The University Of Sheffield.

Department of Electronic & Electrical Engineering

MSc Computer Vision Engineering

Dissertation 2012

Hand Gesture-Based Computer User Interface Using Kinect

Student: **ASAD,** Muhammad

Registration: **110128953**

Date: **7th September 2012**

Supervisor: **Dr. Charith Abhayaratne**

Second Marker: **Dr. Luke Seed**

This dissertation is a part requirement for the degree of MSc in Computer Vision Eng.

Abstract

This project designs and implements a contact less hand gesture based human-computer interface using Microsoft Kinect. This interface recognizes hand gestures which are widely used in everyday interactions with real objects, but are complex for computers to understand, which eventually makes the interface more natural and easy to use. This is achieved by utilizing the depth data from Microsoft Kinect to capture parameters like hand shape, position, orientation and motion.

The design approach used in this project uses hand tracker from OpenNI SDK to track the hand using depth map. The output of this tracker is used to perform an intelligent segmentation of hand region. After extraction, further processing is done using OpenCV libraries to extract three different projections of hand in xy, zx and zy plane. An efficient feature extraction technique is applied to extract these contour features in real-time. These features are then used to construct a neural network model. The output of neural network is used for hidden markov model based gesture recognition. The performance of the proposed approach was real-time execution at 25 frames per second with an average accuracy of 94% for the neural network output.

We have used two dynamic hand gestures, namely swipe right and swipe left. A series of hand postures are used to define these gesture. The output of the recognition process is used to map simple functions like browsing pictures in a picture viewer or moving between different slides in a presentation software. The idea is to map each gesture to a corresponding function in Windows API. This project can be potentially used in any application which involves navigation between left and right side.

Though human genius in its various inventions with various instruments may answer the same end, it will never find an invention more beautiful or more simple or direct than nature, because in her inventions nothing is lacking and nothing is superfluous

Leonardo da Vinci

Contents

List of Figures

List of Tables

Acknowledgments

This thesis work would not have been possible without the support of many people.

I am truly thankful to my supervisor Dr. Charith Abhayaratne for his continuous support and encouragement throughout this project. I am sincerely thankful to my second supervisor Dr. Luke Seed for his technical advice and guidance. I am also thankful to Dr. Dabo Guo for helping and having important discussions in the lab.

I would also like to take this opportunity to thank Dr. Ling Shao and all other professors who taught me during my masters degree.

I owe a special thanks to my mother for her encouragement, support, prayers, patience and endless love. I am thankful to my family especially my sisters for their support and encouragement.

I am heartily thankful to my friends, who filled this MSc with fun and everlasting friendships. I owe a special thanks to my friend Rush for being a great friend.

Finally, I owe a sincere and earnest thanks to my flatmates for making this year an unforgettable experience for me.

Chapter 1

Introduction

With the introduction of Microsoft Kinect in the recent years, there has been a lot of research to use this inexpensive and powerful sensor in almost every domain in the field of Computer Vision. This sensor provides depth data in real-time, adding another dimension to an already well developed field. Combining this with the functionality of different Computer Vision libraries, a whole new category of applications are emerging. Researchers are utilizing the depth map, to both improve the efficiency and robustness of the existing approaches and propose new approaches which were impossible to realize before.

The recent advancements in technology have resulted in the introduction of more and more natural human-computer interaction methods. An example of such methods is the introduction of user friendly multi-touch interfaces in mobile phones, which have revolutionized the whole mobile phone industry and changed the way people interact with them. With these multi-touch screens, users are now able to interact using gestures which are natural and commonly used. On the contrary, there have not been significant improvements in the human-computer interaction for PC. We are still limited by the use of mechanical devices such as mouse and keyboard in order to do our task. Though these devices have become a part of our daily life, they may be the ones limiting the potential work that can be done on computers [1].

A lot of research today is focusing on hand gestures based interfaces. This is due to the fact that these are the most basic, natural and comprehensively defined gestures. Another reason for such choice is the ease of use, as the user does not need prior knowledge, training or memorizing different commands. These are the gestures we use everyday in our interaction with physical objects. Hand gestures can be organized into two categories, based on parameters used for recognition. These categories are depicted below.

1. **Static Hand Gestures:** These type of hand gestures are defined by the hand shape,

position and orientation only. There is no motion or displacement involved. These gestures usually include symbolic gestures, which can be mapped on to some predefined interpretations. They are not natural, and are also difficult for users to learn and to remember. An example of symbolic interpretation of gestures is the use of hand gestures to interpret symbols in sign language.

2. **Dynamic Hand Gestures:** These gestures are temporal integration of static gestures. They include gesture which are more complex and difficult to recognize on computers, as they are defined by a number of different parameters. These parameters are hand position, orientation, motion, acceleration and finger displacement. On the contrary, dynamic hand gestures are already used extensively in our everyday interaction with different physical objects. These gestures have the true potential to make computer interfaces more natural and user friendly as they can assist more and more people to use computers without any prior training.

   Dynamic gestures can be further interpreted as direct interpretation or mapped gestures. For direct interpretation, the movements of the hand are directly utilized. These include applications related to telepresence robotics, where the user can control the movements of the robot by using the movements of his hand. Mapped gestures, on the other hand, use the recognized gesture to trigger a specific function. This function then performs a specific predefined procedure. An example of such gestures is a sliding switch used in mobile phones.

It was observed from the literature review in Chapter 2 that for most of the proposed approaches, the choice of hand gestures used was more focused on improving the performance and reliability of the gesture recognition algorithm. Most of the gestures chosen were easier for computers to recognize, however they were difficult for humans to learn and use. This resulted in deviation of the proposed approaches from the original purpose i.e. to make human-computer interaction more natural and easy to use [2]. This may be the reason why none of these approaches were widely accepted and used across different platforms. Another motivation for this project is the introduction of real-time depth sensors like Microsoft Kinect. The depth information can be used to recognize gestures more accurately, giving a chance to include more natural and user-friendly gestures.

This project proposes a contact less user interface which can recognize dynamic hand gestures and map them to simple functions in an application. The gestures used are natural and common, making the interface more user-friendly.

This report is divided into subsequent chapters. Chapter 2 provides a background study and then presents related work. In Chapter 3 the proposed approach is introduced along

with challenges faced and their proposed solutions. Chapter 4 puts forward the results and comparisons. This report ends with a conclusion and discussion of the future work in Chapter 5.

Chapter 2

Background Study and Literature Review

2.1 Microsoft Kinect Sensor

2.1.1 What is Kinect?

Microsoft Kinect is a depth camera which gives accurate depth values of the scene in view. These depth values are the actual distances of different objects in view from the sensor. The operating range of Kinect sensor is 500mm to 8000mm, which is quantized and stored in an 11-bit grayscale image, called a depth map. This depth map is different in many ways to normal intensity images. It does not gives any detail about the texture, colour or illumination of the scene. This makes the depth maps hard for humans to understand. On the contrary, the information stored in these depth maps makes them easier for computers to segment each object based on their shape, size and distance from the camera. In addition to this depth camera, Kinect sensor also has an On-board RGB camera. This camera is placed at a known distance from depth sensor, which enables the sensor to combine both a depth map and an RGB image to provide a complete view of the scene.

2.1.2 How does it work?

To understand the background and scope of this project, it is essential to first understand the mechanism behind the Kinect depth camera. Figure 2.1 shows a frame captured using Kinect's infrared camera. There are a few points to notice here. First of all, there is a pattern of many infra-red points projected onto the field of view. Secondly, this figure shows that these points

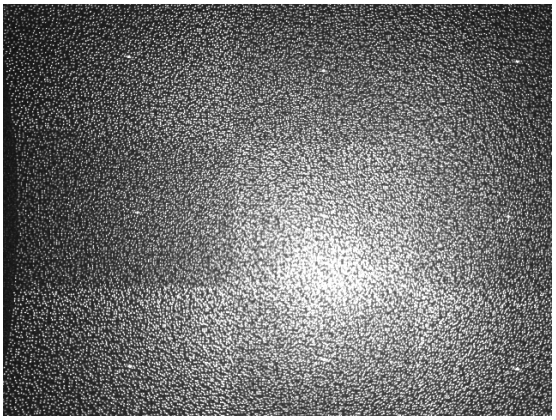
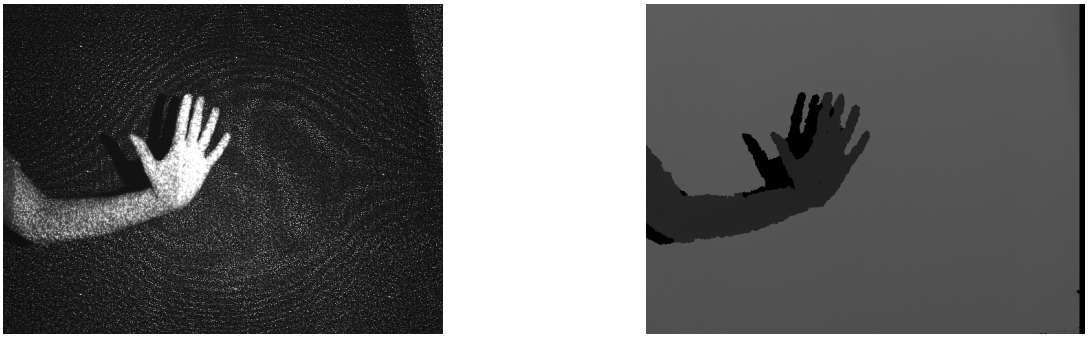


Figure 2.1: View from Kinect's Infrared camera

are spread over a specific distance. Lastly, notice that the pattern is divided into 9 different square regions, with each region's center slightly brighter than the rest of the projected points. While it is not clear as to why there are 9 different patterns, one possible reason could be to reduce the complexity of the on-board depth calculation. Another reason could be to align different regions for disparity map calculation. This simple projection is itself used to calculate the depth of the scene. The distance between these infra-red points depends directly on the distance of the object from the camera. For the calculation of depth map, the deviation in this sequence is correlated to a known distance sequence of infra-red points. This concept becomes clear when an object is moved into the scene. In Fig.2.2(a), a hand is placed in front of the sensor. It can be noticed that the pattern is different on the hand as it is closer to the depth camera. This corresponds to a depth map with hand region detected as closer to the sensor than the background (Fig. 2.2(b)). The purpose of this study is to understand and learn the limitations of Kinect sensor related to hand gesture recognition, more in-depth details about Kinect sensor can be found in [3].

2.1.3 Why Kinect?

There are a number of reasons for choosing Kinect Sensor for this research project. The most significant reason for using this sensor is that it provides a depth map of 640x480 pixel resolution at 30 fps [4]. This real-time depth map also makes it easier to segment different objects based on their location in space. It also provides the 3D shape information, which is an essential feature used in this project to achieve accurate real-time execution. The mechanism used for construction of this depth map makes use of infra-red projection, which in turns makes this depth map invariant to illumination/lighting changes.



(a) Infrared view (b) Depth map

Figure 2.2: Comparison between actual projected points and constructed depth map for a normal hand view

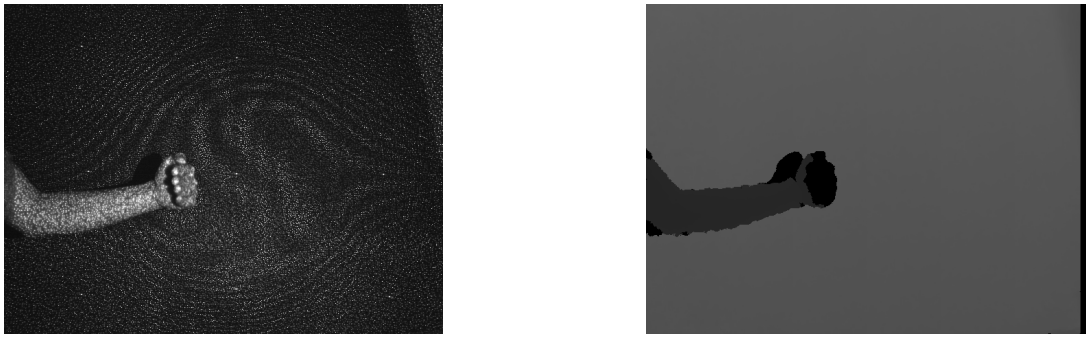
On the contrary, it is difficult to segment different objects using ordinary intensity camera. These techniques also face a lot of problems due to illumination changes and background cluttering. A more detailed discussion about camera based methods is presented in the section 2.2.

2.1.4 Does Kinect have any limitations?

Most of the related research today has focused on listing the advantages Kinect has over other devices. However, none of the papers in the literature have focused on listing its limitations specific to their research area. This section explains these limitations and their possible effects on the implementation of this project. Most of the research in this project is focused on providing a solution to these limitations, however there are certain limitations which can not be overcome for the time being. These methods are discussed in detail in chapter 3.

Analysing the output from infra-red camera in Fig. 2.1 and 2.2, the first limitation is observed in the background. Figure 2.2(a) shows a shadow in the infra-red points in the background. This is the region where these projected points can not reach as a result of occlusion caused by hand. As the depth calculation method is based on the correlation of the points which are reflected back, therefore the depth in these region can not be determined (Fig. 2.2(b)). This also makes it impossible to calculate depth for objects which do not reflect enough light back. It also decreases the accuracy of calculated depth with increased distance from the sensor as the reflection from a distant object contains errors. In [3] the author compared these errors related to distance of the objects from the sensor. They found out that these random errors increase to a maximum value of 4cm at 5 meters distance from the Kinect.

Kinect sensor was originally developed for recognition of whole body actions using five



(a) Infrared view (b) Depth map

Figure 2.3: Comparison between actual projected points and constructed depth map for a hand pointing directly towards sensor

extremities of human body [4], which did not require a lot of accuracy. In the Kinect design a balance was kept between on-board processing capabilities and the amount of data being processed (number of projected infra-red points). This optimization of Kinect for whole body gestures is the reason why detecting a hand from a distance is a challenging problem.

Lastly, looking at figure 2.3(a), it can be observed that the hand regions which are at an almost perpendicular angle to the field of view, reflect less points back. These points are not enough to capture and triangulate the depth data. As a result of this, the corresponding depth map in figure 2.3(b) has a discontinuity around the pointed region of hand.

2.2 Background Study

There has been a lot of previous work done in developing a human-computer interface based on hand gestures. All the proposed approaches can be divided into three categories based on the method used for interpreting human gestures to measurable parameters for computers. These categories are depicted in the subsections below along with the limitations involved in these approaches.

2.2.1 Dataglove/Sensor based methods

These were one of the first approaches used for building a hand gesture based user interface [5], [6], [7], [8]. They relied on the direct measurement of joints position and angle using devices with sensors such as dataglove. These devices were bulky, with a lot of cables connecting them with computers. Although they were used in many specific application domains, they still lacked the freedom and natural way of interaction [1]. These were reliable and robust methods,

which could be used for hand gesture recognition and mapping in some specific fields. However their lack of naturalness limits them to replace conventional human-computer interfaces used today.

2.2.2 Vision based methods

With the advancements in technology, both optical cameras and computers became more powerful and affordable. This led to a significant amount of research in the field of Computer Vision, making it possible to recognize hand gestures in real-time. These types of method relied on colour, shape and motion information and involved building of a contact less hand gesture based interface [9], [10]. Although the gestures that can be recognized are much more natural now, there are a lot of constraints affecting the robustness and reliability of such methods [1]. These limitations are discussed in detail in section 2.3, along with the proposed solutions in literature.

2.2.3 Vision and depth based methods

Over the recent years, there has been significant improvements in the technology used in the imaging sensors. With the introduction of inexpensive and real-time depth sensors like Microsoft Kinect, it is now possible to use actual 3D information for building a hand gesture based human-computer interface.

There are a lot of depth sensor based methods emerging, but most of these methods do not fully utilize the potential provided by the depth information. Some authors used the assumption that hand region is always closest to the camera. They segmented the nearest depth from the Microsoft Kinect to successfully get the hand region [11]. These approaches applied a threshold on the depth to get hand region in 2D, which was then used for processing. In [12], author assumes that the user is standing 3 meters from camera. They also only used depth data to segment and track the hand region with increased reliability. Some approaches use only the 2D shape information from depth sensor for recognizing static gestures [13]. These approaches are still limited to 2D recognition methods proposed earlier. There has been no significant progress, even with the introduction of real-time depth sensors. This category is related to the scope of this project, therefore these approaches are discussed in detail in section 2.4.

2.3 Challenges faced and proposed solutions

This section discusses the challenges faced by previous research. It also details the solution to some of the challenging problems as proposed in previous research. These are discussed in subsequent sections below.

2.3.1 Hand Tracking and Segmentation

A major problem faced in previous work was the localization and segmentation of the hand region. Early methods used different coloured or reflective markers to track the movement of fingers [14]. They were limited by their ease of use. These approaches failed when an object with similar colour as the marker appeared in the background. Later, more advanced methods were introduced, which used skin tone, shape or motion of the hand for localizing and tracking fingers. Performance of methods relying on skin tone-based hand segmentation was significantly affected by the changes in the lighting conditions and the surrounding environment (background cluttering) [15], [16], [17], [18], [19], [20], [21], [22], [23]. Furthermore, it is difficult to construct a skin tone model appropriate for all skin tones [10]. In [24] and [25], a hand gesture recognition method is proposed which uses skin colour from face to track the hand movements. This method takes the assumption that there will always be a face present in the frames. It fails to track hand correctly if there is a skin coloured object in the background or if the hand is inside the face region in the image. Approaches relying on shape, [26] and [27], were evaluated in [28]. The performance of these methods is not satisfactory, as the shape of the hand changes significantly, making it difficult to detect using only RGB data. Methods relying on correlation of shape were slow, and rigid in the sense that they did not cater for changes in orientation and pose of the hand [29], [30]. On the contrary, methods using motion involved segmentation of the hand based on background foreground models [31], [32], [33], [34]. These methods were more invariant to background cluttering and motion blur arising as a result of fast hand motion. However the only limitation faced by these methods was that they were based on the assumption that the only moving object in the sequence was hand. In real world scenarios, this is not always true. Much previous research has focused on a limited number of postures of hand, which limits the capability of these systems. Furthermore, some methods even limit the orientation of hand, to eliminate the effects of rotation.

Most of the previously proposed methods lack the use of a proper hand model with some exceptions [35], [36], [37]. This restricts these methods to only the information which was visible, making it difficult to track hand in situations where self-occlusion occurred. For these methods, most of the processing power was used in reacquiring track of lost fingers. A model based 3D hand tracker is proposed in [38], which can reliably track hand pose and has been tested with different scenarios containing parts of hand occluded by an object of known shape. This approach also addresses the problem of noise with distance and works well with small resolution of the hand. The only limitation with this approach is that it uses a high end pc with GPU processor to only achieve 15 frames per second execution. This makes this approach not suitable for real-time gesture recognition. Another robust approach was presented in [39] which provided a design framework for combining different types of data, including 3D data,

skin colour, contour and edge information, to track hand. This approach also utilized a hand model for prediction and was implemented on a 400 Mhz celeron processor, producing an execution of 3 frames per second. This method had a reliable output, but it is not suitable for our implementation as it has a high execution time.

With the exception of few approaches [32], [17], [37], the background was always taken to be in contrast with the skin tone, making the solution impractical. The problem of hand over face was not properly addressed by most of the methods. Motion blur was a major challenge in most methods as it affected the output of segmentation process. This problem was addressed in [34] where they used a re-sampling technique to eliminate the effects of motion blur.

Another limitation in most of these methods is related to the availability of only 2D intensity information. This makes it very difficult to model hand gestures in their real 3D form, limiting these methods to use simple static hand gestures. These gestures are easy for computers to understand, but they are not natural for users to use in their daily life [40] [24] [41].

2.3.2 Feature Extraction

Okada [42] proposed a method for matching a 3D model using silhouette images of different projections of the model. This method uses similarity measure for matching each shape, which is not reliable. One of the limitations of this approach is that it is not invariant to noise/distortion. A similar but more robust approach was proposed for 3D action recognition [43]. This approach is also invariant to some variations in style and view point. It utilizes Guassian Mixture Models for modelling and recognizing different actions. This also makes this method tolerant to noise to some extent [44].

Deng et al. [9] proposed a method to recognize static hand using shape context based matching. This approach uses shape context for matching, which is invariant to rotation and deformation in shape and requires good alignment of the matching image with the reference [45]. To cater for different viewpoints, a lot of training data is required. In our approach we use a combination of different hand postures to model a dynamic gesture. These postures might change from person to person, depending on the viewpoint, inter-person style and speed differences. Due to this problem, it becomes difficult to use shape context to match each posture efficiently. Another problem in this method is that it requires a lot of processing power. With an execution time of 60 frames per minute [9], it is not suitable for developing a robust and efficient dynamic gesture recognition system.

A significant amount of research has been done to use Hu moments for shape recognition [46]. Hu moments are invariant to scale, rotation and translation [47]. In [48], the author proposed a method to use Hu moments for gait recognition. This method introduced an error region in the form of a circular mask. The size of this mask was increased and at each interval first,

second and eighth Hu moments were recorded. These moments were later used to recognize a specific pose in gait. In [47], the author compared the performance of seven Hu moments with different scales and rotation changes. The findings of this research were that a lot of errors were introduced in the Hu moments of images with resolutions lower than 150 pixels. Using Kinect Sensor, the extracted hand region in the operating region has a resolution ranging from 150x150 pixels to 64x64 pixels. Additionally, there is a lot of noise introduced at greater distances, which can affect the Hu moments. Therefore Hu moments were not a suitable option for our implementation. A comparison of Hu moments with our proposed approach is also presented in chapter 4.

2.3.3 Gesture Recognition

For gesture recognition part, there were a variety of methods used mainly based on the shape of the hand. In [40], the author uses viola-jones method using Haar-like features and integral images for detection. Features are extracted using Hu invariant moments. These feature vectors are then used to train an svm for final recognition. After training, recognition is done using the same method except the svm is now used for classification. This method produces an accuracy of over 90% using simple static symbolic gestures. Although the accuracy of this method is good, but the approach is limited to static symbolic gestures only.

In [24], the author is able to recognize both static and dynamic gestures. The recognition of static gestures is done using Haar-like features for each gestures. Dynamic gestures are detected using motion history image approach. The system provides an accuracy of 94.1% with an execution time of 3.81 ms, making it reliable and robust approach. The gestures used in this approach are only limited to their motion in 2D and the hand tracking algorithm is based on skin colour from face. It assumes that the hand is always next to the face. If it gets in front of the face region, it does not work. It also assumes that there will not be any skin coloured objects moving in the background, which is not the case in real-world scenarios, where people can move in the background.

2.4 Related Work

Previously, there has been some related work done using Microsoft Kinect or similar depth sensors for hand gesture recognition. This related work is reviewed below.

Matthew [12] proposed a hand gesture recognition approach using Kinect depth sensor. This approach uses depth data along with a skin detector to increase the reliability and robustness of hand detection algorithm. It also takes the assumption that the distance between Kinect sensor and user remains constant, making it not suitable for situations where there is a change

in scale. They calculated rotation angle of the hand using full body tracker, which is then used to account for any rotation due to hand movements. A modified SURF features based approach is used to recognize the gestures against a small database. The depth data itself is not used for recognition, however it is used in increasing the accuracy of identifying the pixels that corresponds to hand region. They used a combination of static gesture to make a dynamic gesture. These gestures are highly distinguished, therefore once a gesture is detected, the system has a high probability of going to the state corresponding the other gesture. This probability is used to further enhance the accuracy of this system. They were able to achieve 96.1% accuracy. The gestures used in this approach are very limited and are highly distinguished from each other. This is not the case in real world scenarios. There are a lot of natural gestures which are not easily distinguished from each other, posing a need for a better recognition technique.

Van et al. [28] proposes another method which focuses more on gesture recognition than hand segmentation. They use built-in OpenNI/NITE hand tracker to segment the hand using depth information. This approach might have a limitation as it requires the user to perform a gesture for initialization, however it is efficient and robust solution for tracking hand as compared to techniques discussed in section 2.2. After localizing hand region, this approach estimates its orientation, recognizes the posture and estimates the pointing direction. The gestures used in this paper are all static symbolic gestures, however they use the depth data along with rgb data to calculate haarlet coefficients. These coefficients are compared to an already trained database using nearest neighbour search, and the resulting gesture is then recognized. Using the identified gesture and orientation of hand, this method is able to detect actual pointing direction in 3D. This work is an extension of this author's previous work [49], in which time-of-flight(TOF) camera was calibrated along with an RGB camera to produce a depth map similar to the depth map given by Kinect Sensor. They used skin and depth information for tracking hand, using skin colour from face region. This tracker was a significant improvement over previous works done [25].

In [43] a new action recognition approach is proposed which uses a bag of 3D points. This method first samples the depth points in equal intervals and then uses them to construct silhouette projections of the actions in different Cartesian planes. A collection of these projections over time, are used to define an action. Comparison of different actions is done using action graph approach [44]. Using this method, the author is able to achieve above 90% recognition accuracy while only utilizing 1% of the total 3D points from the depth maps. When compared to 2D silhouette methods, this is more efficient and robust.

The method proposed in [13] for recognition is simple and useful as it uses hand shape for recognition [50]. This method can be further modified and used for such comparisons in 3D. This is discussed in more detail in the next chapter.

The proposed approach for this project uses a method similar to the construction of action

graphs in [43] and is sufficient for modelling complex gestures in 3D. This is combined with segmentation technique similar to the hand region segmentation approach used in [28]. Our proposed method is discussed in more detail in next chapter.

Chapter 3

Method

This chapter explains the proposed method in more detail. This project is divided into five major steps. Figure 3.1 shows the flow chart of the proposed method. These steps are discussed in detail in the following sections.

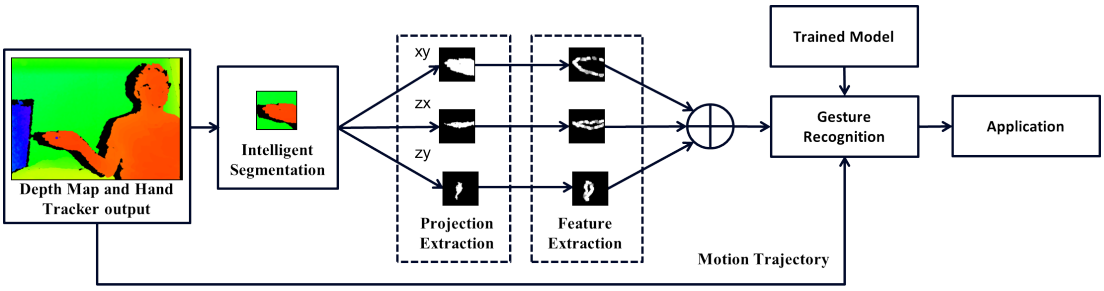


Figure 3.1: Flowchart of the Proposed Method

3.1 Hand Tracking and Segmentation

As the Kinect hardware was originally designed for gestures using five extremities of human body [4], it does not provide sufficient resolution to capture small details like hand postures efficiently. In addition to low resolution, there is also error noise which is directly proportional to the distance and size of the hand. This section looks into the challenges faced in segmenting and extracting relevant features and the solutions proposed to cater for these limitations.

3.1.1 Hand Tracking

The first step in the proposed approach involves localizing and segmenting the hand region. It was seen from the literature review in Chapter 2 that localizing and tracking hand region was one of the major challenge in most of the previously proposed methods. A reliable and robust hand tracking has always resulted in adding more complexity to the system, making real-time gesture recognition a challenging problem.

In [28], the author used a hand tracker presented in OpenNI/NITE library. OpenNI is an open-source library used for accessing data from Kinect sensor, however NITE implementation is not open-source and hence the method used by this tracker is not explained in the literature. This tracker is both reliable and robust for a real-time application as it operates in real-time at 30 frames per second. One limitation for this approach is that it requires the user to perform a wave gesture to initialize the tracker. Another limitation is the absence of a model, which makes it difficult to reacquire hand region if it is lost due to self occlusion or error noise with distance. These limitations are justified by the fact that the main aim of this project is to build a hand gesture based user interface for which there is a strong need for the hand tracker to be real-time. The output from the OpenNI/NITE hand tracker is a hand point defining the location of the hand in real-world 3D coordinates. This hand point is important information for our approach and has been utilized to perform many operations throughout this project. We define this hand point as a point with coordinates (P_x, P_y, P_z) for further reference in next sections.

3.1.2 Intelligent Segmentation

(a) at 700mm (b) at 1700mm

Figure 3.2: Depth map with hand at different distances from Kinect sensor

After localizing hand region, the first step is to segment the hand. The simplest method

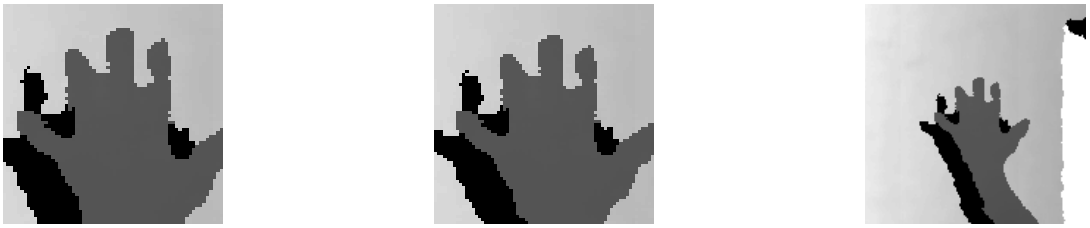
to achieve this is to segment using a fixed sized segmentation window around hand point. However, using this method can lead to undesirable results, as the apparent size of the hand might change depending upon its distance from the sensor. This results in unwanted arm region being included or in segmenting out some hand regions themselves (see Fig. 3.3 and 3.4).

(a) Intelligent Segmentation 156x156 Pixels

(b) Fixed Segmentation 70x70 Pixels

(c) Fixed Segmentation 150x150 Pixels

Figure 3.3: Segmentation output for Fig.3.2(a)



(a) Intelligent Segmentation 64x64 Pixels

(b) Fixed Segmentation 70x70 Pixels

(c) Fixed Segmentation 150x150 Pixels

Figure 3.4: Segmentation output for Fig.3.2(b)

In this project we propose a robust and reliable method to extract hand regions from depth map. This method utilizes the hand point output from hand tracker. The basic idea is that given the distance of the hand from the sensor, the area of hand region in a given frame can be calculated. To establish a relationship between this segmentation size and the distance, we captured a number of depth maps with varying distance of hand from the camera. An open hand pose was used for comparison. As an example two frames with this hand pose at distance 700mm and 1700mm are shown in Fig. 3.2(a) and (b). In both images, hand point location is marked with a white dot on the palm of the hand. It can be observed from these figures that apparent size of the hand decreases with increased distance from the sensor. This is used to build a inversely proportional relationship between size of the hand and it's distance from the sensor. To find out the exact relation, we use the following equation, where S is the length of

one side of the segmented image and D is the distance of the hand from the sensor:

$$S \propto \frac{1}{D} \tag{3.1}$$

This equation can be further evaluated by adding a constant K and then solving for this constant.

$$S = \frac{K}{D} \tag{3.2}$$

This constant K is found by first cropping the hand region in 3.2(a) and (b). For simplicity, the cropped frames are kept to be square. After this, the values for S and D are used to calculate the constant K to be 108000. Using this constant in equation 3.2, the relationship between S and D becomes:

$$S = \frac{108000}{D} \tag{3.3}$$

An intelligent segmentation is done around the hand region using equation 3.3 and hand point as the origin for segmentation. Example output frames of intelligent segmentation for Fig. 3.2 are shown in Fig. 3.3(a) and 3.4(a).

3.2 Feature Extraction

One of the most important step in any computer vision application is the extraction of relevant features. This extraction step should focus on both extracting relevant information and minimizing out all the outliers. This process can then be followed up by much more complex operations only on relevant and small set of features. An efficient feature extraction step is essential for the implementation of a real-time system. This is because these systems have limited time to process and present the results. The proposed system uses several techniques to detect and extract hand features which are explained in subsequent sections below.

Following the intelligent segmentation, the next step in the system is to extract the relevant features. There are a number of techniques presented in the literature to cater for this problem. These approaches were discussed in detail in Chapter 2.

The approach used for feature extraction in this project is similar to [51] and [43]. Using the segmented hand region, different projections are first extracted. These projections are then used for contour feature extraction, as they define the shape of the hand pose. The proposed approach is presented in detail in the next subsections.

(a) Xy Projection (front view)

(b) Zx Projection (top view)

(c) Zy Projection (side view)

Figure 3.5: Projection Extraction Output for Fig.3.2(a)

3.2.1 Projection Extraction

The contribution of this work is to use a series of hand postures to define a dynamic hand gesture. This involves the use of a silhouette projection based approach similar to the action graph approach in [43]. The segmented hand region from intelligent segmentation step is used to construct three projections in xy, zx and zy plane. These projections provide the front, top and side view of hand respectively. Each of these projections are stored as a mask and used in the next step to extract the features. Figure 3.5 shows the constructed projections of hand region for Fig. 3.2(a).

To extract xy projection, the side length S from intelligent segmentation step is used to segment the hand in depth. Hand point location is used as a reference. The segmentation for xy projection is done in z axis, with thresholding done at $P_Z \pm \frac{S}{2}$. This creates an imaginary bounding box in 3D around the hand region, which segments it from the background. This algorithm is shown in detail in Alg. 1. The output is a binary mask image with front view of the hand as shown in Fig.3.5(a).

Algorithm 1: XY_PROJECTION($P_x, P_y, P_z, S, HandImg, XyProj$)

1: Initialize $XyProj$ with $Size(HandImg)$
2: Set $XyProj$ to $Zero$
3: Iterate through $HandImg$ with value($HandImg_x, HandImg_y, HandImg_z$)
4: **if** ($HandImg_z > P_z - \frac{S}{2}$ AND $HandImg_z < P_z + \frac{S}{2}$) **then**
5: Update $XyProj(HandImg_x, HandImg_y) = 1$
6: **end if**

The next projection extracted is zx projection. A simple algorithm is used to extract this projection, which utilizes the hand point as a reference point. The projection is calculated by a similar approach as xy projection, but the construction of mask is done in a different way. The

Algorithm 2: ZX_PROJECTION($P_x, P_y, P_z, S, HandImg, ZxProj$)

1: Initialize $ZxProj$ with $Size(HandImg)$
2: Set $ZxProj$ to $Zero$
3: Iterate through $HandImg$ with value($HandImg_x, HandImg_y, HandImg_z$)
4: **if** ($HandImg_z > P_z - \frac{S}{2}$ AND $HandImg_z < P_z + \frac{S}{2}$) **then**
5: Update $ZxProj(HandImg_x, HandImg_z - P_z + \frac{S}{2}) = 1$
6: **end if**

Algorithm 3: ZY_PROJECTION($P_x, P_y, P_z, S, HandImg, ZyProj$)

1: Initialize $ZyProj$ with $Size(HandImg)$
2: Set $ZyProj$ to $Zero$
3: Iterate through $HandImg$ with value($HandImg_x, HandImg_y, HandImg_z$)
4: **if** ($HandImg_z > P_z - \frac{S}{2}$ AND $HandImg_z < P_z + \frac{S}{2}$) **then**
5: Update $ZyProj(HandImg_z - P_z + \frac{S}{2}, HandImg_y) = 1$
6: **end if**

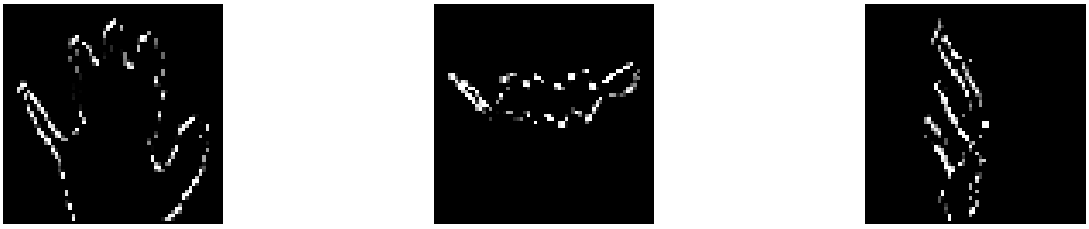
zx projection mask contains output for every z and x coordinates where a hand exists. This is achieved by just setting those values where hand region is found using z and x coordinates from the segmented hand region. The output of this step is shown in Fig.3.5(b). Similarly, the same method is used to construct the zy projection. However instead of taking z and x coordinate axis, this projection uses z and y coordinates for constructing the projection mask. Both these algorithms are summarized in Alg. 2 and 3 respectively. The output of these methods are shown in Fig.3.5(b) and 3.5(c) respectively.

3.2.2 Contour Extraction

The features used in this project for recognizing a set of hand postures are contour features. Further processing is done on projections to extract these features.

There are some errors seen in these projections in the form of black spots. To remove these errors, simple morphological operations are performed with a structuring element 5x5 circular disk for xy projection and 9x9 disk for zx and zy projections. The size of structuring element for zx and zy projection is kept more as these projections contain more errors. As the hand is moved away from the sensor, there are more errors induced in the segmented masks. To deal with these errors a number of techniques are utilized. These problems with their solution are discussed in subsection 3.2.3.

After initial post processing, all of the projection masks are normalized to a size of 64x64



(a) Xy Projection (b) Zx Projection (c) Zy Projection

Figure 3.6: Derivative of Projection Masks in Fig.3.5

(a) Xy Projection (b) Zx Projection (c) Zy Projection

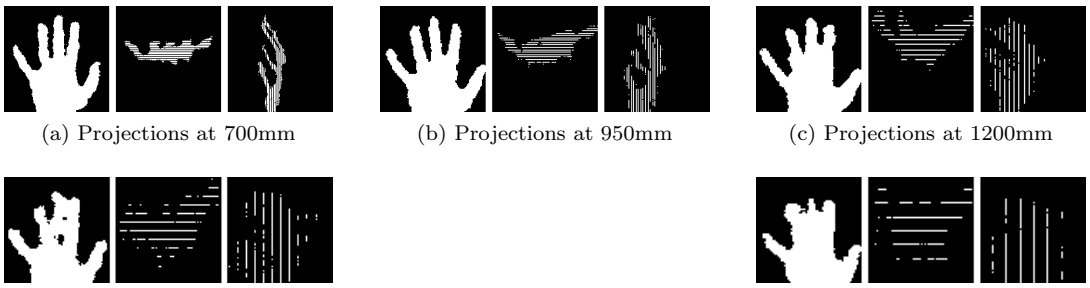
Figure 3.7: Dilated output of Extracted Contours in Fig.3.6

pixels. This normalization is done because size of each of the projection mask is different, depending on its relation with the distance (Equation 3.3). This is then followed by taking derivative of the image using sobel edge detector. Sobel edge detection is used as a robust technique was required. A detailed comparison of edge detectors for recognizing objects is presented in [52]. From their findings, sobel operator has the best trade off for low execution time and moderate accuracy. This makes sobel edge detector suitable for our proposed approach. The output from this step is a 64x64 mask containing the contour of the hand. To cater for differences in style and hand movements, we dilated this contour image with a rectangular 3x3 structuring element. The output of sobel operator and extracted contour after dilation is presented in Fig. 3.6 and 3.7 respectively.

3.2.3 Challenges faced and proposed solutions

The feature extraction steps discussed in this section are able to extract contour feature efficiently. However there are still some limitations imposed by how Kinect works. This subsection discusses these challenging problems and the proposed solutions used in this project.

Challenges faced



(a) Projections at 700mm (b) Projections at 950mm (c) Projections at 1200mm

(d) Projections at 1450mm (e) Projections at 1700mm

Figure 3.8: Projections of same hand pose at different distances from camera

First of all, the depth map is stored in a matrix which has a discrete x and y coordinates. This quantized data gives rise to discontinuities when extracting zx and zy projection using Algorithm. 2 and 3 respectively. There is no quantization error in xy projection as in that the segmentation is a direct relationship of the x and y coordinates in segmented depth map. As the distance of the hand increases from the sensor, the hand area decreases in the segmented depth map. This further increases the gap between the quantized values for hand region. To depict this problem, zx and zy projections for the same hand pose are shown in Fig. 3.8 at varying distance from the sensor. Another observation from these figures is that the error in depth calculation is increased, as the hand is moved away from the sensor. As a result of this, there is a lot of random noise in the data, making it difficult to reconstruct the actual shape using zx and zy projections at distances equal and greater than 1700mm. To solve this problem, two different techniques are proposed which, when combined, are able to reconstruct these projections. These approaches are explained below in subsequent sections.

Morphological Filtering

The first approach used to solve the error arising from quantization of x and y coordinates in depth map is to do morphological filtering. The approach uses two types of structuring elements, depending on the orientation of quantization gap, to perform a morphological closing operation. This closes the gap to some extent. These structuring elements are horizontal line and vertical line of size 3x3 (Fig. 3.9). For zx projection, the discontinuities are oriented in horizontal direction, so therefore a vertical line structuring element is used to perform closing operation this projection (equation 3.4). Similarly for zy projection, the discontinuities are oriented directly opposite direction, i.e. they are vertically oriented, therefore a horizontal

(a) Vertical line SE (b) Horizontal line SE

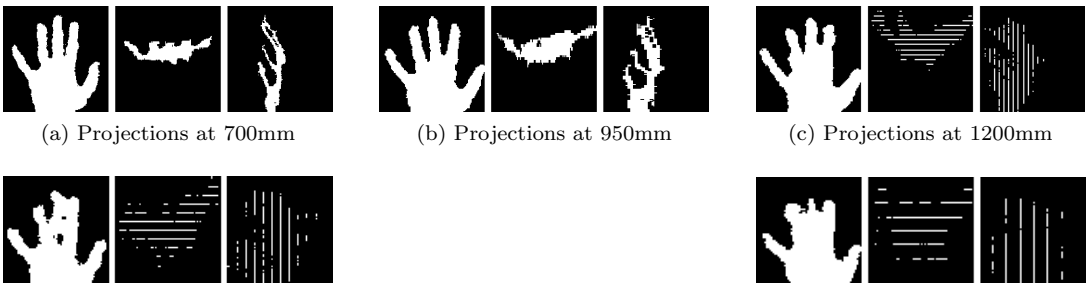
Figure 3.9: Structuring elements for morphological closing

structuring element is used for morphological closing to diminish their effect (equation 3.5). The equation for these operations are given below:

$$(Proj_{zx} \oplus V_{SE}) \ominus V_{SE} \tag{3.4}$$

$$(Proj_{zy} \oplus H_{SE}) \ominus H_{SE} \tag{3.5}$$

The output of these operations on Fig. 3.8 is shown in Fig. 3.10. These figures show that the morphological closing operation only works if the distance of the hand region is less than or equal to 950mm from the sensor. This is due to lower magnitude of discontinuities at less distances. To make this morphological operations work with distances greater than 950mm an interpolation step is added before performing these morphological operations. This interpolation is explained in detail in the next subsection.



(a) Projections at 700mm (b) Projections at 950mm (c) Projections at 1200mm

(d) Projections at 1450mm (e) Projections at 1700mm

Figure 3.10: Projections from Fig.3.8 after morphological closing

Interpolation

The problem of discrete quantization for x and y axis is partly solved by introducing a post processing step involving the use of morphological closing with special structuring elements. However, it is observed that for distances greater than 950mm the morphological operations are not sufficient in diminishing this effect. The approach used to further solve this problem is to interpolate the regions of discontinuities by a simple averaging based interpolation.

Algorithm 4: INTERPOLATE_PROJECTION($Steps, Proj, prev_x, prev_y$)

Iterate through $Proj$ with value(x_i, y_i)
if $(Proj(x_i, y_i) == 1)$ **then**
  Update $prev_x = x_i$
  Update $prev_y = y_i$
  if $(isInsideWhiteRegion == true)$ **then**
    Interpolate $Proj$ with $P1(prev_x, prev_y)$, $P2(x_i, y_i)$ and $Steps$
  else
    Update $isInsideWhiteRegion = true$
  end if
else
  Update $isInsideWhiteRegion = false$
end if

Algorithm 5: INTERPOLATE($Proj, P1(prev_x, prev_y), (x_i, y_i), Steps$)

Perform recursive calls until number of steps equal 0
if $(Steps > 0)$ **then**
  Update $Proj(\frac{(prev_x + x_i)}{2}, \frac{(prev_y + y)}{2}) = 1$
  Update $Steps = Steps - 1$
  INTERPOLATE$(Proj, P1(prev_x, prev_y), P2(\frac{(prev_x + x_i)}{2}, \frac{(prev_y + y_i)}{2}), Steps)$
  INTERPOLATE$(Proj, P1(\frac{(prev_x + x_i)}{2}, \frac{(prev_y + y_i)}{2})), P2(x_i, y_i), Steps)$
end if

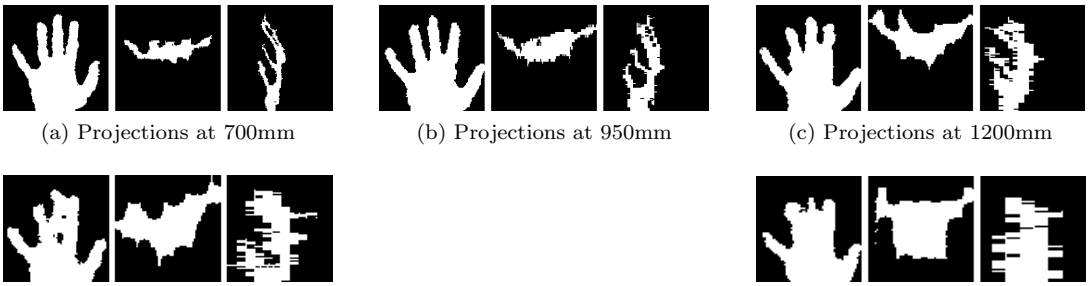
This interpolation is done using a recursive function which is simple yet effective for our implementation. Algorithm 4 explains the method involving the detection of regions with discontinuities, whereas Algorithm. 5 explains the recursive method used for interpolating two points. The basic idea is, if a point in a projection mask is equal to 1, then store it as previous point and use it together with the next point equal to 1 to call interpolate function. The interpolate function works by setting values at midpoint of the two points. Then, depending on the number of steps provided, it calls a recursive call with points which are in between.

It was found out by trying out different values for number of steps that interpolation with 2 steps combined with morphological closing produces a completely reconstructed projection. The computation power required by this interpolation step is directly proportional to the size

(a) Projections at 1200mm (b) Projections at 1450mm (c) Projections at 1700mm

Figure 3.11: Interpolated projections from Fig.3.8

of the input projection and the number of steps. Therefore, this interpolation is only performed when the hand is further than 950mm i.e. only when it is required. Figure 3.11 shows the output of this interpolation step. When combined with the morphological filtering steps, the output is seen in 3.12. From these figures it can be seen that the projections are completely reconstructed, but have a lot of random noise as the distance is increased. This is the reason for imposing a limitation on the operation of this project, as beyond 1700mm it becomes very difficult to accurately estimate the shape of hand in 3D. Another reason is that the resolution size for hand region becomes smaller than 64x64 pixel, making it difficult to estimate the shape of the hand.



(a) Projections at 700mm (b) Projections at 950mm (c) Projections at 1200mm

(d) Projections at 1450mm (e) Projections at 1700mm

Figure 3.12: Reconstructed projections from Fig.3.8

3.3 Model Construction and Recognition

The first step to train a model is to define the relevant stages in a dynamic gesture. The contribution of this project is to model two dynamic gestures using different stages. These

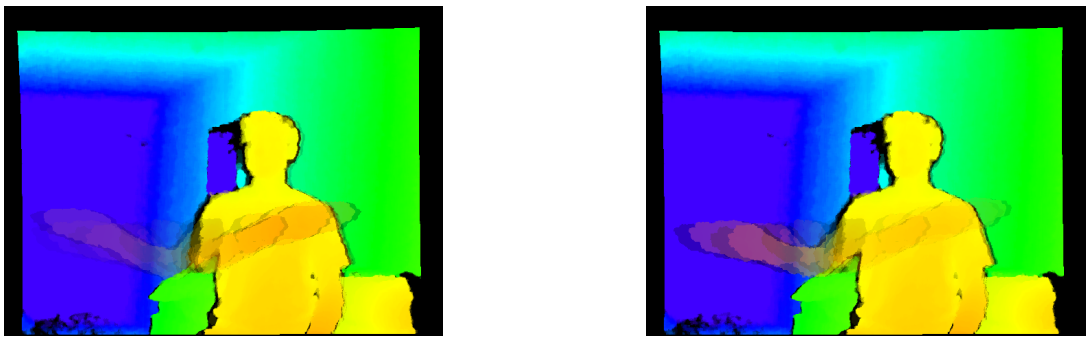
gestures are explained in detail in the sections below, along with their model construction and recognition steps.

3.3.1 Gestures Used

In almost all previous approaches used to model gestures, less importance was given on the choice of a natural and user friendly gesture. In most of the previous research, simple static or dynamic gesture were used. These gestures were not user friendly, but they were easy to recognize on computers (see Chapter 2). In this project, we have modelled two of the most natural and commonly used gestures. This choice makes the user interaction experience more natural and user friendly.

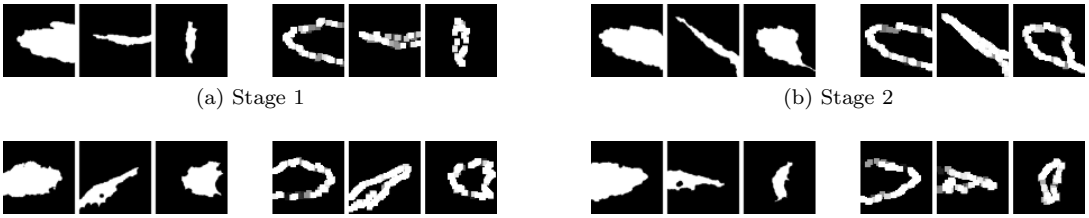
The gestures we use are swipe gestures, which are used everyday with our interaction with printed media. These gestures are natural and have become part of our daily routine, so much so that we don't even think while using these gestures. The reader of this report might have already used these gestures several times now, without even thinking about it, unless if this report is being read on a computer screen, where the interaction experience is not so natural.

Two basic versions of swipe gesture are used, namely swipe left and swipe right. Figure 3.13 shows these two gestures as a motion history image of a sequence of 10 frames. These two gestures are defined by four different postures. For easy comparison, we assign each of these postures a stage number. These stages are shown in Fig. 3.14. Both gestures can be modelled using same four stages. This is due to the fact that the two gestures we use are symmetric, using same four pose stages. The only difference is that the order of detected stages is reversed. This is explained in detail in section 3.3.3.



(a) Swipe Left (b) Swipe Right

Figure 3.13: Gestures Modelled



(a) Stage 1 (b) Stage 2

(c) Stage 3 (d) Stage 4

Figure 3.14: Defined stages for swipe gesture

3.3.2 Model Construction

Following the extraction of features in feature extraction step, the next step is to model and recognize different patterns. Each stage in the gesture results in a different pattern, however these patterns are not linearly separable. As seen in Fig. 3.14, there is an interclass overlap in most of the contour region. These complex non-linear patterns can be easily modelled using neural network based pattern recognition [53]. The approach used in the project has been used to model face detection using neural networks [54]. Similar to our method, in [54] the author normalizes and segments the face region pixels to images of size 20x20. These are then used to train a neural network based model.

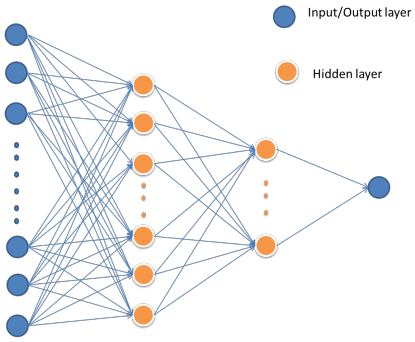


Figure 3.15: Four layered structure of Neural Network

To train the neural network, 1500 samples were collected for each stage in the gestures. The data collection was done carefully with enough variations to model each stage perfectly. To cater for noise with distance, the samples were collected at different distances from the sensor. Projection and features were then extracted from these samples, which were then concatenated and reshaped into a row matrix. A neural network with four layers was trained. The structure of this neural network is shown in Fig. 3.15. Out of these four layers, 2 were input/output layers,

while other two layers were hidden layers. Each stage pose was given a class number starting from 1 to 4. First layer in the neural network contained the same number of neurons as the input row matrix i.e. 64x64x3, whereas the last layers contained only one neuron for identifying the input class. Number of neurons chosen for hidden states are 300 and 15 for the second and third layer respectively. This neural network is trained using back-error propagation algorithm as presented in [55]. In this project we have used the OpenCV machine learning library to train an artificial neural network and use it to classify a given pose. To verify the accuracy of the trained neural network, we tested using testing data. The results of the accuracy tests are presented in Chapter 4.

3.3.3 Gesture Recognition

The gesture recognition part of this project uses several techniques to identify each gesture. First of all, the output of neural network is used as a sequence of observations. The identified gesture directly depends on these observations. Ideally, the output observations from neural network for a swipe left gesture are $1- > 2- > 3- > 4$. Similarly, for a swipe right gesture, the output of neural network is reversed to $4- > 3- > 2- > 1$ as it is symmetric. There are some errors in the transition regions for some of the stages. These errors are directly dependent on the limitation of Kinect sensor for detecting small objects. When hand is directly pointing toward the sensor, there are some discontinuities as discussed in Chapter 2. To cater for this, we train two HMM for each gesture. The training process involves initializing the transition, observation and initial state matrices with some random guess values. These matrices are then trained using a sequence of observations. Training for these two HMM corresponding to each of the gestures was done using baum-welch algorithm [56]. This algorithm is a generalized expectation maximization algorithm. The sequence of observations used to train HMM for swipe left gesture are $1- > 2- > 3- > 4$, where as for swipe right gesture the sequence used was $4- > 3- > 2- > 1$. We used HMM toolbox in matlab for training HMM. The trained output for transition, observation and initial state matrices were then copied to our code in c++. In our code, we used a c++ implementation of HMM to calculate the log-likelihood of each HMM to recognize the gestures. To recognize a gesture, last 10 observations were recorded. This observation sequence was then used with each of the HMM to calculate its log-likelihood. The model with the lowest log-likelihood was declared as recognized gesture. The two trained HMM are presented in equations below, with left matrices for HMM corresponding to swipe left gesture and matrices named right for HMM corresponding to swipe right gesture. It can be observed that the only difference between these HMM is the emission matrix. This is due to the reason that this matrix corresponds directly to the observations. It can be seen that the defining observation for these models is the first and last observation, i.e. pose stage 1 and pose

stage 4 respectively.

$$InitLeft = \begin{bmatrix} 0 & 1 \end{bmatrix}, TransLeft = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}, EmisLeft \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0.33 & 0.33 & 0.33 & 0 \end{bmatrix}$$

$$InitRight = \begin{bmatrix} 0 & 1 \end{bmatrix}, TransRight = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}, EmisRight = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.33 & 0.33 & 0.33 \end{bmatrix}$$

Secondly, to increase the accuracy, this step also uses motion trajectory of the hand. In order to achieve this, a list of last 10 hand points is maintained. When a gesture is detected by HMM, motion trajectory is calculated by finding out distance vector of each point with the previous points. A simple threshold set on the displacement helps to identify the difference between an actual gesture and a false one. For swipe left, the threshold in x is set to -80, whereas for swipe right its 80 as the hand moves in the opposite direction. If the detected gesture has a threshold greater than this threshold value, only then it is detected as a recognized gestures. Another threshold is applied in y axis, where the movement is minimum if performing a swipe gesture. We found out using different experiments, that while performing swipe gestures the trajectory value for previous 10 points remains lower than a value of 30.

3.3.4 Application

The successfully recognized gestures are mapped on to different functions in a number of different applications. These applications include presentation, picture viewer, media player and internet browsing applications. We have made use of Windows API for the application part, hence making it independent of the controlled application. The proposed approach can be used with any application that uses left and right navigation. The aim is to navigate to next and previous step/stage in an application using swipe left and swipe right hand gestures.

3.4 Comparison of OpenNI with Microsoft Kinect SDK

Table 3.1 shows the comparison we did between two programming SDK's that are used with Kinect Sensor. OpenNI is a cross platform SDK, which can be used on any platform. It is also independent of the device used, meaning we can use other devices like ASUS WAVI and Primesense with the same code. It also has build-in hand tracker, which is really useful in development of gesture recognition based applications. OpenNI has hand gesture recognition capabilities which are extensively used in this project. However, there is calibration required every time the application initializes.

OpenNI SDK	Kinect SDK
Framework for hand tracking	No hand tracking
Framework for hand gesture recognition	No hand gesture recognition
Full body tracking: - Need Calibration Pose when Initialized - Occluded joints not estimated - Support hand only mode - Joint Rotation - Consumes less cpu power	Full Body tracking: - No calibration required - Better with occluded joints - No hand only mode - Only calculates position of joints (no rotation)
Support for other sensors, Primesense and Asus wavi	Only Supports Microsoft Kinect
Open source	Not Open source
Available for Windows, Linux and OSX	Only available for Windows
Difficult to setup	Easy to setup
No audio support	Audio support
No motor tilt	Motor tilt

Table 3.1: Comparison between OpenNI and Kinect SDK

On the contrary, Kinect SDK can only be used with Kinect sensor. The functions in this SDK are more accurate in estimating occluded joints. Unlike OpenNI, Kinect SDK does not require any calibration but it lacks the presence of a built-in hand tracking algorithm. We have used OpenNI SDK for the development of this project, as it is much more flexible and platform independent. The main reason for choosing OpenNI/NITE SDK is that there is a hand tracking algorithm which can be used to track and segment hand region in a robust and accurate manner. Another reason is its cross platform compilation capability and independence from device used. This means the code we wrote can be used on a number of depth sensors using any platform. Future work can be extended easily on any device with any language, which is a major advantage of this implementation. Additionally, OpenCV libraries are extensively utilized throughout this project, which are also suitable for development across different platforms.

Chapter 4

Results and Comparisons

This chapter discusses the performance of the proposed approach. To test the performance, an application is implemented, which uses windows api to control left and right navigation functions for a number of different related applications. This interface can be used with picture browsing, internet browsing, media player and presentation softwares. The output of this interface was a real-time application executing at 25 frames per second. This application was tested on Intel Core 2 @ 1.86 Ghz processor with 2 GB RAM. The implementation was done in C++ using OpenCV, OpenNI and NITE libraries.

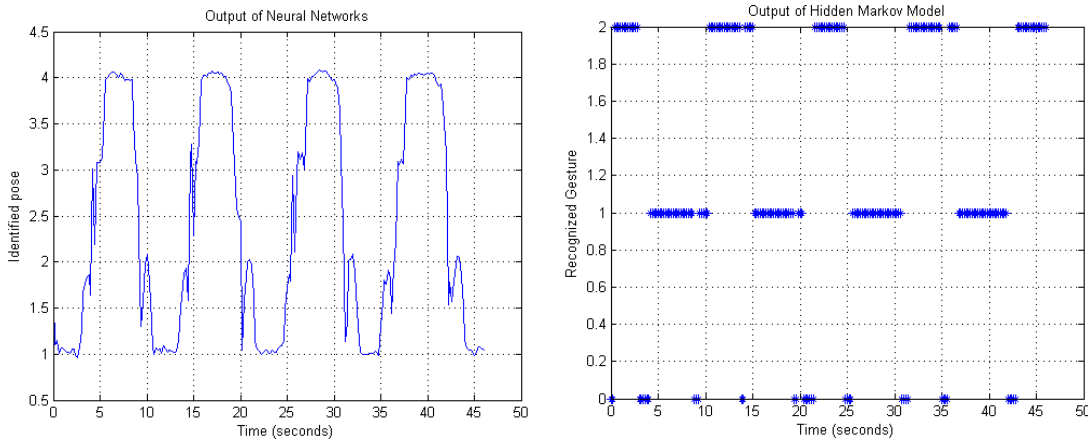
4.1 Results

Pose Number	Accuracy
1	0.96
2	0.95
3	0.92
4	0.95

Table 4.1: Accuracy for Neural Network Output of each trained pose

To analyse the accuracy, the trained neural network based model was tested with a sequence of 2000 sample hand images for each stage in the gesture. The resulting accuracy was generated using the following equation:

$$Accuracy = \frac{truepositive}{truepositive + falsepositive} \tag{4.1}$$



(a) Neural Network Output (b) Hidden Markov Model Output

Figure 4.1: Recognition Output for an input sequence with repeated Swipe left and right gesture

The accuracy for each stage found using equation 4.1 is summarized in Table 4.1. These results show that the average accuracy for neural network output is 0.94 or 94%.

The neural network output for different stages in a repeated swipe left and right gesture is shown in Fig. 4.1(a). From this figure, it can be observed that the output for identified pose is fluctuating from pose 1 to 4 and then back to 1 for a swipe left and right gestures perform simultaneously. There is some discontinuity seen in between these postures,because of the error due to hand pointing directly towards the sensor (as discussed in section 2.1.4).

The output from neural network is a sequence of integers defining the transition between each stage of the defined gestures. To detect this sequence accurately, HMM is used. Figure. 4.1(b) shows the output of HMM for the sequence in Fig. 4.1(a). The output of Hidden Markov Model are three hidden states, which are directly dependent on the input observations from Neural Networks output. As seen in Fig. 4.1(b), state 1 and 2 correspond to swipe left and swipe right gestures respectively. Whereas state 0 is a NULL state, which means that no gesture is detected. The addition of Hidden Markov Model increases the reliability of the overall system, so even if one of the observations are missed by neural network, the HMM will try to match it with the best matching state.

The neural network model was also tested against scale changes in pose. To record these results, the hand was placed in front of sensor with a given pose stage. This hand was slowly moved away from the sensor, while recording the output of neural networks along with the distance of hand from the sensor. This depicted the difference between a person using the

interface at different distances from the sensor. Figure 4.2 shows the output of neural network against the distance of the hand from the sensor. It is observed from this figure that the output of neural network is accurate till 1600mm, after which it starts giving false positive results. These false predictions are the result of the limitations discussed in Chapter 3. To make a good comparison of our proposed approach with previous research, we compare these results with the output of Hu invariant moments of the same postures. The results are presented and discussed in the next section.

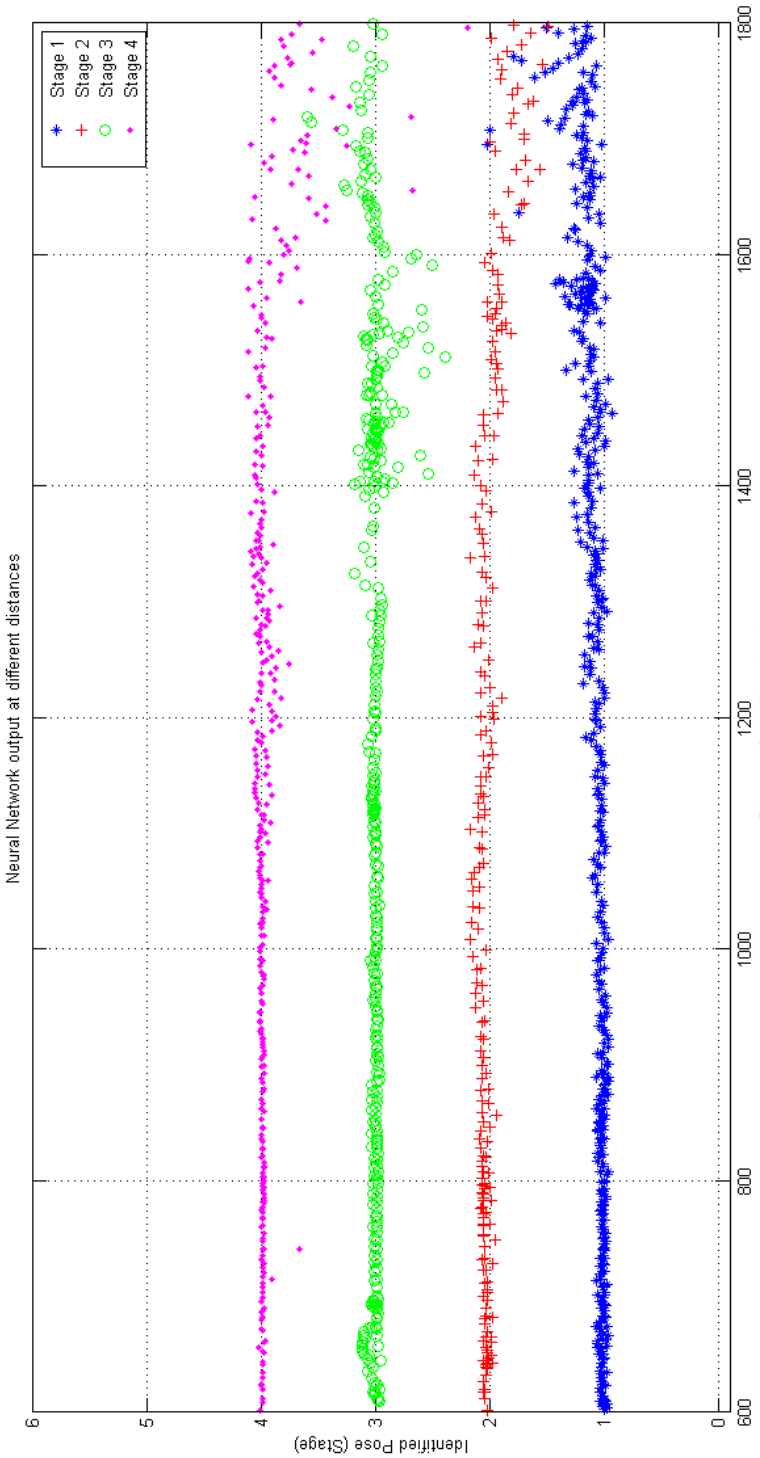
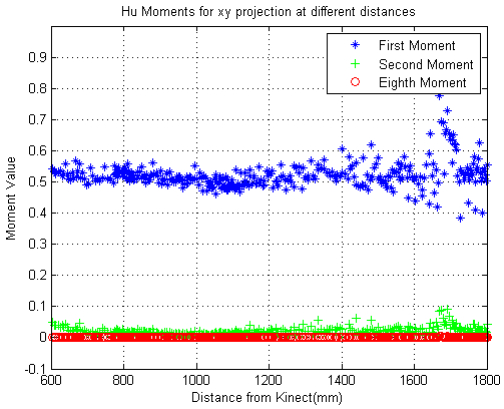


Figure 4.2: Neural Network output of four different pose stages at varying distance from the sensor

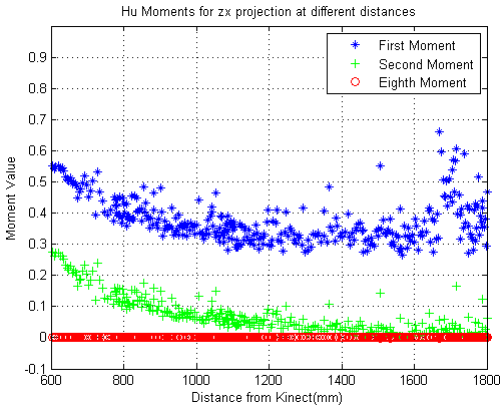
4.2 Comparison with Hu Moment based feature extraction

The proposed approach uses contour features to recognize individual postures. In our research for this project, we also came across Hu moments based feature extraction for silhouette images. As discussed in Chapter 2, Hu moments are a good choice for feature extraction as these moments are rotation, translation and scale invariant. For our project, these moments would have been a good feature extraction technique. However, due to the noise errors induced by increased distance of hand from the camera and relatively small size of the hand region, this method was not accurate in detecting a specific posture [47]. To test Hu moments with our system, we extracted the first, second and eighth moments, of all four stages in our dynamic gesture. These moments were chosen as these are the most defining moments for a given shape [48]. These moments were extracted while varying the distance of the hand from the sensor. To analyse the output, these moments were then plotted against their corresponding distance values. The results are presented in Fig. 4.3 - 4.6.

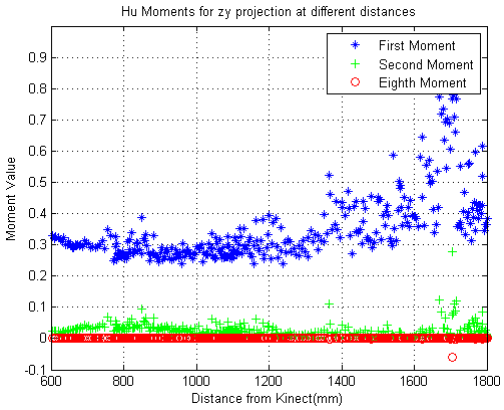
Analysing these figures, the first thing that we can notice is that the moments do not change much for different pose stages. All values for moment range in the same area, irrespective of the pose. Secondly, there are a lot of variations seen for first hu moment. These results also support the findings in [47], where they found out that, for binary images below the resolution of 150x150 pixel, Hu moments are not effective. Therefore, the method we proposed used more details from the contour instead of extracting features like Hu moments as the size of the hand image is too small for Hu moments to be effective.



(a) Hu Moments for Xy Projection

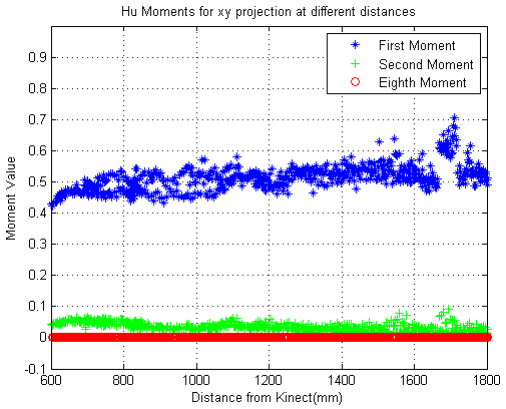


(b) Hu Moments for Zx Projection

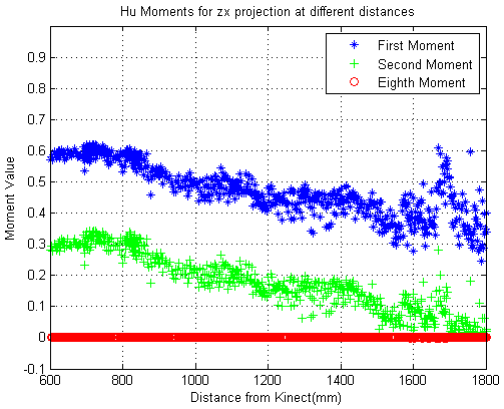


(c) Hu Moments for Zy Projection

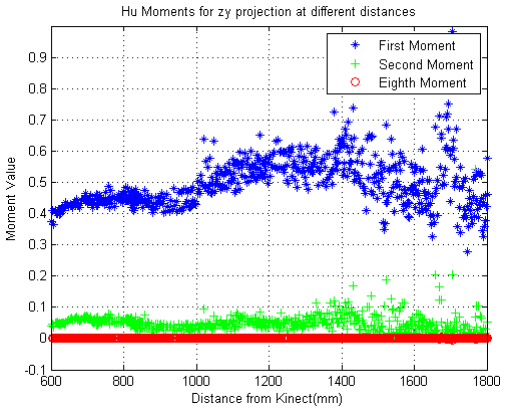
Figure 4.3: Hu Moments for Stage 1 at different distances



(a) Hu Moments for Xy Projection

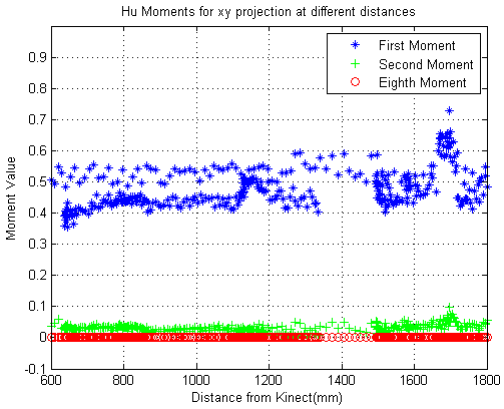


(b) Hu Moments for Zx Projection

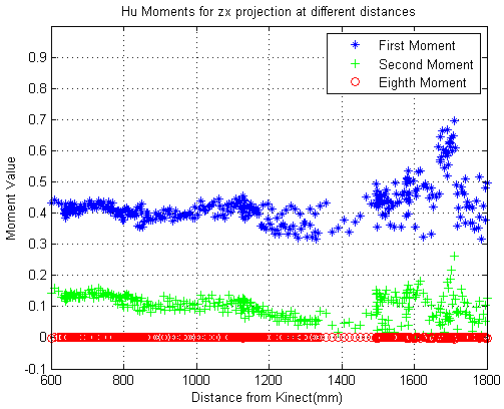


(c) Hu Moments for Zy Projection

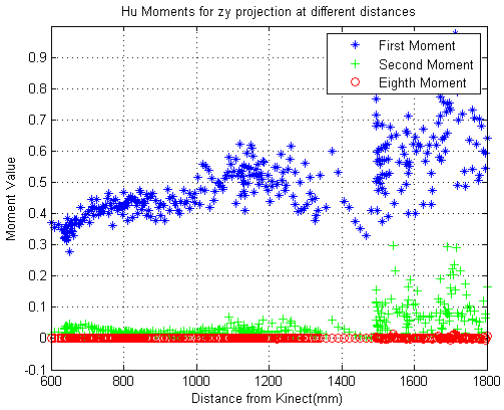
Figure 4.4: Hu Moments for Stage 2 at different distances



(a) Hu Moments for Xy Projection

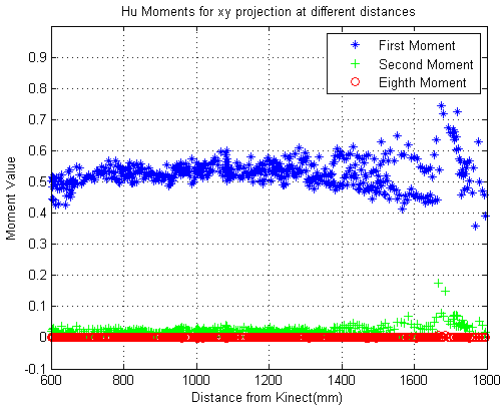


(b) Hu Moments for Zx Projection

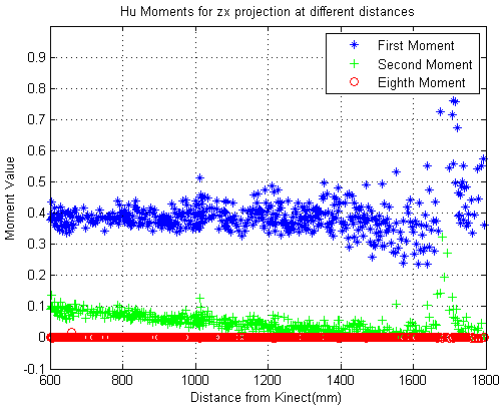


(c) Hu Moments for Zy Projection

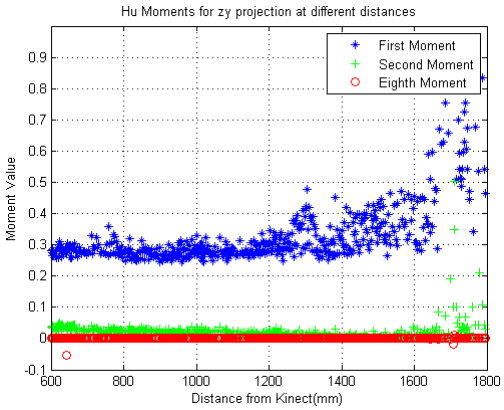
Figure 4.5: Hu Moments for Stage 3 at different distances



(a) Hu Moments for Xy Projection



(b) Hu Moments for Zx Projection



(c) Hu Moments for Zy Projection

Figure 4.6: Hu Moments for Stage 4 at different distances

Chapter 5

Conclusion and Future Work

This chapter details the conclusion and future work.

5.1 Conclusion

The introduction of Kinect sensor has revolutionized the field of Computer Vision. It has the potential to improve human-computer interface just as the iPhone revolutionized the mobile phone interfaces. At the moment, this potential is not fully utilized in literature due to limited approaches. Most of the research done has focused on hand detection and segmentation. Many researchers ignored that careful selection of hand gesture was the most integral part of making a natural and easy to use hand gesture based interface.

This project proposed an approach which was used to recognize two dynamic hand gestures. The gestures selected were swipe left and swipe right, which are the most basic, natural and easy to use gestures. Additionally, each gesture was divided into four different hand posture based stages.

A detailed analysis of Microsoft Kinect sensor was presented. This analysis gave in depth details of the sensor's limitations in context with the domain of this project. Hand region occupies a small part of the depth image, and because of the method used for depth calculation, smaller objects have a lot of errors. These errors increased if the distance of the object from the sensor was increased. Additionally, there was quantization error in x and y axis of depth map, and this quantization gap increased with increased distance from the sensor.

Hand region was tracked using OpenNI/NITE hand tracker. This tracker requires the user to perform a wave gesture to initialize the tracking process. The contribution of this work is the introduction of intelligent segmentation approach, which used distance of hand from the sensor to segment only the hand region. This approach was based on the fact that apparent size of

hand is inversely proportional to its distance from the sensor. Following this, three different projections were constructed in xy, zx and zy plane. The limitations of the Kinect sensor are that it introduces random noise and quantization gap in the projections. These limitations were overcome to some extent using morphological closing operation, with special structuring element, combined with a simple interpolation technique. The results were promising, however it was found that at a distance of 1700mm the errors in the projections were irrecoverable. This imposed a space limitation of 1700mm for our approach. Contours were extracted from these projection images. The contour images were normalized to 64x64 pixels and used directly to train a 4 layered neural network. The output from neural network was a sequence of hand pose stages, which were used by a simple hidden markov models to identify each gesture. The accuracy of system was enhanced by verifying the output with the motion trajectory of the hand point.

The proposed approach was used to implement a user interface for a variety of applications in windows. This interface produced real-time execution at 25 frames per second. The accuracy of the neural network model was found to be 94%.

5.2 Future Work

Further work on this project can realize a hand gesture based contact less user interface, which can replace already existing mechanical interfaces like mouse and keyboard. This interface will be much more user friendly and it will have the capability of adding and mapping more gestures. Essentially, much work needs to be done to improve the following aspects of this project.

1. **Reliable Hand Tracking:** The focus of this project is to recognize hand gesture, therefore OpenNI's NITE hand tracker is utilized. This hand tracker is robust and reliable than many existing approaches for tracking hand. However the lack of use of any hand model makes it vulnerable to problem hand self-occlusion, motion blur and reinitializing hand using a gesture. The future work on this project can implement a more reliable model based hand tracker, which will be capable of handling problems like self-occlusion and motion blur. This approach will be similar to [38], but more emphasis will be required to make it efficient for real-time execution.

2. **Improving the hardware:** Major challenges faced in this project arose due to limitations in Kinect sensor. In our research, we found out that the Kinect sensor is designed for detecting much larger objects. This limits reliable detection of hand region from a distance. There has been a number of attempts to improve the output of Kinect hardware, of which the most significant ones are presented in [57] and [58]. These approaches

use simple vibration devices to filter out the noise, significantly improving the output. Similarly, multiple Kinect sensors can be used with these approaches. Combining output of more than one sensor will significantly improve both the accuracy and operating range of the proposed approach.

3. **Flexible Application:** Future work can focus on online training of the interface with user defined gestures. Online machine learning techniques can be introduced which can learn and adapt themselves according to user, eventually making it more reliable and accurate. An interface can be implemented for easy addition of user defined gestures, and a wide variety of functions available to be attached with each gesture.

4. **Embedded devices:** The approach used in this project has low computational complexity, which makes it suitable for implementing it on embedded devices such as beagleboard. The implementation of this project is based on OpenCV and OpenNI libraries which can easily be ported for embedded devices like beagleboard. These portable devices can then be used for making stand-alone interfaces for remote-less television sets.

The proposed approach introduces a different dynamic gesture recognition based user interface. The work in this project can be extended and improved for further implementation of a complete contact-less computer user interface for PC and TV.

Bibliography

[1] V. Pavlovic, R. Sharma, and T. Huang, "Visual interpretation of hand gestures for human-computer interaction: A review," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 19, no. 7, pp. 677–695, 1997.

[2] B. Salem, *An analysis of hand gestures for implementation in a user interface.* University of Sheffield, Dept of Electronic and Electrical Engineering, Ph.D Thesis 2004.

[3] K. Khoshelham and S. Elberink, "Accuracy and resolution of kinect depth data for indoor mapping applications," *Sensors*, vol. 12, no. 2, pp. 1437–1454, 2012.

[4] G. Borenstein, *Making Things See: 3D Vision with Kinect, Processing, Arduino, and MakerBot.* Make Books, 2012.

[5] T. Baudel and M. Beaudouin-Lafon, "Charade: remote control of objects using free-hand gestures," *Communications of the ACM*, vol. 36, no. 7, pp. 28–35, 1993.

[6] D. Sturman and D. Zeltzer, "A survey of glove-based input," *Computer Graphics and Applications, IEEE*, vol. 14, no. 1, pp. 30–39, 1994.

[7] D. Quam, "Gesture recognition with a dataglove," in *Aerospace and Electronics Conference, 1990. NAECON 1990., Proceedings of the IEEE 1990 National*, pp. 755–760, IEEE, 1990.

[8] M. Ishikawa and H. Matsumura, "Recognition of a hand-gesture based on self-organization using a dataglove," in *Neural Information Processing, 1999. Proceedings. ICONIP'99. 6th International Conference on*, vol. 2, pp. 739–745, IEEE, 1999.

[9] L. Deng, D. Lee, H. Keh, and Y. Liu, "Shape context based matching for hand gesture recognition," in *Frontier Computing. Theory, Technologies and Applications, 2010 IET International Conference on*, pp. 436–444, IET, 2010.

[10] C. Chen, Y. Chen, P. Lee, Y. Tsai, and S. Lei, "Real-time hand tracking on depth images," in *Visual Communications and Image Processing (VCIP), 2011 IEEE*, pp. 1–4, IEEE, 2011.

[11] J. Raheja, A. Chaudhary, and K. Singal, "Tracking of fingertips and centers of palm using kinect," in *Computational Intelligence, Modelling and Simulation (CIMSiM), 2011 Third International Conference on*, pp. 248–252, IEEE, 2011.

[12] M. Tang, "Recognizing hand gestures with microsofts kinect," *Palo Alto: Department of Electrical Engineering of Stanford University:[sn]*, 2011.

[13] Z. Ren, J. Meng, J. Yuan, and Z. Zhang, "Robust hand gesture recognition with kinect sensor," in *Proceedings of the 19th ACM international conference on Multimedia*, pp. 759–760, ACM, 2011.

[14] K. Dorfmuller-Ulhaas and D. Schmalstieg, "Finger tracking for interaction in augmented environments," in *Augmented Reality, 2001. Proceedings. IEEE and ACM International Symposium on*, pp. 55–64, IEEE, 2001.

[15] O. Gallo, S. Arteaga, and J. Davis, "Camera-based pointing interface for mobile devices," in *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*, pp. 1420–1423, IEEE, 2008.

[16] B. Lee and J. Chun, "Manipulation of virtual objects in marker-less ar system by fingertip tracking and hand gesture recognition," in *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human*, pp. 1110–1115, ACM, 2009.

[17] K. Hsiao, T. Chen, and S. Chien, "Fast fingertip positioning by combining particle filtering with particle random diffusion," in *Multimedia and Expo, 2008 IEEE International Conference on*, pp. 977–980, IEEE, 2008.

[18] B. Al-Maqaleh, M. Al-dohbai, and H. Shahbazkia, "Real time fingers and palm locating using dynamic circle templates," *International Journal of Computer Applications*, vol. 41, no. 6, pp. 33–34, 2012.

[19] W. Tsang and K. Pun, "A finger-tracking virtual mouse realized in an embedded system," in *Intelligent Signal Processing and Communication Systems, 2005. ISPACS 2005. Proceedings of 2005 International Symposium on*, pp. 781–784, IEEE, 2005.

[20] T. Brown and R. Thomas, "Finger tracking for the digital desk," in *User Interface Conference, 2000. AUIC 2000. First Australasian*, pp. 11–16, IEEE, 2000.

[21] A. Tóth and A. Várkonyi-Kóczy, "Hand gesture controlled interface for intelligent space applications," in *Intelligent Engineering Systems, 2009. INES 2009. International Conference on*, pp. 239–244, IEEE, 2009.

[22] M. Hasan and P. Mishra, "Novel algorithm for multi hand detection and geometric features extraction and recognition," *International Journal of Scientific & Engineering Research*.

[23] P. Song, S. Winkler, S. Gilani, and Z. Zhou, "Vision-based projected tabletop interface for finger interactions," *Human–Computer Interaction*, pp. 49–58, 2007.

[24] C. Hsieh, D. Liou, and D. Lee, "A real time hand gesture recognition system using motion history image," in *Signal Processing Systems (ICSPS), 2010 2nd International Conference on*, vol. 2, pp. V2–394, IEEE, 2010.

[25] M. Van Den Bergh, E. Koller-Meier, F. Bosche, and L. Van Gool, "Haarlet-based hand gesture recognition for 3d interaction," in *Applications of Computer Vision (WACV), 2009 Workshop on*, pp. 1–8, IEEE, 2009.

[26] M. Kölsch and M. Turk, "Robust hand detection," in *Proc. IEEE Intl. Conference on Automatic Face and Gesture Recognition*, pp. 614–619, 2004.

[27] E. Ong and R. Bowden, "A boosted classifier tree for hand shape detection," in *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*, pp. 889–894, IEEE, 2004.

[28] M. Van den Bergh, D. Carton, R. De Nijs, N. Mitsou, C. Landsiedel, K. Kuehnlenz, D. Wollherr, L. Van Gool, and M. Buss, "Real-time 3d hand gesture interaction with a robot for understanding directions from humans," in *RO-MAN, 2011 IEEE*, pp. 357–362, IEEE, 2011.

[29] J. Crowley, F. Berard, and J. Coutaz, "Finger tracking as an input device for augmented reality," in *International Workshop on Gesture and Face Recognition, Zurich*, pp. 195–200, Citeseer, 1995.

[30] M. Bencheikh-el hocine, M. Bouzenada, and M. Batouche, "A new method of finger tracking applied to the magic board," in *Industrial Technology, 2004. IEEE ICIT'04. 2004 IEEE International Conference on*, vol. 2, pp. 1046–1051, IEEE, 2004.

[31] Q. Yuan, A. Thangali, V. Ablavsky, and S. Sclaroff, "Learning a family of detectors via multiplicative kernels," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, no. 3, pp. 514–530, 2011.

[32] C. Lee, C. Shih, and B. Jeng, "Fingertip-writing alphanumeric character recognition for vision-based human computer interaction," in *Broadband, Wireless Computing, Communication and Applications (BWCCA), 2010 International Conference on*, pp. 533–537, IEEE, 2010.

[33] J. Letessier and F. Bérard, "Visual tracking of bare fingers for interactive surfaces," in *Proceedings of the 17th annual ACM symposium on User interface software and technology*, pp. 119–122, ACM, 2004.

[34] L. Jin, D. Yang, L. Zhen, and J. Huang, "A novel vision based finger-writing character recognition system," in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, vol. 1, pp. 1104–1107, IEEE, 2006.

[35] H. Zhou and Q. Ruan, "Finger contour tracking based on model," in *TENCON'02. Proceedings. 2002 IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering*, vol. 1, pp. 503–506, IEEE, 2002.

[36] Z. Feng, Y. Zheng, B. Yang, W. Gai, Y. Li, Y. Lin, H. Tang, and X. Song, "Freehand tracking based on behavioral model analysis," in *Virtual Reality and Visualization (ICVRV), 2011 International Conference on*, pp. 103–108, IEEE, 2011.

[37] Z. Wu, M. Betke, J. Wang, V. Athitsos, and S. Sclaroff, "Tracking with dynamic hidden-state shape models," *Computer Vision–ECCV 2008*, pp. 643–656, 2008.

[38] I. Oikonomidis, N. Kyriazis, and A. Argyros, "Efficient model-based 3d tracking of hand articulations using kinect," *Procs. of BMVC, Dundee, UK (August 29–September 10 2011)[547]*, 2011.

[39] C. Jennings, "Robust finger tracking with multiple cameras," in *Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems, 1999. Proceedings. International Workshop on*, pp. 152–160, IEEE, 1999.

[40] L. Yun and Z. Peng, "An automatic hand gesture recognition system based on viola-jones method and svms," in *Computer Science and Engineering, 2009. WCSE'09. Second International Workshop on*, vol. 2, pp. 72–76, IEEE, 2009.

[41] W. Du and H. Li, "Vision based gesture recognition system with single camera," in *Signal Processing Proceedings, 2000. WCCC-ICSP 2000. 5th International Conference on*, vol. 2, pp. 1351–1357, IEEE, 2000.

[42] Y. Okada, "3d model matching based on silhouette image matching," *Proc. of CSCC2002 (Recent Advances in Circuits, Systems and Signal Processing), WSEAS Press*, p. 3804385, 2002.

[43] W. Li, Z. Zhang, and Z. Liu, "Action recognition based on a bag of 3d points," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, pp. 9–14, IEEE, 2010.

[44] W. Li, Z. Zhang, and Z. Liu, "Expandable data-driven graphical modeling of human actions based on salient postures," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 18, no. 11, pp. 1499–1510, 2008.

[45] S. Belongie, J. Malik, and J. Puzicha, "Shape context: A new descriptor for shape matching and object recognition," *Advances in neural information processing systems*, pp. 831–837, 2001.

[46] H. Zhang and X. Zhang, "Shape recognition using a moment algorithm," in *Multimedia Technology (ICMT), 2011 International Conference on*, pp. 3226–3229, IEEE, 2011.

[47] Z. Huang and J. Leng, "Analysis of hu's moment invariants on image scaling and rotation," in *Computer Engineering and Technology (ICCET), 2010 2nd International Conference on*, vol. 7, pp. V7–476, IEEE, 2010.

[48] J. Foster, M. Nixon, and A. Prugel-Bennett, *Gait recognition by moment based descriptors*. 2002.

[49] M. Van den Bergh and L. Van Gool, "Combining rgb and tof cameras for real-time 3d hand gesture interaction," in *Applications of Computer Vision (WACV), 2011 IEEE Workshop on*, pp. 66–72, IEEE, 2011.

[50] E. Keogh, L. Wei, X. Xi, S. Lee, and M. Vlachos, "Lb_keogh supports exact indexing of shapes under rotation invariance with arbitrary representations and distance measures," in *Proceedings of the 32nd International Conference on very large data bases*, pp. 882–893, VLDB Endowment, 2006.

[51] K. Oka, Y. Sato, and H. Koike, "Real-time fingertip tracking and gesture recognition," *Computer Graphics and Applications, IEEE*, vol. 22, no. 6, pp. 64–71, 2002.

[52] M. Shin, D. Goldgof, and K. Bowyer, "Comparison of edge detectors using an object recognition task," in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, vol. 1, IEEE, 1999.

[53] A. Jain, R. Duin, and J. Mao, "Statistical pattern recognition: A review," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 1, pp. 4–37, 2000.

[54] H. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 20, no. 1, pp. 23–38, 1998.

[55] D. Rumelhart, G. Hintont, and R. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[56] L. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains," *The annals of mathematical statistics*, vol. 41, no. 1, pp. 164–171, 1970.

[57] A. Maimone and H. Fuchs, "Reducing interference between multiple structured light depth sensors using motion," 2012.

[58] D. Butler, S. Izadi, O. Hilliges, D. Molyneaux, S. Hodges, and D. Kim, "Shake'n'sense: reducing interference for overlapping structured light depth cameras," in *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, pp. 1933–1936, ACM, 2012.