Automated Gesture Recognitions Using Hidden Markov Model (HMM)

By

Mr. Naveed Shah

AUIC-14-0775

Master of Science in Electrical Engineering (MSEE)

A thesis proposal submitted in partial fulfillment of the requirements for the award of degree of Master of Science in Electrical Engineering at Abasyn University.

Supervisor

Dr. Imran Shafi

****

Department of Computing and Technology

Abasyn University Islamabad Campus

April 2016

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Sr #** | **Content** | **Page no.** |
| I | Introduction | 3 |
| II | Problem Statement | 4 |
| III | Literature Review | 5 |
| IV | Limitations of the Current Research Models | 5 |
| V | Research Objectives | 6 |
| VI | Significance of the Research | 6 |
| VII | Research Design and Methodology   1. Image Acquisition and Preprocessing 2. Hand Detection and Segmentation 3. Extract Trajectory 4. Normalize Trajectory 5. Recognizing using Hidden Markov Model (HMM) 6. Research Flow Chart Overview | 6  7  7  8  8  9  10 |
| VIII | Thesis Timeline | 10 |
| IX | References | 11 |

# **INTRODUCTION**

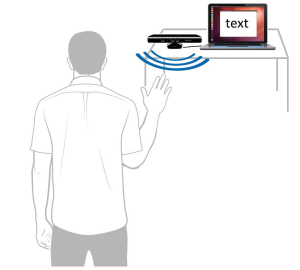
Gestures are used everywhere whether we notice or not but gestures are used frequently by deaf who do not have the ability to communicate verbally like everyone else. Hand gestures, hand shapes and hand movements are used mostly by deaf to convey their messages. We call these gestures as Sign Language. Those who are deaf by birth, they learn to communicate via sign language at a very early age so it’s never an issue for them. The problem arises when ordinary people try to communicate with deaf because ordinary people can communicate verbally but they don’t understand sign language.

The purpose of this project is to overcome the communication barrier between deaf and ordinary people. We can overcome the barrier by introducing a middleware between deaf and ordinary people which will translate sign language to written and verbal form. To capture gestures accurately Microsoft Kinect version 2.0 will be used. The reason for choosing Microsoft Kinect version 2.0 is its accuracy which is achieved by making use of RGB camera as well as infrared camera. Our final deliverable is supposed to capture a gesture and translate it to natural language verbally as well as written according to sign language standards.



**Figure 1. Points Detects by Kinect**

The project uses a matching algorithm in real time and compares the gesture function with a gesture that is already in our database.



**Figure 2. Gesture Recognized by Kinect device**

There are various types of methods based on signal detection mode. Libraries use Kinect and OpenNI to pursue hands tend to focus more on applications, strategies containment and settlement and shows that the next hand OpenNI strategy is suitable for applications tested so far has not yet been tested, but Kinect sensors and other depth barriers for signal detection in applications and test situations.

-----------------------more intro------------

Computers have evolved and spread into every field of industry and entertainment. We use them every day at work, at home, at school, simply almost everywhere and computers, in any form, have become an integral part of our lives. Today, when someone speaks about using a computer, we usually imagine typing on the keyboard and moving the mouse device on the table. These input methods have been invented in 1960s as a kind of artificial control allowing users to use computers with limited computational power. Today the technological advancement is making significant progress in the development of sensing technology and makes it possible to gradually substitute the artificial way of human–computer interaction by more natural interactions called Natural User Interface (NUI).

The NUI has already found its place in mobile devices in the form of multi– touch screens. Selecting items, manipulating with images and multimedia using touch makes the human–computer interaction more natural than it is with the traditional peripheries. However, in the past years the evolution of the sensing technology has gone much further beyond the limits of the currently used human– computer interaction. The technological advancement in computer vision enabled computers to discern and track movements of the human body.

Starting with the Microsoft Kinect for Xbox 360 introduced in November 2010, the new touch–less interaction has unleashed a wave of innovative solutions in the field of entertainment, shopping, advertising, industry or medicine. The new interaction revealed a world of new possibilities so far known only from sci–fi movies like Minority Report.

The goal of this thesis is to design and implement the touch–less interface using the Microsoft Kinect for Windows device and investigate the usability of various approaches in different designs of the touch–less interactions by conducting subjective user tests. Finally, on the basis of the results of the performed user tests the most intuitive and comfortable design of the touch–less interface is integrated with the ICONICS GraphWorX64™ application as a demonstration of using the touch–less interactions with the real application.

# **PROBLEM STATEMENT**

Gesture recognition is important not only for the deaf, but also for different purposes.

Real-time recognition of gestures is a difficult task. Each country has its own sign language and these signs are stored in the database. These signs are converted to word and the word translated into the spoken word. Using Kinect v1 can detect one user at real time for performing the gesture While Kinect V2 can detect 25 points of six people for performing gesture. Here is our main objective understanding of the basic symptoms that can occur in everyday life. This method describes the development and implementation of 3D-robust system, which allows simple control gestures to communicate spatial relationship in a complex environment.

# **LITERATURE REVIEW**

Gesture Recognition Research is going all over the world. Some Languages like American Sign Language [2, 18], Japanese Sign Language [3], and Swedish Sign Language [4] have achieved good level of results using gesture recognition techniques. In countries like India, Pakistan and Afghanistan research on Sign Language began very late because of the absence of advance technology. In acknowledgment stage, distinctive techniques were utilized like Well [9, 10], KNN [7], SVM [5, 6] and DTW [8]. The majority of the scientists were used glove for hand detection. Use the glove over the skin; in this way it is easily to get hand the hand component. Kishore and Kumar [16] dealt with Sign Language using fuzzy logic and achieved about 96% accuracy. They used the extraction of components on the basis of the color.

There exist much research methodology for learning and recognition of visual manners. However because of recent research to the vision group, just a little numbers have been used Hidden Markov Models. We compress a couple of the fascinating works identified with the paper. Kalin and Jonas prepared the framework with Hidden Markov Models. 51 signs were tested and which were achieved 89.7% accuracy [4]. Moni M. A. and Ali have dissected different methodologies in motion acknowledgment for communication via gestures acknowledgment utilizing Hidden Markov Models [13].

# **LIMITATIONS OF THE CURRENT RESEARCH MODELS**

There are many limitations in the current research models. The first and main problem is that they don’t consider all 25 points in of human body. Our way of overcoming the limitation is to consider all 25 parts of human body.

Current research models consume too much processing. The reason for time complexity is their lengthy process, first they read image then they apply processing on every single pixel which increases time complexity.

Current research models consume a lot of memory. These models store images from database into memory during processing which often exceed limit of memory and the system goes out of memory. To overcome this issue we will store path only.

# **RESEACH** **OBJECTIVES**

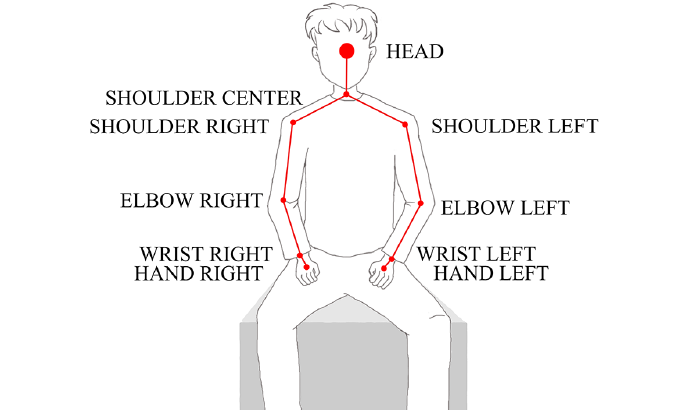
My research objective is to provide ease to people who do not understand Sign Language by providing an intermediate to them which will translate Sign Language to text and Speech. Kinect Sensor can capture 3D gesture which we will be using as input, based on that particular gesture, we will find out what this means by comparing it to stored information about gestures using complex Algorithms. Once we have translated the gesture to text then we will display this text as well as convert it to speech. This is how gestures will be translated to Text and Speech.

# **SIGNIFICANCE OF THE RESEARCH**

The world is a global village and people connected via the Internet and telephone. The use of gesture recognition is important when communicating with deaf people because they cannot communicate verbally. With Kinect to detect gestures in real time we must understand the signs of the deaf person. Gesture recognition module is a fundamental analysis of human activity. To communicate with a deaf need to recognize all parts of the human body, and know your body movements.

# **RESEARCH DESIGN AND METHODOLOGY**

Our system consists of multiples modules like image acquisition, extract hand features and hand gestures recognition using Hidden Markov Models. In image acquisition module; pixels depth, angles of coordinates and skeleton information are captured. In feature extraction; the joint information of coordinate of the skeletons, angle between joints and then data normalization is applied on it. In hand gestures recognition module, maximum likelihood of Hidden Markov Models parameter is derived with the help of Baum Welch algorithms for the given set of sequence.

****

**Figure 3. Kinect Skeleton**

## Setup environment

In our project we will use kinect sensor. The invention of sensors capable of depth sensing in real–time enabled computers to see spatially without the need of complex visual analysis that is required for images captured by regular sensors. This advantage in additional depth information made it easier for computer vision and allowed to create algorithms such as Skeletal Tracking, Face Tracking or Hand Detection. The Skeletal Tracking is able to track body motion and enables the recognition of body language. The Face Tracking extends the body motion sensing by recognition and identification of facial mimics. Lastly, the Hand Detection enables tracking fingers [4] or recognizing hand gestures [5].

The computer’s ability to understand body movements led to the design of a whole new kind of human–computer interaction, which was termed: Touch–less Interface [6]. The touch–less interface indicates that touch interaction and mouse input will not be the only broadly accepted ways that users will engage with interfaces in the future.

The most common design for touch–less interface is using the user’s hands for moving a cursor over the screen. This technique uses the Skeletal Tracking that can be combined with a Hand Detection for performing a click. A usual scenario for use of such a touch–less interface is that the user stands facing the sensor and with his hand in certain distance from his body and high above the floor, he can move a cursor on the screen by his hand’s movement. This kind of NUI is used by Microsoft for Kinect for Xbox 360 dashboard (Figure 2.1) and also the company promotes it for use with the Kinect for Windows targeted for PC. The design, however, requires it to be combined with a traditional GUI for creating the user’s interface and giving

advices which means that this kind of natural interaction is still not a pure NUI but it’s getting closer to it.



Figure 2.1 – An illustration of the Kinect for Xbox 360 touch–less interface.

Another design for a touch–less interface takes advantage of the possibility to track the user’s body movement and translate them to specific gestures. Gestures are something what all people use independently in language and, moreover, in the knowledge in controlling computers. They can use them naturally and learn them very fast. Even though, innate gestures may have different meanings in different parts of the world, computers can learn them and translate them to predefined actions correctly. For example, the most often used gesture is waving, its meaning is very understandable because people use wave for getting attention to them. Analogously, the wave gesture may be used for login to start an interaction with computer. Other common gesture is swipe which usually people use in a meaning of getting something next or previous. The mentioned gestures for wave and swipe are quite simple to recognize but there is an opportunity to teach computers even more difficult ones using, for instance, machine learning algorithms and learn computers to understand a hand write or the Sign Language.

Lately, the computing performance and electronics miniaturization gave birth to even more advanced types of touch–less Interfaces. One of the most interesting projects is certainly Gaze Interaction unveiled on CES 2013 by a company Tobii [9]. The gaze interaction is using an Eye tracking for enabling naturally select item on the screen without any need of using any periphery device or even hands and that all only by looking at the item. Another interesting project is a project Leap Motion [10]. This sensor is based on the depth sensing but it disposes of very high resolution which allows much precise fingers tracking.

**Microsoft Kinect Sensor:** The Kinect sensor has been developed and patented [11] by Microsoft Company originally under a project Natal since 2006. The intention to create a revolutionary game controller for Xbox 360 was initiated by the unveiling of the Wii console at the 2005 Tokyo Game Show conference. The console introduced a new gaming device called the Wii Remote which can detect movement along three axes and contains an optical sensor that detects where it is pointing. This induced the Microsoft’s Xbox division to start on a competitive device which would surpass the Wii. Microsoft created two competing teams to come up with the intended device: one working with a PrimeSense technology and other working with technology developed by a company called 3DV. Eventually, the final product has been named Kinect for Xbox 360 and was built on the PrimeSense’s depth sensing technology.

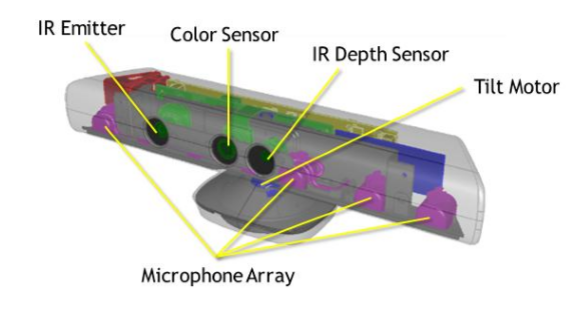
At this time, Microsoft offers two versions of the Kinect device. The first one, Kinect for Xbox 360, is targeted on the entertainment with Xbox 360 console and was launched in November 2010. After the Kinect was hacked and many various applications spread through the Internet, Microsoft noticed the existence of a whole new market. On the basis of this finding Microsoft designed a second version of the sensor, Kinect for Windows, targeted on the development of commercial applications for PC. Technically, there are only slight differences between both versions; however, the official Software Development Kit from Microsoft limits the support of Kinect for Xbox 360 for development only. The most important difference between Kinect for Xbox 360 and Kinect for Windows is especially in an additional support of depth sensing in near range that enables the sensor to see from 40 centimeters distance instead of 80 centimeters.

The Kinect device is primarily based on a depth sensing technology that consists of an Infra–Red (IR) camera and IR emitter positioned in a certain distance between them. The principle of the depth sensing is an emitting of a predefined pattern by the IR emitter and a capturing of its reflected image that is deformed by physical objects using the IR camera. The processor then compares the original pattern and its deformed reflected image and determines a depth on the basis of

7

variations between both patters. The resulting depth image has a horizontal resolution of 1920 pixels, vertical 1080 and depth resolution of 8 meters divided by millimeters.

The device is additionally equipped with the color (RGB) camera with up to 1280 960 pixels resolution, which may be used as another data source for recognition. Other device’s component is a multi–array microphone for spatial voice input with ability to recognize a direction of a voice source. The device’s tilt angle is possible to set using a motor in range from -27 to 27 degrees which increases a final vertical sensor’s field of view. Additionally, the device contains a 3–axis accelerometer primarily used for determining a device’s tilt angle but it can be used for additional further applications. Figure 2.2 describes a layout of the Kinect’s components.



**Software Development Kits:** There are several Software Development Kits (SDK) available for enabling a custom application development for the Kinect device. The first one is a libfreenect library which was created as a result of the hacking effort in 2010, at the time when Microsoft had not published public drivers and held back with providing any development kits for PC. The library includes Kinect drivers and supports a reading of a depth and color stream from the device. It also supports a reading of accelerometer state and interface for controlling motorized tilt.

Another SDK, available before the official one, is OpenNI released in 2010, a month after the launch of Kinect for Xbox 360. The OpenNI library was published by PrimeSense Company, the author of the depth sensing technology used by Kinect. The SDK supports all standard inputs and in addition includes a Skeletal Tracking. Since its release an OpenNI community has grown and developed a number of interesting projects including 3D scanning and reconstruction or 3D fingers tracking.

The Microsoft’s official SDK for Kinect was unveiled in its beta version in July 2011 and its first release was on February 2012 as the Kinect for Windows SDK version 1.0. Currently, there is available the newest version of the SDK, a version 1.7. An evolution and features of the SDK are described in the following chapter.

**Microsoft Kinect for Windows SDK:** Microsoft published an official SDK after it had realized the Kinect’s potential in opening a new market. The first final version of the SDK was officially released in February 2012 as a Kinect for Windows SDK along with unveiling a commercial version of the sensor, Kinect for Windows. The SDK supports a development in C++, C#, VB.NET, and other .NET based languages under the Windows 7 and later operating systems. The latest version of the SDK is available for free on its official website [16].

The Kinect for Windows SDK started by its very first beta version that was released in July 2011. The beta was only a preview version with a temporary Application Programming Interface (API) and allowed users to work with depth and color data and also supported an advanced Skeletal Tracking which, in comparison with an open–source SDKs, did not already require T–pose to initialize skeleton tracking as is needed in other Skeletal Tracking libraries. Since the first beta Microsoft updated the SDK gradually up to version 1.7 and included a number of additional functions.

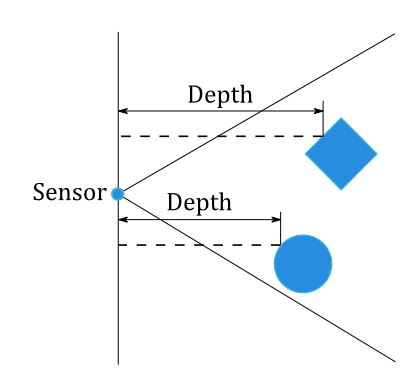
The first major update came along with the 1.5 version that included a Face Tracking library and Kinect Studio, a tool for recording and replaying sequences captured by the sensor. The next version 1.6 extended SDK by the possibility of reading an infrared image captured by the IR camera and finally exposed the API for reading of accelerometer data. The currently latest Kinect for Windows SDK version 1.7 was released in March 2013 and included advanced libraries such as Kinect Fusion, a library for 3D scanning and reconstruction, and a library for hand grip detection which has opened doors for more natural way of interaction.

The API of the Kinect for Windows SDK provides sensor’s depth, color and skeleton data in a form of data streams. Each of these streams can produce actual data frame by polling or by using an event that is raised every time a new frame is available [17]. The following chapters describe particular data streams and their options.

**Data from Microsoft kinect**

In our approach we will get three types of data.

**Depth data:** Data from the Kinect’s depth camera are provided by the depth stream. The depth data are represented as a frame made up of pixels that contain the distance in millimeters from the camera plane to the nearest object as is illustrated by the Figure 2.4.



The pixel merges the distance and player segmentation data. The player segmentation data stores information about a relation to the tracked skeleton that enables to associate the tracked skeleton with the depth information used for its tracking. The depth data are represented as 16–bit unsigned integer value where the first 3 bits are reserved for the player segmentation data and the rest 13 bits for the distance. It means that the maximal distance stored in the depth data can be up to 8 meters.

The depth frame is available in different resolutions. The maximum resolution is 640 480 pixels and there are also available resolutions 512\*424 pixels. Depth frames are captured in 30 frames per seconds for all resolutions.



In the above equation, d as actual distance (unit by met), d(gray )as grayscale of image,

K =0.1236m, @=0.037 m.

The depth camera of the Kinect for Windows sensor can see in two range modes, the default and the near mode. If the range mode is set to default value the sensor captures depth values in range from 0.8 meter to 4.0 meters, otherwise when the range mode is set to near value the sensor captures depth values in range from 0.4 meter to 3.0 meters.

**Color data**: Color data available in different resolutions and formats are provided through the color stream. The color image’s format determines whether color data are encoded as RGB, YUV or Bayer.

The RGB format represents the color image as 32–bit, linear X8R8G8B8– formatted color bitmap. A color image in RGB format is updated at up to 30 frames per seconds at 640 480 resolution and at 12 frames per second in high–definition 1280 960 resolution. [19]

The YUV format represents the color image as 16–bit, gamma–corrected linear UYVY–formatted color bitmap, where the gamma correction in YUV space is equivalent to standard RGB gamma in RGB space. According to the 16–bit pixel representation, the YUV format uses less memory to hold bitmap data and allocates less buffer memory. The color image in YUV format is available only at the 640 480 resolution and only at 15 fps. [19]

The Bayer format includes more green pixels values than blue or red and that makes it closer to the physiology of human eye [20]. The format represents the color image as 32–bit, linear X8R8G8B8–formatted color bitmap in standard RGB color space. Color image in Bayer format is updated at 30 frames per seconds at 640 480 resolution and at 12 frames per second in high–definition 1280 960 resolution. [19]

Since the SDK version 1.6, custom camera settings that allow optimizing the color camera for actual environmental conditions have been available. These settings can help in scenarios with low light or a brightly lit scene and allow adjusting hue, brightness or contrast in order to improve visual clarity.

Additionally, the color stream can be used as an Infrared stream by setting the color image format to the Infrared format. It allows reading the Kinect’s IR camera’s image. The primary use for the IR stream is to improve external camera calibration using a test pattern observed from both the RGB and IR camera to more accurately determine how to map coordinates from one camera to another. Also, the IR data can be used for capturing an IR image in darkness with a provided IR light source.

**Skeletal Tracking:** The crucial functionality provided by the Kinect for Windows SDK is the Skeletal Tracking. The skeletal tracking allows the Kinect to recognize people and follow their actions [21]. It can recognize up to six users in the field of view of the sensor, and of these, up to two users can be tracked as the skeleton consisted of 20 joints that represent locations of the key parts of the user’s body (Figure 2.7). The joints locations are actually coordinates relative to the sensor and values of X, Y, Z coordinates are in meters. The Figure 2.6 illustrates the skeleton space.

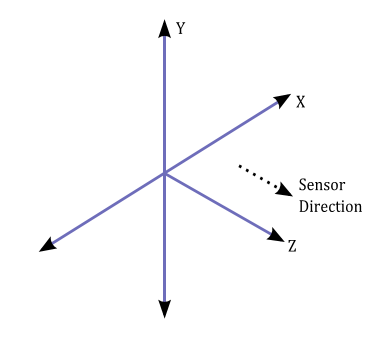


Figure 2.6 – An illustration of the skeleton space.

The tracking algorithm is designed to recognize users facing the sensor and in the standing or sitting pose. The tracking sideways poses is challenging as part of the user is not visible for the sensor. The users are recognized when they are in front of the sensor and their head and upper body is visible for the sensor. No specific pose or calibration action needs to be taken for a user to be tracked.

The skeletal tracking can be used in both range modes of the depth camera, see also 2.3.1. By using the default range mode, users are tracked in the distance between 0.8 and 4.0 meters away, but a practical range is between 1.2 to 3.5 meters due to a limited field of view. In case of near range mode, the user can be tracked between 0.4 and 3.0 meters away, but it has a practical range of 0.8 to 2.5 meters.

The tracking algorithm provides two modes of tracking [22]. The default mode is designed for tracking all twenty skeletal joints of the user in a standing pose. The seated mode is intended for tracking the user in a seated pose. The seated mode tracks only ten joints of upper body. Each of these modes uses different pipeline

13

for the tracking. The default mode detects the user based on the distance of the subject from the background. The seated mode uses movement to detect the user and distinguish him or her from the background, such as a couch or a chair. The seated mode uses more resources than the default mode and yields a lower throughput on the same scene. However, the seated mode provides the best way to recognize a skeleton when the depth camera is in near range mode. In practice, only one tracking mode can be used at a time so it is not possible to track one user in seated mode and the other one in default mode using one sensor.

The skeletal tracking joint information may be distorted due to noise and inaccuracies caused by physical limitations of the sensor. To minimize jittering and stabilize the joint positions over time, the skeletal tracking can be adjusted across different frames by setting the Smoothing Parameters. The skeletal tracking uses the smoothing filter based on the Holt Double Exponential Smoothing method used for statistical analysis of economic data. The filter provides smoothing with less latency than other smoothing filter algorithms [23]. Parameters and their effect on the tracking behavior are described in [24].

## Image/video Acquisition, video stream

Here the image is captured through Microsoft Kinect. Kinect camera is used to capture hand sign at rate of 30 frames per second. The dimension of each frame is 640 \* 480.

**Depth Stream:** A depth image is represented and implemented by the DepthFrame class. The class contains information about a depth image’s format, image’s dimensions, time of its capture and above all the depth pixels data and user index data. Depth data contains data of the depth image that are represented as an array of 16–bit signed integer values where each value corresponds to a distance in physical space measured in millimeters. An invalid depth value -1 means that the pixel is a part of a shadow or it is invisible for the sensor. User index data are represented as a byte array of the same length as the depth data array. The user index data contain information about which pixel is related to which tracked user.

The DepthFrame class provides an interface for a basic manipulation with depth and user index data such as getting and setting a depth or user index value at given and coordinates, flipping and cropping depth image. The class also provides a method Clone() for creation of its copy.

A class diagram describing a DepthFrame object representation is illustrated by the Figure 3.18.(change the diagrame)

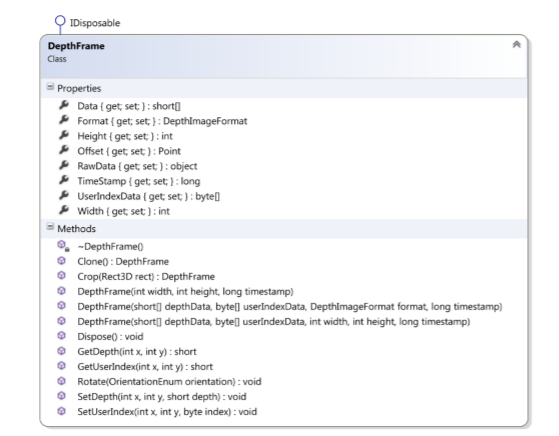


Figure 3.18 – A class diagram describing the depth frame data structure.

**Color stream:** A color image is represented and implemented by the ColorFrame class. The class contains information about a color image format, image dimensions and image data. The image data are represented as a byte array. The color image is stored in ARGB format, it means, the image pixels are stored as a sequence of four bytes in order blue, green, red and alpha channel.

The ColorFrame class provides an interface for a basic manipulation with pixel data such as getting and setting pixel color at given and coordinates and flipping the image. The class also provides a method Clone() for creation of its copy.

A class diagram describing a ColorFrame object representation is illustrated by the Figure 3.19.

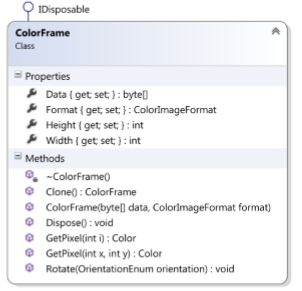


Figure 3.19 – A class diagram describing the color frame data structure.

**Skeleton stream:** Information about all tracked skeletons is represented and implemented by the SkeletonFrame class. This class contains an array of currently tracked skeletons. The class provides a method for getting a skeleton by a given id and also provides a method Clone() for creation of its deep copy.

A tracked skeleton is represented and implemented by the Skeleton class. The skeleton is identified by its ID stored in the property Id. An association of the skeleton to the user’s information in the depth image is realized by the property UserIndex which identifies depth pixels related to the tracked skeleton. The skeleton data are composed of 20 types of joints representing user’s body parts of interest. All of these 20 tracked joints are stored in the skeleton’s collection Joints. In addition, the Skeleton class provides a property Position containing a position of the tracked user blob [33] in physical space. The property TrackingState contains information about a state of skeleton’s tracking. If the skeleton is tracked, the state is set to a value Tracked, when the skeleton is not tracked but the user’s blob position is available, the state has a value PositionOnly, otherwise the skeleton is not tracked at all and the state is set to a value NotTracked. In case of the user’s blob is partially out of the sensor’s field of view and it’s clipped the property ClippedEdges indicates from which side the tracked user blob is clipped.

The joint is represented and implemented by the SkeletonJoint class. The class contains a position of the joint in physical space. A tracking state of the joint is stored in property TrackingState. If the joint is tracked the state is set to a value Tracked. When the joint is overlaid by another joint or its position is not possible to determine exactly the tracking state has a value Inferred, although, the position is tracked it could be inaccurate. Otherwise, when the joint is not tracked its tracking state is set to a value NotTracked.

A class diagram describing a SkeletonFrame object representation is illustrated by the Figure 3.20.

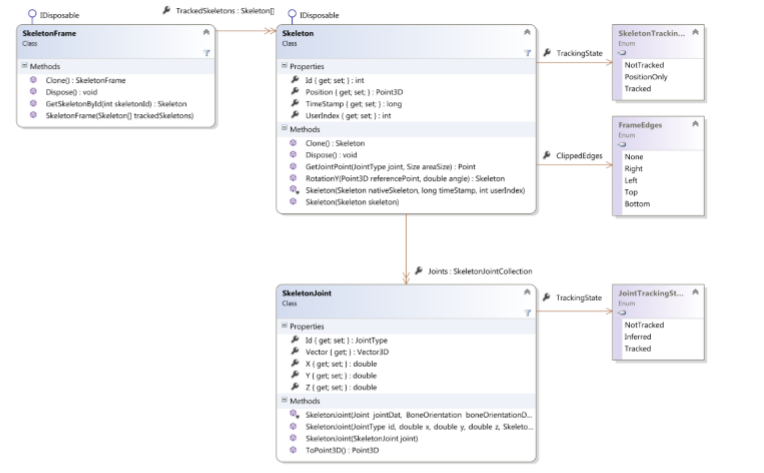


Figure 3.20 – A class diagram describing architecture of the skeleton frame data structure.

## Data Sources

Logic for processing of obtained data from the sensor is implemented by data sources. There are four types of data sources: DepthSource, ColorSource, SkeletonSource and FaceSource which are additionally composed into the KinectSource which handles the logic for obtaining data from the sensor. Each data source implements logic for handling a given data input and processes obtained data into the corresponding data structure. When data processing is finished the data source forwards the result data structure through its event–based interface for the further data processing.

**Depth Source:**

A depth image obtained from the sensor is processed into the DepthFrame data structure using logic implemented by the KinectDepthSource class. The processing is handled by the method ProcessDepthImage() that passes a native depth image represented by the Microsoft.Kinect.DepthImage-Frame structure as its parameter. Depth pixel data are copied into the internal buffer and then each pixel is decomposed into the depth and user index component, see also chapter 2.3.1 for the depth pixel’s format description. When the depth pixel data processing is done a new instance of the DepthFrame data structure is created on the processed data and it is passed on by raising an event DepthFrameReady.

The KinectDepthSource class also provides properties that describe physical parameters of the depth sensor such as a value of the minimal or maximal depth which the sensor is able to capture and a value of the nominal horizontal, vertical and diagonal field of view in degrees. The class also provides a property for selecting between default and near range mode of the depth sensor, see also chapter 2.3.1.

Before the depth image data processing can be started the depth source has to be enabled. It can be done by setting the Enabled property to true which initializes the sensor’s depth data stream. In default the depth source is disabled so the first step before performing a depth image data processing is its initialization by setting the Enabled property to true.

A class diagram describing a KinectDepthSource object representation is illustrated by the Figure 3.23.

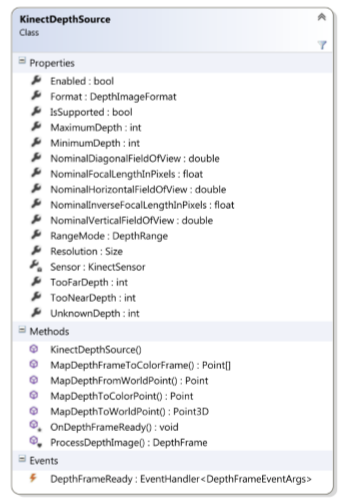


Figure 3.23 – A class diagram describing an object model of the depth data source.

**Color Source**: A color image obtained from the sensor is processed into the KinectColorFrame data structure using logic implemented by the ColorSource class. The processing is handled by the method ProcessColorImage() that passes a native color image represented by the Microsoft.Kinect.ColorImageFrame structure as its parameter. Color pixel data are copied

Figure 3.23 – A class diagram describing an object model of the depth data source.

41

into the internal buffer which is then used for creating of a new ColorFrame instance. Finally, the new instance of the ColorFrame is passed on by raising an event ColorFrameReady.

Before the color image data processing can be started the color source has to be enabled. It can be done by setting the Enabled property to true which initializes the sensor’s RGB camera data stream. In default the color source is disabled so the first step before performing a color image data processing is its initialization by setting the Enabled property to true.

A class diagram describing a KinectColorSource object representation is illustrated by the Figure 3.24

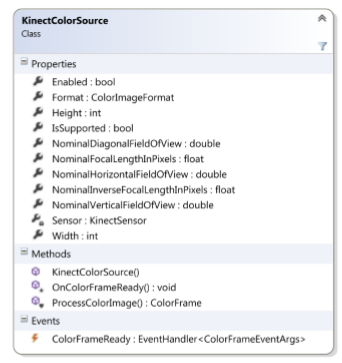


Figure 3.24 – A class diagram describing an object model of the color data source.

**Skeleton Source**: Logic for a processing of skeleton data obtained from the Kinect’s skeleton data stream is implemented by the KinectSkeletonSource class. The processing is handled by the method ProcessSkeletonData() that passes a native skeleton frame represented by the Microsoft.Kinect.SkeletonFrame structure as its parameter. Skeletons data are copied into the internal buffer. The processing algorithm goes through all skeletons and finds those which are in tracked state. The tracked skeleton’s data are used for creating of a new instance of Skeleton class and the new instance is inserted into the list of tracked skeletons. After all skeletons are processed, a new instance of the SkeletonFrame class is created on the basis of the list of tracked skeletons.

Figure 3.24 – A class diagram describing an object model of the color data source.

Figure 3.25 – A class describing an object model of the skeleton data source.

42

Before the skeleton data processing can be started the skeleton source has to be enabled. It can be done by setting the Enabled property to true which initializes the sensor’s skeleton data stream. In default the skeleton source is disabled so the first step before performing a skeleton data processing is its initialization by setting the Enabled property to true.

A class diagram describing a KinectSkeletalSource object representation is illustrated by the Figure 3.25

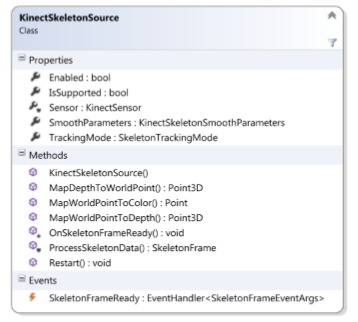


Figure 3.25 – A class describing an object model of the skeleton data source.

## Body position

Find the body position, kinect device detect the body which is in the range of .5meter to 5 meter. The vertical angle can be moved within range from -27 to +27 degrees up and down by using the sensor tilt. Additionally, the depth camera is limited in its view distance. It can see within range from 0.4 meter to 8 meters but for the practical use there are recommended values within 1.2 meter to 3.5 meters.

## Hand state

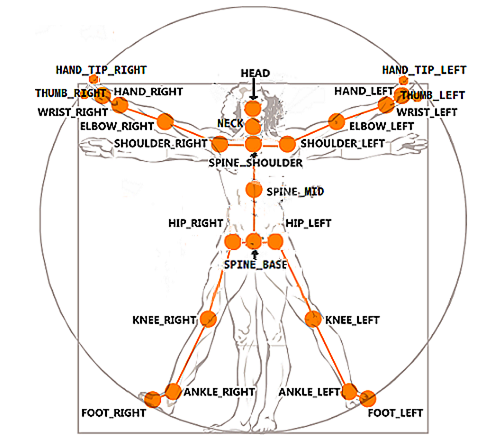
Hand is our main point of interest. We will stick to hand state.as the hand is moving from point to point we will see if it state is changing or not. Check if hand is open or closed. if hand is closed or unknown then we will not consider it as a gesture. We will consider only open hand for the gesture sign.

## Get the body

Detect the human body which is in front of camera. kinect camera can detect 25 points of six people at one time. Here it is insure that only person will be standing in front of camera.

**Find the joints:** There are 25 joints mapped for each body, each one has a position and orientation. This exercise creates bones from those joints by storing each joint as a pair and the bone is simply a starting and ending point. Then these bones and joints are drawn to a canvas using simple XAML ellipses and lines. This is the first exercise which does not result in rendering a bitmap and it should help you understand that a good Kinect experience does not have to show the live color feed. An application can use shapes and assets to express the same data.

|  |  |  |
| --- | --- | --- |
| **AnkleLeft** | 14 | Left ankle |
| **AnkleRight** | 18 | Right ankle |
| **ElbowLeft** | 5 | Left elbow |
| **ElbowRight** | 9 | Right elbow |
| **FootLeft** | 15 | Left foot |
| **FootRight** | 19 | Right foot |
| **HandLeft** | 7 | Left hand |
| **HandRight** | 11 | Right hand |
| **HandTipLeft** | 21 | Tip of the left hand |
| **HandTipRight** | 23 | Tip of the right hand |
| **Head** | 3 | Head |
| **HipLeft** | 12 | Left hip |
| **HipRight** | 16 | Right hip |
| **KneeLeft** | 13 | Left knee |
| **KneeRight** | 17 | Right knee |
| **Neck** | 2 | Neck |
| **ShoulderLeft** | 4 | Left shoulder |
| **ShoulderRight** | 8 | Right shoulder |
| **SpineBase** | 0 | Base of the spine |
| **SpineMid** | 1 | Middle of the spine |
| **SpineShoulder** | 20 | Spine at the shoulder |
| **ThumbLeft** | 22 | Left thumb |
| **ThumbRight** | 24 | Right thumb |
| **WristLeft** | 6 | Left wrist |
| **WristRight** | 10 | Right wrist |



Out of 25 joints we are only consider the 4 joints, shoulder left, elbow left, hand left, shoulder right, elbow right and hand right.

## Find elbow angle for position

Find the elbow Angle, we are consider our coordinate as a 3imensional coordinates, so we will find the angle between the the two joints. To find the angle between left elbow first find the distance between left shoulder and left elbow and then calculate the vector norm from these distance like similarly find the distance between left elbow and left wrist and then calculate the vector norm for the distance.

If the hand angle is between 100 and 80 degree,start processing otherwise the hand is not positioned for gesture.

## Extract Trajectory

Extract 3D path of the hand from the given frame; take every tenth frame of the hand because the hand motion is very slow. When the hand moves from one point to another point the speed of the hand also varies with time, so just take the 3D road. The coordinates (x, y, z) is the distance of the point along (x, y, z) axis from the kinect camera.

**3d trajectory**: We are taking three dimensions data. In three dimension trajectory we have three points(x,y,z). Get a single point, two points or three points for further processing. We will discuss all these point one by one and will take the one which will give us the good result.

1.single Point:

**X dimension**: in this case we just take the one dimension(x dimension) value of the trajectory and eliminate the other two dimensions(y,z dimensions) points. Taking only x dimension give us the good result but there are some scenario in which y and z dimensions or only one of these dimension is changing with time and x dimension is constant or only variant on some points. Suppose the word “right” here the x dimension is changing with time and y, z dimension is constant or changing after some points, so in this case it give us the good result. Now let take the word “umbrella” and “necktie”, here in both words only y and z dimension is changing with time and x dimension is little variant, so in this case we can consider only the x dimension. Taking the only the x dimension will not give us the good result. So we will consider x dimension in only those scenario where y,z dimension is constant or slightly variant and x dimension is changing with time.

**Y point**: in this case we just take the one dimension(y dimension) value of the trajectory and eliminate the other two dimensions(x,z dimensions) points. Taking only y dimension give us the good result but there are some scenario in which x and z dimensions or only one of these dimension is changing with time and y dimension is constant or only variant on some points. Suppose the word “right” here the y dimension is changing with time and x, z dimension is constant or changing after some points, so in this case it give us the good result. Now let take the word “umbrella” and “necktie”, here in both words only x and z dimension is changing with time and y dimension is little variant, so in this case we can consider only the y dimension. Taking the only the y dimension will not give us the good result. So we will consider y dimension in only those scenario where x,z dimension is constant or slightly variant and y dimension is changing with time.

**Z point**: in this case we just take the one dimension(z dimension) value of the trajectory and eliminate the other two dimensions(x,y dimensions) points. Taking only z dimension give us the good result but there are some scenario in which x and y dimensions or only one of these dimension is changing with time and z dimension is constant or only variant on some points. Suppose the word “right” here the z dimension is changing with time and x, y dimension is constant or changing after some points, so in this case it give us the good result. Now let take the word “umbrella” and “necktie”, here in both words only x and y dimension is changing with time and z dimension is little variant, so in this case we can consider only the z dimension. Taking the only the y dimension will not give us the good result. So we will consider y dimension in only those scenario where x,y dimension is constant or slightly variant and z dimension is changing with time.

2.Two Points:

**XY points**: in this case we will take the combinations of two dimensions and eliminate one dimension. Consider the combination of x,y dimensions. Split the three dimension array into x,y and z dimension. Take x,y dimensions to make a sequence, this sequence will be used for further training. Suppose if we have the word “right” in this case only the x,y dimensions is changing with time and z dimension is constant or changing after some points. There are also some words like “left” in which x,z dimensions or y,z dimensions or even only z dimension is changing with time and others dimensions are constant or slightly changing. So this scenario is not good for such type of words in which the other dimension is changing which we are not considering and the one or both dimensions we are considering are changing with time.

**YZ points**: in this case we will take the combinations of two dimensions and eliminate one dimension. Consider the combination of y,z dimensions. Split the three dimension array into y,z and z dimension. Take y,z dimensions to make a sequence, this sequence will be used for further training. Suppose if we have the word “right” in this case only the y,z dimensions is changing with time and z dimension is constant or changing after some points. There are also some words like “left” in which y,z dimensions or y,z dimensions or even only z dimension is changing with time and others dimensions are constant or slightly changing. So this scenario is not good for such type of words in which the other dimension is changing which we are not considering and the one or both dimensions we are considering are changing with time.

**XZ points**: in this case we will take the combinations of two dimensions and eliminate one dimension. Consider the combination of x,z dimensions. Split the three dimension array into x,z and z dimension. Take x,z dimensions to make a sequence, this sequence will be used for further training. Suppose if we have the word “right” in this case only the x,z dimensions is changing with time and z dimension is constant or changing after some points. There are also some words like “left” in which x,z dimensions or x,z dimensions or even only z dimension is changing with time and others dimensions are constant or slightly changing. So this scenario is not good for such type of words in which the other dimension is changing which we are not considering and the one or both dimensions we are considering are changing with time.

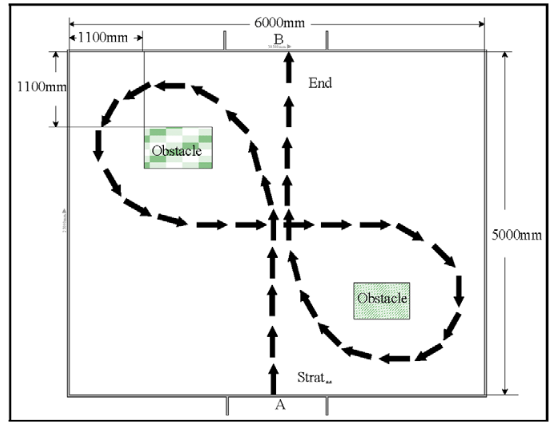
3. three points

**XYZ points**: in this case we will take the combinations of three dimensions .in this scenario we are considering the combination of all the three points(x,y,z) data. Take these points(x,y,z) to make a sequence, this sequence will be used for further training. Taking three dimensions is good for all type f result but there are some words in which only two dimensions or even one dimension is changing and the others are remain constant or slightly change with time.so in the case in which we don’t need the three dimension data. The time and data consuming to use all the three dimensions.

**Dimension to choose:** Now the points come in which we will decide which dimension we will be choosing. From the above discussion we conclude that taking the entire three dimensions will give us the good result in all scenarios.

## Normalize Trajectory

Furthermore, 150 frames are sufficient for expediently acting any gesture, on average: 30 frames form one part of the gesture. Considering the training set, the hand varies very slowly so take every tenth frame. The combination of 20 states and 15 surveillance symbols formed a Hidden Markov Model.



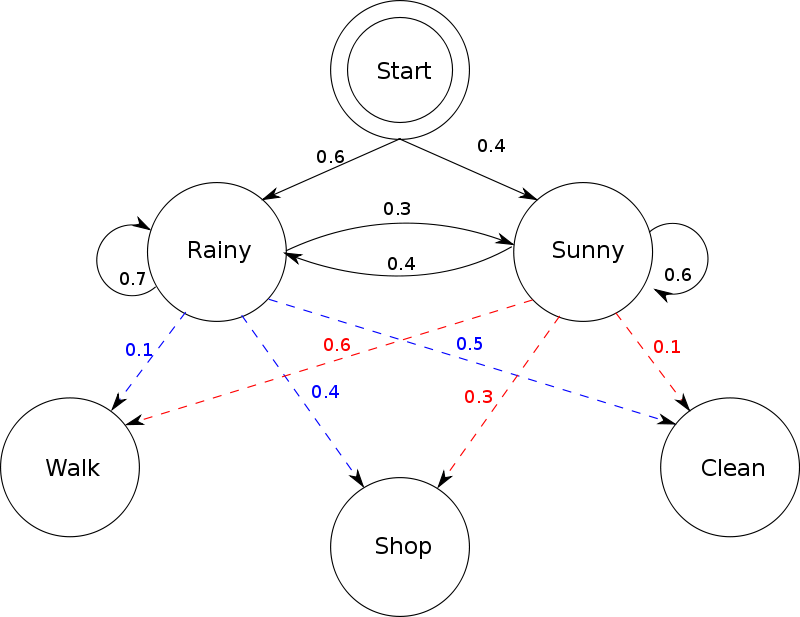
**Figure 5. Trajectory Path**

## Recognizing using Hidden Markov Model (HMM)

The normalized feature of each trajectory form an observation signs for HMM stage. HMM is a best choice for sequential data.

Suppose we have two states Rainy and Sunny, three observations Walk, shop and clean.

Start probability for rainy is 0.6 and sunny is 0.4. The transition probability represents the change. In our example, we have only a 60% chance that tomorrow will be rain if today is sunny. The emission probability represents the activity on each day. If it is sunny, there is a 50% chance that he will shop; if it is rainy, there is a 40% chance that the person will clean his apartment.



**Figure 6. Hidden Markov Model**

**Literature overview**

**(the paper which I following) :** LITERATURE SURVEY Research is going on across the world in this area. Research in computerization of American Sign Language [2, 18], Japanese SL [3], and Swedish SL [4] have reached substantial level. In India, research on ISL interpretation started late due to lack of standardization and very less work is going on at present to convert ISL into sound language. In recognition phase, different methods were used like SVM [5, 6], KNN [7], HMM [9, 10] and DTW [8]. Most of the researchers were using data glove for continuous SLR. Typical phases include: segmentation of the hands based on the colour of gloves or the skin, then extraction of the features mainly the location, orientation and velocity of motion. In India, Kishore and Kumar [16] worked on ISL word recognition using fuzzy logic and achieved 96% accuracy. They used the extraction of features based on the colour. There exist many reported research projects related to learning and recognizing visual behavior. However due to its recent introduction to the vision community, only a small number have been reported which use Hidden Markov Models. HMM has been traditionally used as tool for speech recognition tool. Recent researches have begun relating the speech variations to visual gestures. We summarize a few of the interesting works related to the paper. Kalin and Jonas trained the system with HMM model. 51 signs were analyzed which achieved 89.7% average recognition accuracy [4].Moni M. A. and Ali have analyzed various techniques and approaches in gesture recognition for sign language recognition using HMM [13]. They have provided an overview of HMM and its use in vision based applications, working in two stages that of image capturing and processing using cameras and the second stage for identifying and learning models has eliminated the need of previously used sensor embedded equipment such as gloves for tracking of a gesture. T. E. Starner has employed HMM in 1995 in identifying the American Sign Language. On similar grounds authors Gaus Y. et al. have successfully recognized the Malaysian Sign Language [12]. It consists of skin segmentation procedure throughout frames and feature extraction by centroids, hand distances and orientation has been used, gesture paths define the hand trajectory. Kalaman filters have been used by researchers to identify overlapping hand-head and hand-hand regions. Elmezain et al. have quantized features form spatio-temporal trajectories into code words [9]. They have used a novel method of tracking the gesture by using 3D depth map along with colour information, this helps at separating the same colour at different surfaces in a complex background. In order to separate continuous gestures a special zero codeword is defined, using the start and end points of meaningful gestures the viterbi algorithm is employed. In [10] the authors have used the LRB topology along with forward algorithm to achieve the best performance. With recognition rate of 95.87% Arabic numbers have been identified. Shrivastav R [11] has used OpenCV image processing library to perform the isolation of gesture frames, the entire process form per-processing to testing. In coordination with this processing, Baum-Welch algorithm and LRB topology with forward algorithm is applied for recognition. With the advent of sensor based camera Kinect, due toits benefits over the traditionally used vision based cameras, research orientation was shifted towards use of Kinect [17]. Kalin and Jonas [4] have developed educational signing game based on isolated sign recognition of Swedish sign language using Microsoft Kinect. Frank and Sandy [18] have used Kinect for interpretation of American Sign Language for 10 different isolated words. Recognition accuracy of 97% was achieved using support vector machine. Yanhua et al. [3] presented recognition system for Japanese sign language using Microsoft Kinect sensor. A method was developed to employ two Kinects for getting more dataset of hand signs for which point cloud library (PCL) was used to get processed data. Zang et al. [6] have used improved SURF algorithm and SVM to recognize static sign using Kinect. Most of these techniques required large number of training samples and are mostly dependent on the signer. In reality, signer independent method is more practical and desirable.

**(Other papers literature)** There is large body of information about hand detection. In recent years, there are many method has been developed for hand detection using depth by different devices such as Kinect and Zcam .some of researcher worked on hand segmentation, hand counter, color distribution, etc. Kd-tree structure is used (Suau, Ruiz-Hidalgo, & Casas, 2012) to obtain color candidate cluster.Blue and yellow cluster can be merged since Hausdorff distance between them is small enough. (Park, Hasan, Kim, & Chae, 2012) proposed adaptive hand detection approach by using 3-dimentional information from Kinect and tracks the hand using GHT based method. 3D hand model with label vertices was proposed by (Yao & Fu, 2012). During the training stage both RGB and depth data used as a input variable and produced two classifier such as random forest and Bayesian classifier. Per-78pixel hand part classification was obtained by forest classifier. (Chen, Lin, & Li, 2012) used hand region growing techniques which includes 2 steps. In the first stage, hand position detection was obtained by using hand moving with a velocity. In second stage tried to segment entire hand region by using region growing technique on 3D point. Hybrid RGB and TOF were proposed (Ren, Meng, Yuan, & Zhang, 2011) and (Van den Bergh & Van Gool, 2011). They used skin color segmentation based on two methods. a) Gaussian mixture model (GMM), was trained offline and it was able to detect skin color under lighting condition. b) Histogram-based method which was trained online. Hybrid method can be obtained by multiplying the GMM-based skin color probability with histogram based skin color probability. Representing hand in box used by (Frati & Prattichizzo, 2011). They used threshold approach to compute bounding box. Depth data which is too close and too far from Kinect are considered as zero with fixing lower and upper thresholds. The function “cvFindContours ()” was used to find the counter for the object. Combine both a training stage and estimation stage was used by (Yao & Fu, 2012). RGB and depth images used as an input in training stage and generate random forest classifier and a Bayesian classifier. The former is for the per-pixel hand parts classification; and the latter is employed as an object classifier to locate hand. Threshold Method is a simple method of depth which was used to isolate the hand by (Mo & Neumann, 2006) and (Breuer, Eckes, & Müller, 2007) and (Liu & Fujimura, 2004 and (Biswas & Basu, 2011). Depth thresholding determines the hands to be used to those points between some near and far distance thresholds around the Z (depth) value of the expected centroid of the hand – which can be either predetermined and instructed to the user, or determined as the nearest point in the scene. (Hamester, Jirak, & Wermter, 2013) perform foreground segmentation on depth images to reduce the region of interest. After that, edge detection in the foreground depth image provides a set of candidate contours. Randomized Decision applied this method for accurate hand detection and pose estimation with great accuracy result. (Keskin, Kıraç, Kara, & Akarun, 2012) introduced novel method to tackle the complexity problem. The idea is to reduce the complexity of the model by dividing the training set into smaller clusters, and to train PCFs on each of these compact sets. Most of researcher tries to detect hand and head by using Kinect skeleton which can detect and track hand and head easily. (Xiao, Mengyin, Yi, & Ningyi, 2012) and (Zainordin, Lee, Sani, Wong, & Chan, 2012) used skeleton model for hand detection. They usually crop hand based on coordinate x, y which obtains from the skeleton. However, when the subject's hand is far from the sensor, the captured hand is small in the captured image.

**(Other overview ):** Numerous papers have been written on the subject of generalized hand gesture recognition using RGB-D sensors. One technique considered early on was proposed by Du [3]. The idea is to trace the contours of a hand silhouette and detect concavities and convexities. The convexity points correspond to the finger tips, while the concavity points define the region between fingers. A simple classifier is defined for the digits 0 to 5 by counting the number of these points. However, this descriptor and classification scheme are already applied to Kinect sensor information for gesture recognition. So, instead, the Malima [4] paper is considered. As with the Du method, the scheme proposed in Malima has simple, yet robust feature selection and classification. In addition, it is scale, translation, and rotation invariant. However, their method is applied to images captured with an RGB camera, where segmentation is performed by thresholding the color channels based on skin color values. In principle, though, the circle-based descriptor is applicable to depth images as well. This is one of the

2 Cliff Chan, Seyed Sepehr Mirfakhraei

descriptors implemented as part of the project. It is described in greater detail in the following section. Unfortunately, although simple and relatively robust, the Du and Malima systems are limited to the counting digits, 0 to 5. They are both analyzing local contour features of the hand outline, but do not recognize the shape of the hand as a whole. So, more generalized gesture methods were explored. One such paper is the covariance matrix approach in Guo [7]. This approach performs human action recognition by looking at a sequence of whole body silhouettes over time, a silhouette tunnel, captured by the Kinect. A 13 dimensional feature vector is defined, where 3 values are row, column, and time, 8 values are based on the shape of the silhouette (described later), and the last 2 are a measure of the temporal similarity. However, this feature vector is general enough to work with hand silhouettes in addition to full bodies. So, it is possible to modify the shape feature vector to work with static hand gestures by reducing the dimensionality. This can be accomplished by removing the time dependence and temporal similarity terms. The result is a generalized 10 dimensional feature vector applicable to static shape recognition. The specific implementation and classification details are defined in the next section. Fourier descriptors were also researched. A nice survey of Fourier based shape representations is provided in [6]. Out of all the methods surveyed, the centroid (central) distance Fourier descriptor provides the best results. This method is then used by Kulshreshth [5] to perform gesture recognition with the Kinect. Surprisingly, however, they limit themselves to just recognizing digits as in finger counting. It is implied that the comparatively low resolution of the depth images from the Kinect is the limiting factor. In addition, there is no testing done on the optimal number of contour points to sample. So, we chose to explore the central distance Fourier descriptor further(Jungong Han; Ling Shao; Dong Xu; Shotton, J., "Enhanced Computer Vision With Microsoft Kinect Sensor: A Review," Cybernetics, IEEE Transactions on , vol.43, no.5, pp.1318,1334, Oct. 2013 [2] Suarez, J.; Murphy, R.R., "Hand gesture recognition with depth images: A review," RO-MAN, 2012 IEEE , vol., no., pp.411,417, 9-13 Sept. 2012 [3] H. Du and T. To, "Hand Gesture Recognition Using Kinect," Boston University, 2011. [4] Malima, A.; Ozgur, E.; Cetin, M., "A Fast Algorithm for Vision-Based Hand Gesture Recognition for Robot Control," Signal Processing and Communications Applications, 2006 IEEE 14th , vol., no., pp.1,4, 17-19 April 2006 [5] Kulshreshth, A.; Zorn, C.; LaViola, J.J., "Poster: Real-time markerless Kinect based finger tracking and hand gesture recognition for HCI," 3D User Interfaces (3DUI), 2013 IEEE Symposium on , vol., no., pp.187,188, 16-17 March 2013 [6] D. Zhang and G. Lu, “A comparative study of Fourier descriptors for shape representation and retrieval,” in Proc. 5th Asian Conference on Computer Vision, 2002. [7] Kai Guo; Ishwar, P.; Konrad, J., "Action Recognition in Video by Covariance Matching of Silhouette Tunnels," Computer Graphics and Image Processing (SIBGRAPI), 2009 XXII Brazilian Symposium on , vol., no., pp.299,306, 11-15 Oct. 2009 [8] A Threshold Selection Method from Gray-Level Histograms," Systems, Man and Cybernetics, IEEE Transactions on , vol.9, no.1, pp.62,66, Jan. 1979 [9] Rafael C. Gonzalez and Richard E. Woods. 2006. Digital Image Processing (3rd Edition). Prentice-Hall, Inc., Upper Saddle River, NJ, USA.)

# **REFERENCES**

1. A. Ghotkar, G. Kharate 2015. Dynamic Hand gesture recognition for sign words and Novel Sentence Interpretation Algorithm for Indian Sign Languagne using Microsoft Kinect Sensor. Journal of Pattern Recognition Research, 10(1), pp.25-38.
2. G. Kharate and A. Ghotkar 2016. Vision based Multifeature Hand gesture Recognition for Indian Sign Language Manual Signs. International Journal on Smart Sensing and Intilligent System,9(1),pp. 124-147.
3. C. Vogler, D. Metaxas 2001. A Framework for Recognizing the Simultaneous Aspects of American SignLanguage. Computer Vision and Image Understanding pp.358-384.
4. Y. Sun,N. Kuwahara and K. Morimoto 2013. Analysis of recognition system of Japanese sign language using 3D image sensor. IASDR pp.1-7.
5. K. Stefanovand J.Beskow 2013. A kinect corpus of Swedish sign language Signs. Proceedings Work-shop on Multimodal Corporation pp.1-5.
6. A. Kuznetsova, L. Leal-Taixe, and B. Rosenhahn 2013. Real-time sign language recognition usingconsumer depth camera pp.83-90.
7. Z.Hu, L.Yang, L. Luo, Y. Zhang, and X.Zhou 2014.The Research and Application of SURF AlgorithmBased on Feature Point Selection Algorithm. Sensor and Transducers IFSA publishing pp.67-72.
8. T. Shanablehand K. Assaleh 2007. Arabic sign language recognition in user-independent mode. InProc. int. Conf. Intell. Adv. Syst pp.597-600.
9. J. Lichenauer, E. Hendriks and M. Reinders 2008 .Sign Language Recognition by Combining Statistical DTW and Independent Classification IEEE Transaction on Pattern Analysis and Machine Intelligence 30(11) pp. 2040-2046.
10. Elmezain, M. , Al-Hamadi, A. , Michaelis B 2009. Hand trajectory based gesture spotting and recognition using HMM 16th IEEE International Conference on Image Processing (ICIP)
11. Elmezain, M. , Al-Hamadi, A. , Michaelis, B 2008. A Hidden Markov Model-based continuous gesture recognition system for hand motion trajectory Pattern Recognition19th International Conference , ICPR.
12. Shrivastava R. A 2013. Hidden Markov model based dynamic hand gesture recognition system using OpenCV Advance Computing Conference (IACC).
13. Gaus, Y.F.A. ,Wong, F 2012. Hidden Markov Model Based Gesture Recognition with Overlapping HandHead/Hand-Hand Estimated Using Kalman Filter, Intelligent Systems, Modelling and Simulation (ISMS) Third International Conference.
14. Moni M.A., Ali A.B.M.S. 2009. HMM based hand gesture recognition: A review on techniques andapproaches, Computer Science and Information Technology ICCSIT 2009.
15. Wilson, A.D. Media Lab. MIT, Cambridge, Bobick, A.F. Parametric hidden Markov models for gesture recognition, Pattern Analysis and Machine Intelligence, IEEE Transactions ,21(9).
16. P. Kishore, Rajesh Kumar, E. Kiran Kumar and S.Kishore 2011. Video Audio Interface for Recognizing Gestures of Indian Sign Language. International Journal of Image Processing (IJIP),5(4), pp. 479-503.
17. J. Han, L. Shao, D. Xu and J.Shotton 2013. Enhanced Computer Vision with Microsoft Kinect Sensor: A Review. IEEE transaction on Cybernetics,43(5), pp.1318-1334.
18. S. Wu, F. Jiang, and D. B. Zhao 2012. Hand gesture recognition based on skeleton of point clouds. IEEE fifth International Conference on Advanced Computational Intelligence (ICACI), pp. 566-569.
19. Y. Zhang, S. Zhang, Y. Luo, and X. D. Xu 2010. Gesture trajectory identification and application based Kinect depth image. Application research of computer, vol. 29, no. 9.
20. Z. Ren and J. S. Yuan 2013. Robust part-based hand gesture recognition using kinect sensor. IEEE Transactions on Multimedia, vol. 15, pp. 1110-1120.
21. P.Karthick , N.Pratibha, V.B.Rekha, S.Thanalaxmi 2014. Transforming Indian Sign Language into Text using Leap Motion. International Journal of Innovative Research in science, Engineering and Technology, pp:10906-10907.
22. Shiguo Lian, Wei Hu, Kai Wang 2014. Automatic User State Recognition for Hand Gesture Based Low Cost Television Control System. IEEE Transaction on Consumer Electronics. pp: 107-112.
23. Murat Kus , Umit Kacar, Murvet Kirci, Ece Olcay Gunnes 2013. ARM Based Ear recognition Embedded System. Eurocon, pp: 2021-2024.
24. Yuan Yao, Yun Fu 2013. Contour Model Based Hand Gesture Recognition Using Kinect Sensor. IEEE Transaction on circuits and System for video Technology. pp: 1-6.
25. Cao Dong, Ming C. Leu, Zhaozheng Yin 2015. American sign language Alphabet Recognition Using Microsofy Kinect. Computer vision and Pattern Recognition workshop, IEEE conference.