast multi-scale analysis. In *Image and Graphics, 2007. ICIG 2007. Fourth International Conference on*, pages 694–698. IEEE, 2007.

[28] G David Forney Jr. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[29] William T Freeman and Michal Roth. Orientation histograms for hand gesture recognition. In *International Workshop on Automatic Face and Gesture Recognition*, volume 12, pages 296–301, 1995.

[30] Pragati Garg, Naveen Aggarwal, and Sanjeev Sofat. Vision based hand gesture recognition. *World Academy of Science, Engineering and Technology*, 49(1):972–977, 2009.

[31] Konstantinos Giannakouris. Regional population projections europop2008: Most eu regions face older population profile in 2030. *Population and social conditions, Eurostat Statistics in Focus*, 1, 2010.

[32] Juha Häikiö, Arto Wallin, Minna Isomursu, Heikki Ailisto, Tapio Matinmikko, and Tua Huomo. Touch-based user interface for elderly users. In *Proc. of the 9th international conference on Human computer interaction with mobile devices and services*, pages 289–296. ACM, 2007.

[33] Tony Heap and David Hogg. Towards 3d hand tracking using a deformable model. In *Automatic Face and Gesture Recognition, 1996., Proceedings of the Second International Conference on*, pages 140–145. IEEE, 1996.

[34] Pengyu Hong, Matthew Turk, and Thomas S Huang. Gesture modeling and recognition using finite state machines. In *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on*, pages 410–415. IEEE, 2000.

[35] Kazuyuki Imagawa, Shan Lu, and Seiji Igi. Color-based hands tracking system for sign language recognition. In *Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on*, pages 462–467. IEEE, 1998.

[36] C Keskin, A Erkan, and L Akarun. Real time hand tracking and 3d gesture recognition for interactive interfaces using hmm. *ICANN/I-CONIPP*, 2003:26–29, 2003.

[37] Amlan Kundu, Yang He, and Paramvir Bahl. Recognition of handwritten word: first and second order hidden markov model based approach. *Pattern recognition*, 22(3):283–297, 1989.

[38] Longin Jan Latecki, Qiang Wang, Suzan Koknar-Tezel, and Vasileios Megalooikonomou. Optimal subsequence bijection. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 565–570. IEEE, 2007.

[39] Der-Tsai Lee. Medial axis transformation of a planar shape. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (4):363–369, 1982.

[40] Hyeon-Kyu Lee and Jin-Hyung Kim. An hmm-based threshold model approach for gesture recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(10):961–973, 1999.

[41] Frédéric Leymarie and Martin D Levine. Simulating the grassfire transform using an active contour model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(1):56–75, 1992.

[42] Xiaolin Li, Marc Parizeau, and Réjean Plamondon. Training hidden markov models with multiple observations-a combinatorial method. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(4):371–377, 2000.

[43] Attila Licsár and Tamás Szirányi. User-adaptive hand gesture recognition system with interactive training. *Image and Vision Computing*, 23(12):1102–1114, 2005.

[44] André Lieutier. Any open bounded subset of rn has the same homotopy type as its medial axis. *Computer-Aided Design*, 36(11):1029–1046, 2004.

[45] Nianjun Liu, Brian C Lovell, Peter J Kootsookos, and Richard IA Davis. Model structure selection & training algorithms for an hmm gesture recognition system. In *Frontiers in Handwriting Recognition, 2004. IWFHR-9 2004. Ninth International Workshop on*, pages 100–105. IEEE, 2004.

[46] Xia Liu and Kikuo Fujimura. Hand gesture recognition using depth data. In *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*, pages 529–534. IEEE, 2004.

[47] Ray Lockton. Hand gesture recognition using computer vision. *4th Year Project Report-Oxford University*, 2001.

[48] Sushmita Mitra and Tinku Acharya. Gesture recognition: A survey. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 37(3):311–324, 2007.

[49] MA Moni and ABM Shawkat Ali. Hmm based hand gesture recognition: A review on techniques and approaches. In *Computer Science and Information Technology, 2009. ICCSIT 2009. 2nd IEEE International Conference on*, pages 433–437. IEEE, 2009.

[50] Ugo Montanari. Continuous skeletons from digitized images. *Journal of the ACM (JACM)*, 16(4):534–549, 1969.

[51] GRS Murthy and RS Jadon. A review of vision based hand gestures recognition. *International Journal of Information Technology and Knowledge Management*, 2(2):405–410, 2009.

[52] Elizabeth D Mynatt, A-S Melenhorst, A-D Fisk, and Wendy A Rogers. Aware technologies for aging in place: understanding user needs and attitudes. *Pervasive Computing, IEEE*, 3(2):36–41, 2004.

[53] Kai Nickel and Rainer Stiefelhagen. Pointing gesture recognition based on 3d-tracking of face, hands and head orientation. In *Proceedings of the 5th international conference on Multimodal interfaces*, pages 140–146. ACM, 2003.

[54] Rachel Nuwer. Armband adds a twitch to gesture control. *New Scientist*, 217(2906):21, 2013.

[55] R Ogniewicz and M Ilg. Voronoi skeletons: Theory and applications. In *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR'92., 1992 IEEE Computer Society Conference on*, pages 63–69. IEEE, 1992.

[56] Robert L Ogniewicz and Olaf Kübler. Hierarchic voronoi skeletons. *Pattern recognition*, 28(3):343–359, 1995.

[57] Iason Oikonomidis, Nikolaos Kyriazis, and Antonis Argyros. Efficient model-based 3d tracking of hand articulations using kinect. In *British Machine Vision Conference*, pages 101–1, 2011.

[58] Kenji Oka, Yoichi Sato, and Hideki Koike. Real-time fingertip tracking and gesture recognition. *Computer Graphics and Applications, IEEE*, 22(6):64–71, 2002.

[59] Atsuyuki Okabe, Barry Boots, Kokichi Sugihara, and Sung Nok Chiu. *Spatial tessellations: concepts and applications of Voronoi diagrams*, volume 501. Wiley. com, 2009.

[60] Vladimir I Pavlovic, Rajeev Sharma, and Thomas S. Huang. Visual interpretation of hand gestures for human-computer interaction: A review. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(7):677–695, 1997.

[61] Franco P Preparata and Michael Ian Shamos. *Geometric Searching.* Springer, 1985.

[62] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[63] ASM Rahman, M Anwar Hossain, Jorge Parra, and Abdulmotaleb El Saddik. Motion-path based gesture interaction with smart home services. In *Proc. of the 17th ACM international conference on Multimedia*, pages 761–764. ACM, 2009.

[64] Aditya Ramamoorthy, Namrata Vaswani, Santanu Chaudhury, and Subhashis Banerjee. Recognition of dynamic hand gestures. *Pattern Recognition*, 36(9):2069–2081, 2003.

[65] Vinoth K Ranganathan, Vlodek Siemionow, Vinod Sahgal, and Guang H Yue. Effects of aging on hand function. *Journal of the American Geriatrics Society*, 49(11):1478–1484, 2001.

[66] Gerhard Rigoll, Andreas Kosmala, and Stefan Eickeler. High performance real-time gesture recognition using hidden markov models. In *Gesture and Sign Language in Human-Computer Interaction*, pages 69–80. Springer, 1998.

[67] Stan Salvador and Philip Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007.

[68] Yoichi Sato, Yoshinori Kobayashi, and Hideki Koike. Fast tracking of hands and fingertips in infrared images for augmented desk interface. In *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on*, pages 462–467. IEEE, 2000.

[69] Jennifer Schlenzig, Edd Hunter, and Ramesh Jain. Recursive identification of gesture inputs using hidden markov models. In *Applications of Computer Vision, 1994., Proceedings of the Second IEEE Workshop on*, pages 187–194. IEEE, 1994.

[70] Steven C Seow, Dennis Wixon, Ann Morrison, and Giulio Jacucci. Natural user interfaces: the prospect and challenge of touch and gestural computing. In *Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems*, pages 4453–4456. ACM, 2010.

[71] Doron Shaked and Alfred M Bruckstein. Pruning medial axes. *Computer vision and image understanding*, 69(2):156–169, 1998.

[72] Lei Shi, Yangsheng Wang, and Jituo Li. A real time vision-based hand gestures recognition system. In *Advances in Computation and Intelligence*, pages 349–358. Springer, 2010.

[73] Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124, 2013.

[74] Raymond W Smith. Computer processing of line images: a survey. *Pattern recognition*, 20(1):7–15, 1987.

[75] Yale Song and Ying Yin. Sign language recognition. 2013.

[76] Thad Starner, Joshua Weaver, and Alex Pentland. Real-time american sign language recognition using desk and wearable computer based video. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(12):1371–1375, 1998.

[77] Bjoern Stenger, Paulo RS Mendonça, and Roberto Cipolla. Model-based 3d tracking of an articulated hand. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 2, pages II–310. IEEE, 2001.

[78] Jesus Suarez and Robin R Murphy. Hand gesture recognition with depth images: A review. In *RO-MAN, 2012 IEEE*, pages 411–417. IEEE, 2012.

[79] Hari Sundar, Deborah Silver, Nikhil Gagvani, and S Dickinson. Skeleton based shape matching and retrieval. In *Shape Modeling International, 2003*, pages 130–139. IEEE, 2003.

[80] Matthew Tang. Recognizing hand gestures with microsoftÕs kinect. *Palo Alto: Department of Electrical Engineering of Stanford University:[sn]*, 2011.

[81] Jochen Triesch and Christoph von der Malsburg. Classification of hand postures against complex backgrounds using elastic graph matching. *Image and Vision Computing*, 20(13):937–943, 2002.

[82] Pedro Trindade, Jorge Lobo, and Joao P Barreto. Hand gesture recognition using color and depth images enhanced with hand angular pose data. In *Multisensor Fusion and Integration for Intelligent Systems (MFI), 2012 IEEE Conference on*, pages 71–76. IEEE, 2012.

[83] UNPD UN. World population ageing: 1950-2050. *United Nations, New York*, 2002.

[84] Nahapetyan Vahagn. Asl fingerspelling recognition. *PFUR Bulletin. Series: mathematics, computer science, physics*, (2):105–113, 2013.

[85] Michael Van den Bergh and Luc Van Gool. Combining rgb and tof cameras for real-time 3d hand gesture interaction. In *Applications of Computer Vision (WACV), 2011 IEEE Workshop on*, pages 66–72. IEEE, 2011.

[86] Andrew Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *Information Theory, IEEE Transactions on*, 13(2):260–269, 1967.

[87] Christian Vogler and Dimitris Metaxas. Asl recognition based on a coupling between hmms and 3d motion analysis. In *Computer Vision, 1998. Sixth International Conference on*, pages 363–369. IEEE, 1998.

[88] Christian Vogler and Dimitris Metaxas. Parallel hidden markov models for american sign language recognition. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 116–122. IEEE, 1999.

[89] Christian Vogler and Dimitris Metaxas. A framework for recognizing the simultaneous aspects of american sign language. *Computer Vision and Image Understanding*, 81(3):358–384, 2001.

[90] Juan Pablo Wachs, Mathias Kölsch, Helman Stern, and Yael Edan. Vision-based hand-gesture applications. *Communications of the ACM*, 54(2):60–71, 2011.

[91] Andrew D. Wilson and Aaron F. Bobick. Parametric hidden markov models for gesture recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(9):884–900, 1999.

[92] Andrew D Wilson and Aaron F Bobick. Realtime online adaptive gesture recognition. In *Pattern Recognition, 2000. Proc. 15th International Conference on*, volume 1, pages 270–275. IEEE, 2000.

[93] Ying Wu and Thomas S Huang. Vision-based gesture recognition: A review. *Urbana*, 51:61801, 1999.

[94] Ying Wu, John Y Lin, and Thomas S Huang. Capturing natural hand articulation. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 2, pages 426–432. IEEE, 2001.

[95] Junji Yamato, Jun Ohya, and Kenichiro Ishii. Recognizing human action in time-sequential images using hidden markov model. In *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR'92., 1992 IEEE Computer Society Conference on*, pages 379–385. IEEE, 1992.

[96] Zhong Yang, Yi Li, Yang Zheng, Weidong Chen, and Xiaoxiang Zheng. An interaction system using mixed hand gestures. In *Proc. of the 10th asia pacific conference on Computer human interaction*, pages 125–132. ACM, 2012.

[97] Yu Yuan, Ying Liu, and Kenneth Barner. Tactile gesture recognition for people with disabilities. In *Acoustics, Speech, and Signal Processing, 2005. Proc.(ICASSP'05). IEEE International Conference on*, volume 5, pages v–461. IEEE, 2005.

[98] Hong-Min Zhu and Chi-Man Pun. Real-time hand gesture recognition from depth image sequences. In *Computer Graphics, Imaging and Visualization (CGIV), 2012 Ninth International Conference on*. IEEE, 2012.

# Appendix A

# Matlab Codes For Evaluation and Testing

In this appendix the MATLAB codes that are used for evaluation and testing are added

## A.1 DTW Algorithm

As the implementation of DTW algorithm, the implementation by Timothy Felty is used [2]. However, to adapt it to compare skeletons some changes have been made to the original source code. Listing A.1 shows the code that was used for calculating the DTW distance between two point sequences that belong to the source and the target skeletons.

```matlab
function [Dist,D,k,w]=dtw(t,r)
    %Dynamic Time Warping Algorithm
    %Dist is unnormalized distance between t and r
    %D is the accumulated distance matrix
    %k is the normalizing factor
    %w is the optimal path
    %t is the vector you are testing against
    %r is the vector you are testing
    [M,¬]=size(t);
    [N,¬]=size(r);
    for n=1:N
        for m=1:M
            d(n,m)= sqrt((t(m,1)−r(n,1))^2 + ...
                (t(m,2)−r(n,2))^2);
        end
    end
    D=zeros(size(d));
    D(1,1)=d(1,1);
    for n=2:N
        D(n,1)=d(n,1)+D(n-1,1);
    end
    for m=2:M
        D(1,m)=d(1,m)+D(1,m-1);
    end
    for n=2:N
        for m=2:M
            D(n,m)=d(n,m)+min([D(n-1,m),D(n-1,m-1),D(n,m-1)]);
        end
    end
    Dist=D(N,M);
    n=N;
    m=M;
    k=1;
    w=[];
    w(1,:)=[N,M];
    while ((n+m)≠2)
        if (n-1)==0
            m=m-1;
        elseif (m-1)==0
            n=n-1;
        else
            [values,number]=min([D(n-1,m),D(n,m-1),D(n-1,m-1)]);
            switch number
                case 1
                    n=n-1;
                case 2
                    m=m-1;
                case 3
                    n=n-1;
                    m=m-1;
            end
        end
        k=k+1;
        w=cat(1,w,[n,m]);
    end

    xt1 = t(:,1);
    yt1 = t(:,2);
    plot(yt1,xt1,'bx','LineWidth',2);
    hold on
    xt2 = r(:,1);
    yt2 = r(:,2);
    plot(yt2,xt2,'ro','LineWidth',2);
```

## A.2 OSB Algorithm

The OSB code in MATLAB and C++ was provided by the writers of [38]. The original algorithm is changed to adapt it to comparing two skeletons. Listing A.2 shows the codes for this algorithm.

```
% Longin Jan Latecki, latecki@temple.edu, June 2006; ...
    updated Jan 2010
% Updated June 2010 Suzan Koknar–Tezel, tezel@sju.edu
% Parameters:
%   t1 – the first (query) time series. A row vector.
%   t2 – the second (target) time series. A row vector.
%
% Optional Parameters:
%   warpwin – the warping window, same as that used in DTW
%   queryskip – a restriction on how many consecutive ...
    elements of sequence t1 can be skipped
%   targetskip – a restriction on how many consecutive ...
    elements of sequence t2 can be skipped
%             best set  warpwin=queryskip=targetskip
%   jumpcost – the cost of jumps in t1 or t2 used to ...
    balance the similarity of elements.
%   if jumpcost==–1 or not present, then it will be estimated
% Return values:
%    pathcost – the cost of the cheapest matching path ...
    between t1 and t2;
%               it is a combination of distances between ...
    corresponding elements
%                and the panelties for jumping (skipping)
%    indxrow – the index of elements in a substring of t1 ...
    that best match t2
%    indxcol – the index of elements in a substring of t2 ...
    that best match t1
%    d  – squared Euclidean distance of correspndoning ...
    elements (no jump penalty)
%
% Example:
%    t1=[ 1 2 8 12 6 8]; t2=[ 1 2 9  3 3  5 9];
%    t1=[ 1 2 8 12 6 8.5]; t2=[ 1 2 9  3 3  5.5 9];
%    [pathcost,indxrow,indxcol,jumpcost, d] = ...
    OSBv5(t1,t2,5,5,5,–1,1)
%    %jumpcost is determined automatically in this example

function [pathcost,indxrow,indxcol,jumpcost,d] = ...
    OSBv5(t1,t2,warpwin,queryskip,targetskip,jumpcost,fig)

[m,¬]=size(t1);
```

```
[n,¬]=size(t2);

%if warpwin is not given then
if ¬exist('warpwin')
    warpwin = Inf;
end
%if targetskip is not given then
if ¬exist('queryskip')
    queryskip = Inf;
end
%if queryskip is not given then
if ¬exist('targetskip')
    targetskip = Inf;
end

matx = zeros(m,n);
% create a distance matrix, each entry i,j contains squared ...
    Euclidean distance of i in t1 and j in t2
for i=1:m
    for j=1:n
        matx(i,j) = sqrt((t2(j,1) − t1(i,1))^2 + (t2(j,2) − ...
            t1(i,2))^2) ;
    end
end

%computing the jumpcost, it is symmetric
if ¬exist('jumpcost') || jumpcost==−1
    statmatx=min(matx');
    statmatx2=min(matx);
    jumpcost=min( mean(statmatx)+std(statmatx), ...
        mean(statmatx2)+std(statmatx2) )+ eps;
end

matxE = Inf(m+2,n+2);
%we add one row and one column to beginning and end of matx
%to ensure that we can skip the first and last elements
matxE(2:m+1,2:n+1)=matx;
matxE(1,1)=0;
matxE(m+2,n+2)=0;

% calling the main function
[pathcost,indxrow,indxcol] = ...
    findpathDAG(matxE,warpwin,queryskip,targetskip,jumpcost);
%we normalize path cost
pathcost=pathcost/length(indxrow);
%squared Euclidean distance of correspndoning elements
d = sqrt(sum((t1(indxrow)−t2(indxcol)).^2))/length(indxrow);

xt1 = t1(:,1);
```

```
yt1 = t1(:,2);
plot(yt1,xt1,'bx','LineWidth',2);
hold on
xt2 = t2(:,1);
yt2 = t2(:,2);
plot(yt2,xt2,'ro','LineWidth',2);

for i=1:length(indxrow)
    hold on;
    line([t1(indxrow(i),2) ...
        t2(indxcol(i),2)],[t1(indxrow(i),1) ...
        t2(indxcol(i),1)],'Color','k','LineWidth',2);
end
return

%------------------------------------------------------------------
% function [pathcost,indxrow,indxcol] = ...
    findpathDAG(matx,warpwin,queryskip,targetskip,jumpcost)
% finds path with min cost
% input: the extended difference matrix
% output: the cost of cheapest path, and indices of the ...
    cheapest path
% The cheapest path is computed on DAG
%------------------------------------------------------------------
function [pathcost,indxrow,indxcol] = ...
    findpathDAG(matx,warpwin,queryskip,targetskip,jumpcost)

%this matx=matxE, thus it has one row an column more than ...
    the original matx above
[m,n]=size(matx);
%the weight of the actually cheapest path
weight=Inf(m,n);
%the index of the parent col where the cheapest path came from
camefromcol=zeros(m,n);
%the index of the parent row where the cheapest path came from
camefromrow=zeros(m,n);

%initialize first row
weight(1,:)=matx(1,:);
 %initialize first column
weight(:,1)=matx(:,1);

for i=1:m-1 %index over rows
    for j=1:n-1 %index over columns
        %difference between i and j must be smaller than ...
            warpwin
        if abs(i-j)≤warpwin
            stoprowjump=min([m, i+queryskip]);
            %second index over rows
            for rowjump=i+1:stoprowjump
                stopk=min([n, j+targetskip]);
                %second index over columns
                for k=j+1:stopk
                    %we favor shorter jumps by multiplying ...
                        by jummpcost
                    newweight = ( weight(i,j) +  ...
                        matx(rowjump,k) + ...
                        ((rowjump-i-1)+(k-j-1))*jumpcost) ;
                    if weight(rowjump,k) > newweight
                        weight(rowjump,k) = newweight;
                        camefromrow(rowjump,k)=i;
                        camefromcol(rowjump,k)=j;
                    end
                end
            end
        end
    end
end

% collecting the indices of points on the cheapest path
pathcost=weight(m,n);   % pathcost: minimal value
mincol=n;
minrow=m;
indxcol=[];
indxrow=[];
while (minrow>1 && mincol>1)
    indxcol=[ mincol indxcol];
    indxrow=[ minrow indxrow];
    mincoltemp=camefromcol(minrow,mincol);
    minrow=camefromrow(minrow,mincol);
    mincol=mincoltemp;
end
indxcol = indxcol(1:end-1);
indxrow = indxrow(1:end-1);
indxcol = indxcol-1;
indxrow = indxrow-1;
return
```

## A.3 Hand Pose Estimation

A comparison with each previously saved hand posed is done using OSB and DTW algorithms and the most similar hand pose is selected as the estimated hand pose. These data are then saved in DTW format for further analysis. The listing A.3 shows the MATLAB code used for this analysis. In this code it is assumed that we have 21 saved hand poses that we use as the comparison source. These data are collected and saved using the online application.

```
%Reading the previously saved reference hand poses from ...
    files for comparison
ref0 = csvread('tests/0/0 (1)/skeleton.csv');
ref1 = csvread('tests/1/1 (1)/skeleton.csv');
ref2 = csvread('tests/2/2 (1)/skeleton.csv');
ref3 = csvread('tests/3/3 (1)/skeleton.csv');
ref4 = csvread('tests/4/4 (1)/skeleton.csv');
ref5 = csvread('tests/5/5 (1)/skeleton.csv');
ref6 = csvread('tests/6/6 (1)/skeleton.csv');
ref7 = csvread('tests/7/7 (1)/skeleton.csv');
ref8 = csvread('tests/8/8 (1)/skeleton.csv');
ref9 = csvread('tests/9/9 (1)/skeleton.csv');
ref10 = csvread('tests/10/10 (1)/skeleton.csv');
ref11 = csvread('tests/11/11 (1)/skeleton.csv');
ref12 = csvread('tests/12/12 (1)/skeleton.csv');
ref13 = csvread('tests/13/13 (1)/skeleton.csv');
ref14 = csvread('tests/14/14 (1)/skeleton.csv');
ref15 = csvread('tests/15/15 (1)/skeleton.csv');
ref16 = csvread('tests/16/16 (1)/skeleton.csv');
ref17 = csvread('tests/17/17 (1)/skeleton.csv');
ref18 = csvread('tests/18/18 (1)/skeleton.csv');
ref19 = csvread('tests/19/19 (1)/skeleton.csv');
ref20 = csvread('tests/20/20 (1)/skeleton.csv');

ref = [ref0; ref1; ref2; ref3; ref4; ref5; ref6; ref7; ...
    ref8; ref9; ref10; ref11; ref12; ref13; ref14; ref15; ...
    ref16; ref17; ref18; ref19; ref20];

L = [length(ref0); length(ref1); length(ref2); ...
    length(ref3); length(ref4);
length(ref5); length(ref6); length(ref7); length(ref8); ...
    length(ref9); length(ref10); length(ref11); ...
    length(ref12); length(ref13);
length(ref14); length(ref15); length(ref16); length(ref17); ...
    length(ref18); length(ref19); length(ref20)];

% We have saved 100 samples of 21 hand pose
TestPoseNo = 100;
PoseNo = 21;

m = zeros(TestPoseNo*PoseNo,10);

dtw_result = zeros(TestPoseNo*PoseNo,24);
osb_result = zeros(TestPoseNo*PoseNo,24);
osb_d_result = zeros(TestPoseNo*PoseNo,24);
```

```
osb_path_jump= zeros(TestPoseNo*PoseNo,3);


for j=0:PoseNo-1

    for i=1:TestPoseNo

    try
        % Reading the test hand pose for pose estimation
        sub = ...
            csvread(strcat('tests/',num2str(j),'/',num2str(j),' ...
            (',num2str(i),')/skeleton.csv'));
        % Reading the number of fingertips
        fingertips = ...
            csvread(strcat('tests/',num2str(j),'/',num2str(j),' ...
            (',num2str(i),')/fingertip.csv'));
        Dist = zeros(PoseNo,1);
        pathcost = zeros(PoseNo,1);
        jumpcost = zeros(PoseNo,1);
        d = zeros(PoseNo,1);

        temp = 1;
        for k = 1:PoseNo
            [Dist(k),¬,¬,¬] = dtw(sub, ...
                ref(temp:temp+L(k)-1,:));
            [pathcost(k),indxrow,¬,jumpcost(k), d(k)] = ...
                OSBv5(sub,ref(temp:temp+L(k)-1,:),0,100,100,-1,1);
            temp = temp+L(k);
        end

        [min_DTW_dist, gest_dtw_index] = min(Dist);
        [min_OSB_dist, gest_osb_index] = min(pathcost);
        [OSBpath_cost, OSBpath_cost_index] = ...
            min(pathcost.*jumpcost);
        [min_OSB_d, gest_osb_d_index] = min(d);

       m(TestPoseNo*j+i,:) = [j fingertips(1) min_DTW_dist ...
           gest_dtw_index-1 min_OSB_dist gest_osb_index-1 ...
           min_OSB_d  gest_osb_d_index-1 OSBpath_cost ...
           OSBpath_cost_index-1];
      dtw_result(TestPoseNo*j+i,:) = [j gest_dtw_index-1 ...
          fingertips(1) Dist(1) Dist(2) Dist(3) Dist(4) ...
          Dist(5) Dist(6) Dist(7) Dist(8) Dist(9) Dist(10) ...
          Dist(11) Dist(12) Dist(13) Dist(14) Dist(15) ...
          Dist(16) Dist(17) Dist(18) Dist(19) Dist(20) ...
          Dist(21)];
       osb_result(TestPoseNo*j+i,:) = [j gest_osb_index-1 ...
           fingertips(1) pathcost(1) pathcost(2) ...
           pathcost(3) pathcost(4) pathcost(5) pathcost(6) ...
```

```
              pathcost(7) pathcost(8) pathcost(9) pathcost(10) ...
              pathcost(11) pathcost(12) pathcost(13) ...
              pathcost(14) pathcost(15) pathcost(16) ...
              pathcost(17) pathcost(18) pathcost(19) ...
              pathcost(20) pathcost(21)];
          osb_d_result(TestPoseNo*j+i,:) = [j ...
              gest_osb_d_index-1 fingertips(1) d(1) d(2) d(3) ...
              d(4) d(5) d(6) d(7) d(8) d(9) d(10) d(11) d(12) ...
              d(13) d(14) d(15) d(16) d(17) d(18) d(19) d(20) ...
              d(21)];
         osb_path_jump(TestPoseNo*j+i,:) = [j ...
             OSBpath_cost_index-1 gest_osb_index-1];
        catch
            display('error:');
            display(num2str(j));
            display(num2str(i));
        end
     end
 end
 csvwrite('data/poseEstimationAll.csv',m);
 csvwrite('data/poseEstimationDTW.csv',dtw_result);
 csvwrite('data/poseEstimationOSB.csv',osb_result);
 csvwrite('data/poseEstimationOSBPathJump.csv',osb_path_jump);
 csvwrite('data/poseEstimationOSBd.csv',osb_d_result);
```

## A.4 Hand Pose Comparison Using a Single Reference Pose

Another test have been done in which the poses data are compared to a single reference hand pose (in our experiments an open hand). The comparison is done using both OSB and DTW algorithms. And some other features such as number of fingertips, width and height of the hand pose are also added for better differentiating the poses.

```
ref = csvread('tests/5/5 (1)/skeleton.csv');
TestPoseNo = 100;
PoseNo = 21;
m = zeros(TestPoseNo*PoseNo,7);
for i=1:TestPoseNo*PoseNo
    try
         sub = csvread(strcat('tests/1 ...
             (',num2str(i),')/skeleton.csv'));
         fingertips = csvread(strcat('tests/1 ...
             (',num2str(i),')/fingertip.csv'));
         [Dist,¬,k,¬] = dtw(sub,ref);
```

```
            [pathcost,indxrow,indxcol,jumpcost, d] = ...
                OSBv5(sub,ref,0,100,100,−1,1);
            w = max(sub(:,1)) − min(sub(:,1));
            l = max(sub(:,2)) − min(sub(:,2));
            m(100*j+i,:) = [j fingertips(1) Dist k pathcost ...
                length(indxrow) jumpcost d w l];
        catch
            %do nothing
        end
end
csvwrite('data/singleSourceComparison.csv',m);
```

## A.5 Visualization of Hand Pose Comparison Using a Single Reference Pose

The data that was provided by the procedure in the previous section can be visualized to have a more clear idea about how separable they are and what features best discriminate the different hand poses from each other. This code is specifically written for the set of hand poses that are defined for testing in this thesis and should be modified to be able to use it with a different set of poses.

```
raw = csvread('data/singleSourceComparison.csv');
%scaling normalization
for i=3:size(raw,2)
    raw(:,i) = (raw(:,i) − min(raw(:,i)))/max(raw(:,i));
end

%separate gestures
g0 = raw(raw(:,1)==0,:);
g1 = raw(raw(:,1)==1,:);
g2 = raw(raw(:,1)==2,:);
g3 = raw(raw(:,1)==3,:);
g4 = raw(raw(:,1)==4,:);
g5 = raw(raw(:,1)==5,:);
g6 = raw(raw(:,1)==6,:);
g7 = raw(raw(:,1)==7,:);
g8 = raw(raw(:,1)==8,:);
g9 = raw(raw(:,1)==9,:);
g10 = raw(raw(:,1)==10,:);
g11 = raw(raw(:,1)==11,:);
g12 = raw(raw(:,1)==12,:);
g13 = raw(raw(:,1)==13,:);
g14 = raw(raw(:,1)==14,:);
```

```
g15 = raw(raw(:,1)==15,:);
g16 = raw(raw(:,1)==16,:);
g17 = raw(raw(:,1)==17,:);
g18 = raw(raw(:,1)==18,:);
g19 = raw(raw(:,1)==19,:);
g20 = raw(raw(:,1)==20,:);

gs = [g0; g1; g2; g3; g4; g5; g6; g7; g8; g9; g10; g11; ...
    g12; g13; g14; g15; g16; g17; g18; g19; g20];
codes =   [0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 ...
    19 20]';
%gssize = [size(g0,1) size(g1,1) size(g2,1) size(g3,1) ...
    size(g4,1) size(g5,1) size(g10,1) size(g11,1) ...
    size(g21,1) size(g30,1)]';
ngs = length(codes);
markers = ['d' 'x' 'o' 'v' '^' 's' 'x' 'x' 'o' 'o' 'v' 'v' ...
    'o' 's' 's' '^' '^' 'o' 'o' 'v' 'v']';
colors =  ['k' 'm' 'c' 'r' 'r' 'b' 'k' 'c' 'b' 'm' 'g' 'm' ...
    'k' 'r' 'g' 'b' 'g' 'g' 'r' 'k' 'b']';
font = 'Arial';

% different features are presented in the csv file, here ...
    the OSB path cost, the OSB jumpcost, and the number of ...
    fingers are considered
 f1 = 5; f2 = 7;

%gesture 2d for 0 fingertips

    gest = raw(raw(:,1)==codes(1),:);
    plot(gest(:,f1),gest(:,f2),markers(1)
        ,'MarkerSize',10,'MarkerFaceColor', colors(1));
    % Manually set axis limits, since TEXT does not do this ...
        for you
    xlim([min(gest(:,f1)) max(gest(:,f1))])
    ylim([min(gest(:,f2)) max(gest(:,f2))])
    hold on
    xlabel('Optimal Path Cost');
    ylabel('Jump Cost');

    xlim([0 1])
    ylim([0 1])

% gestures 2d for 1 fingertips
    gest = raw(raw(:,1)==codes(2),:)

    plot(gest(:,f1),gest(:,f2),markers(2)
        ,'MarkerSize',10,'color', colors(2));
    hold on

    gest = raw(raw(:,1)==codes(8),:);
    plot(gest(:,f1),gest(:,f2),markers(8)
        ,'MarkerSize',10,'color', colors(8));
    hold on

    gest = raw(raw(:,1)==codes(7),:);
    plot(gest(:,f1),gest(:,f2),markers(7)
        ,'MarkerSize',10,'color', colors(7));
    hold on

    xlabel('Optimal Path Cost');
    ylabel('Jump Cost');

    xlim([0 1])
    ylim([0 1])

    % gestures 2d for 2 fingertips
    gest = raw(raw(:,1)==codes(3),:)
    plot(gest(:,f1),gest(:,f2),markers(3)
        ,'MarkerSize',10,'MarkerFaceColor', colors(3));
    hold on

    gest = raw(raw(:,1)==codes(9),:);
    plot(gest(:,f1),gest(:,f2),markers(9)
        ,'MarkerSize',10,'MarkerFaceColor', colors(9));
    hold on

    gest = raw(raw(:,1)==codes(10),:);
    plot(gest(:,f1),gest(:,f2),markers(10)
        ,'MarkerSize',10,'MarkerFaceColor', colors(10));
    hold on

    gest = raw(raw(:,1)==codes(13),:);
    plot(gest(:,f1),gest(:,f2),markers(13)
        ,'MarkerSize',10,'MarkerFaceColor', colors(13));
    hold on

    gest = raw(raw(:,1)==codes(19),:);
    plot(gest(:,f1),gest(:,f2),markers(19)
        ,'MarkerSize',10,'MarkerFaceColor', colors(19));
    hold on

     gest = raw(raw(:,1)==codes(18),:);
    plot(gest(:,f1),gest(:,f2),markers(18)
        ,'MarkerSize',10,'MarkerFaceColor', colors(18));
    hold on
   xlim([0 0.15])
    ylim([0 0.55])
    xlabel('Optimal Path Cost');
    ylabel('Jump Cost');

    % gestures 2d for 3 fingertips

    gest = raw(raw(:,1)==codes(4),:)
    plot(gest(:,f1),gest(:,f2),markers(4)
        ,'MarkerSize',10,'MarkerFaceColor', colors(4));
    hold on

    gest = raw(raw(:,1)==codes(11),:)
    plot(gest(:,f1),gest(:,f2),markers(11)
        ,'MarkerSize',10,'MarkerFaceColor', colors(11));
    hold on

    gest = raw(raw(:,1)==codes(12),:);
     plot(gest(:,f1),gest(:,f2),markers(12)
        ,'MarkerSize',10,'MarkerFaceColor', colors(12));
    hold on

      gest = raw(raw(:,1)==codes(20),:);
     plot(gest(:,f1),gest(:,f2),markers(20)
        ,'MarkerSize',10,'MarkerFaceColor', colors(20));
    hold on

    gest = raw(raw(:,1)==codes(21),:);
    plot(gest(:,f1),gest(:,f2),markers(21)
        ,'MarkerSize',10,'MarkerFaceColor', colors(21));
    hold on

    xlabel('Optimal Path Cost');
    ylabel('Jump Cost');

    % gestures 2d for 4 fingertips
    gest = raw(raw(:,1)==codes(5),:);
    plot(gest(:,f1),gest(:,f2),markers(5)
        ,'MarkerSize',10,'MarkerFaceColor', colors(5));
    hold on

    gest = raw(raw(:,1)==codes(16),:);
    plot(gest(:,f1),gest(:,f2),'^'
        ,'MarkerSize',10,'MarkerFaceColor', colors(16));
    hold on

    gest = raw(raw(:,1)==codes(17),:);
    plot(gest(:,f1),gest(:,f2),'^'
        ,'MarkerSize',10,'MarkerFaceColor', colors(17));
    hold on

    xlabel('Optimal Path Cost');
    ylabel('Jump Cost');

     % gestures 2d for 5 fingertips

    gest = raw(raw(:,1)==codes(6),:)
    plot(gest(:,f1),gest(:,f2),'s'
        ,'MarkerSize',10,'MarkerFaceColor', colors(6));
    hold on

     gest = raw(raw(:,1)==codes(14),:)
    plot(gest(:,f1),gest(:,f2),'s'
        ,'MarkerSize',10,'MarkerFaceColor', colors(14));
    hold on

     gest = raw(raw(:,1)==codes(15),:)
    plot(gest(:,f1),gest(:,f2),'s'
        ,'MarkerSize',10,'MarkerFaceColor', colors(15));
    hold on

    xlabel('Optimal Path Cost');
    ylabel('Jump Cost');

      % gestures 2d for all
    for i=1:21
        gest = raw(raw(:,1)==codes(i),:)
        plot(gest(:,f1),gest(:,f2),markers(i)
            ,'MarkerSize',10,'MarkerFaceColor', colors(i));
        hold on
    end

     ylabel('Number of fingers');
    xlabel('Optimal Path Cost');

    %gesture 3d
for i=1:21
    %ind = find(raw(:,1)==codes(i),1);
    gest = raw(raw(:,1)==codes(i),:);

    plot3(gest(:,f1),gest(:,f2),gest(:,f3),markers(i),
        'MarkerSize',10,'MarkerFaceColor', colors(i));
    grid on;
    hold on;
    %%%%%%%%%%%
    gest_red = gest(:,[f1 f2 f3]);
    med = median(gest_red);
    gest_std = std(gest_red);
    xlabel('Optimal Path Cost');
    zlabel('Number of fingers');
    ylabel('Jump Cost');
    %%%%%%%%%%%
end
```

## A.6 HMM Train and Recognition

Listing A.6 shows a sample code for training and testing the HMM models for 5 gestures using the Hidden Markov Model Toolbox for MATLAB [3].

```
%%%%%%%%% MAIN PARAMS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

O = 21;  %number of symbols
Q = 3;  %number of states
T = 10;  %gesture length

k = 5;  %number of hmms (number of gestures)

addpath(genpath('C:\Users\Mahsa\osb2\HMMall\HMMall'));

%initial hmm parameters
prior0 = normalise(rand(Q,1));
transmat0 = mk_stochastic(rand(Q,Q));
obsmat0 = mk_stochastic(rand(Q,O));

%%%%%%%%% MAIN PARAMS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%% BUILD HMMs %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
hmm = struct('LL',zeros(1,1),'prior',zeros(Q,1)
,'transmat',zeros(Q,Q),'obsmat',zeros(Q,O));
load 'opclfin.mat';
load 'rot89.mat';
load 'clop.mat';
load 'open5tocl.mat';
load 'rot5right.mat';

for i=1:k

     if (i==1)
         data = opclfin;
     end
     if (i==2)
         data = rot89;
     end
     if (i==3)
         data = rot5right;
     end
     if (i==4)
```

```
            data = clop;
        end
        if (i==5)
            data = open5tocl;
        end

        %train the ith hmm with the acquisted data
        [LL, prior, transmat, obsmat] = dhmm_em(data, prior0, ...
           transmat0, obsmat0, 'max_iter', 12);
        hmm(i).LL = LL(end);
        hmm(i).prior = prior;
        hmm(i).transmat = transmat;
        hmm(i).obsmat = obsmat;
    end

%%%%%%%%% BUILD HMMs %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%% RECOGNITION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
TEST_NO = 50;
Symb_no = 10;
test_data1 = zeros(Symb_no, TEST_NO);
test_data2 = zeros(Symb_no, TEST_NO);
test_data3 = zeros(Symb_no, TEST_NO);
test_data4 = zeros(Symb_no, TEST_NO);
test_data5 = zeros(Symb_no, TEST_NO);
for i = 16:65
   temp1 = ...
       csvread(strcat('2013-8-28/test-open-close-finger/1 ...
       (',num2str(i),')/gest.csv'));
   temp2 = csvread(strcat('2013-8-28/test-rotate-8-to-9/1 ...
       (',num2str(i),')/gest.csv'));
   temp3 = csvread(strcat('2013-8-28/test-5-to-right/1 ...
       (',num2str(i),')/gest.csv'));
   temp4 = csvread(strcat('2013-8-28/test-open-close/1 ...
       (',num2str(i),')/gest.csv'));
   temp5 = ...
       csvread(strcat('2013-8-28/test-5-to-close-fingers/1 ...
       (',num2str(i),')/gest.csv'));

   temp1(1) = temp1(2);
   temp2(1) = temp2(2);
   temp3(1) = temp3(2);
   temp4(1) = temp4(2);
   temp5(1) = temp5(2);

   test_data1(:,i-16+1) = [temp1; ...
       temp1(end)*ones(10-length(temp1),1)];
   test_data2(:,i-16+1) = [temp2; ...
       temp2(end)*ones(10-length(temp2),1)];
    test_data3(:,i-16+1) = [temp3; ...
        temp3(end)*ones(10-length(temp3),1)];
    test_data4(:,i-16+1) = [temp4; ...
        temp4(end)*ones(10-length(temp4),1)];
    test_data5(:,i-16+1) = [temp5; ...
        temp5(end)*ones(10-length(temp5),1)];
end
test_data1 = test_data1' +1;
test_data2 = test_data2' +1;
test_data3 = test_data3' +1;
test_data4 = test_data4' +1;
test_data5 = test_data5' +1;

loglik1 = zeros(k,TEST_NO);
I1 =zeros(TEST_NO,1);

loglik2 = zeros(k,TEST_NO);
I2 =zeros(TEST_NO,1);

loglik3 = zeros(k,TEST_NO);
I3 =zeros(TEST_NO,1);

loglik4 = zeros(k,TEST_NO);
I4 =zeros(TEST_NO,1);

loglik5 = zeros(k,TEST_NO);
I5 =zeros(TEST_NO,1);

for obs = 1:TEST_NO

    for i=1:k
        loglik1(i,obs) = dhmm_logprob(test_data1(obs,:), ...
            hmm(i).prior, hmm(i).transmat, hmm(i).obsmat);
        loglik2(i,obs) = dhmm_logprob(test_data2(obs,:), ...
            hmm(i).prior, hmm(i).transmat, hmm(i).obsmat);
        loglik3(i,obs) = dhmm_logprob(test_data3(obs,:), ...
            hmm(i).prior, hmm(i).transmat, hmm(i).obsmat);
        loglik4(i,obs) = dhmm_logprob(test_data4(obs,:), ...
            hmm(i).prior, hmm(i).transmat, hmm(i).obsmat);
        loglik5(i,obs) = dhmm_logprob(test_data5(obs,:), ...
            hmm(i).prior, hmm(i).transmat, hmm(i).obsmat);
    end

    [¬,I1(obs)] = max(loglik1(:,obs));
    [¬,I2(obs)] = max(loglik2(:,obs));
    [¬,I3(obs)] = max(loglik3(:,obs));
    [¬,I4(obs)] = max(loglik4(:,obs));
    [¬,I5(obs)] = max(loglik5(:,obs));
end
```

```

save 'test_data1';
save 'test_data2';
save 'test_data3';
save 'test_data4';
save 'test_data5';

save 'I1';
save 'I2';
save 'I3';
save 'I4';
save 'I5';

save 'loglik1';
save 'loglik2';
save 'loglik3';
save 'loglik4';
save 'loglik5';
%choose the most probabale gesture

%%%%%%%%% RECOGNITION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```