HAND GESTURE RECOGNITION USING KINECT

By

Yi Li

B.S., Communication University of China, 2010

A Thesis
Submitted To the Faculty of the
J. B. Speed School of Engineering of the University of Louisville
in Partial Fulfillment of the Requirements
for the Degree of

Master of Science

Department of Computer Engineering and Computer Sciences
University of Louisville
Louisville, Kentucky

May 2012

HAND GESTURE RECOGNITION USING KINECT

By

Yi Li
B.S., Communication University of China, 2010

A Thesis Approved On

April 9, 2012

by the following Committee

Adviser - Ming Ouyang, Ph.D.

Dar-jen Chang, Ph.D.

Ibrahim Imam, Ph.D.

Thomas Tretter, Ed.D.

ACKNOWLEDGMENTS

This thesis would not have been possible without the help from all the kind people. I thank these people for kicking me towards the goal for my best.

Above all, I am heartily thankful to my adviser, Dr. Ming Ouyang, whose encouragement, guidance, and support from the beginning to the final stage enabled me to complete this thesis.

It is an honor for me to have Dr. Chang, Dr. Imam, and Dr. Tom Tretter on my thesis committee. I am grateful for their comments, insights, and suggestions.

I would like to thank my host family, especially Christina House, for treating me like family to help me adjust to a new country ever since I came to US. I would also like to thank Hong Zhou, my best friend, for having always been there to offer moral support.

Last but not least, I offer my sincere regards and blessings to all those who supported me in any respect during the completion of my Master's degree.

I dedicate this work to my wonderful parents, who love me the most in this world.

ABSTRACT

HAND GESTURE RECOGNITION USING KINECT

Yi Li

April 18, 2012

Hand gesture recognition (HGR) is an important research topic because some situations require silent communication with sign languages. Computational HGR systems assist silent communication, and help people learn a sign language.

In this thesis, a novel method for contact-less HGR using Microsoft Kinect for Xbox is described, and a real-time HGR system is implemented with Microsoft Visual Studio 2010. Two different scenarios for HGR are provided: the Popular Gesture with nine gestures, and the Numbers with nine gestures. The system allows the users to select a scenario, and it is able to detect hand gestures made by users, to identify fingers, and to recognize the meanings of gestures, and to display the meanings and pictures on screen. The accuracy of the HGR system is from 84% to 99% with single-hand gestures, and from 90% to 100% if both hands perform the same gesture at the same time.

Because the depth sensor of Kinect is an infrared camera, the lighting conditions, signers' skin colors and clothing, and background have little impact on the performance of this system. The accuracy and the robustness make this system a versatile component that can be integrated in a variety of applications in daily life.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Motivation

There is always a need to communicate using sign languages, such as chatting with speech and hearing challenged people. Additionally, there are situations when silent communication is preferred; for example, during an operation, a surgeon may gesture to the nurse for assistance. It is hard for most people who are not familiar with a sign language to communicate without an interpreter. Thus, software that transcribes symbols in sign languages into plain text can help with real time communication, and it may also provide interactive training for people to learn a sign language. Gesture recognition has become an important research field with the current focus on interactive emotion recognition and HGR.

Traditionally, gesture recognition requires high quality stereoscopic cameras and complicated computer vision algorithms to recognize hand signals; the systems often turn out to be expensive and require extensive setup. Microsoft Kinect provides an inexpensive and easy way for real-time user interaction. Kinect, originally designed for gaming on the Microsoft Xbox platform, uses a color sensor and a depth sensor to capture color (RGB) images and the associated depth (distance) data. It allows the development of algorithms that classify and perform recognition of the image data. The software driver released by Microsoft called Kinect Software Development Kit (SDK) with Application Programming Interfaces (API) give access to raw sensor data streams as well as skeletal tracking. However, there is no hand specific data available for gesture recognition, although it does include information of the joints between hands and arms. Previous researches on computer vision and hand detection have established solid groundwork for gesture recognition. However, only a few Kinect-based systems were developed for HGR, and only a few gestures were recognized. Related work will be reviewed in Chapter 2.

Another traditional approach for hand-tracking is known as glove-based technologies. All kinds of glove devices have been invented as input devices to determine the positions of fingers and palms in the 3D space, and thus to track hand motions and recognize gestures. Glove-based technology produced good applications in translating sign languages in the past decades. Recently, there are several commercial products for human-computer interaction (HCI). For example, the Wii Remote sensor (Wiimote) by Nintendo provides motion sensing using accelerometer and optical sensor technology to allow users to interact with gaming consoles. A programmer Johnny Lee has developed a finger tracking system using Wii Remote, which works in the 2D space and is capable of tracking up to four finger points at the same time. PlayStation Move by Sony Computer Entertainment (SCE) is another example of motion-sensing game controllers that can be used in human-computer interaction research. It uses a PlayStation Eye to get the distance through the uniform spherical shape and the known size of the orb light source at the head of this controller. The precision and accuracy of distance is very high. However, these devices have the same shortcoming: They all depend on some input devices that are attached to or held by hands in order to measure distance data.

In this thesis, an HGR system using Microsoft Kinect for Xbox has been implemented. A user of the system may first choose one of several predefined scenarios such as Popular Gestures and the Numbers. The system then detects hand gestures made by the user, compares them with the signs in the chosen set, and displays the matched meaning and the corresponding picture on the screen. The system represents a new approach to human-computer interaction that uses nothing but bare hands as the input media.

1.2 Hand Gesture Recognition System

The HGR system can be divided into three parts according to its processing steps: hand detection, finger identification, and gesture recognition. The system has two major advantages. First, it is highly modularized, and each of the three steps is capsuled from others; second, the edge/contour detection of hand as well as gesture recognition is an add-on layer, which can be easily transplanted to other applications. The details of the methods and implementation of this system are discussed in Chapter 3.

Depth data is generated and converted from the raw image data of Kinect sensor by an open-source framework called OpenNI (Natural Interaction), with an open-source driver called SensorKinect by PrimeSense, which makes Kinect for Xbox compatible with Microsoft Windows 7. Using the newest version of graphics driver, the system is developed in C# of Microsoft Visual Studio 2010. The accuracy of the HGR system is from 84% to 99% with single-hand gestures, and from 90% to 100% if both hands perform the same gesture at the same time. The identification of all individual fingers has never been done in the literature with the Kinect platform to the best of my knowledge.

This system has several key features:

- Capable of capturing images in the dark
- Identifying fingers of up to two hands, under all reasonable rotations of the hands
- Translating and displaying gestures in real time

- Allowing user to choose different scenarios

This system is able to accomplish its task in the dark because Kinect uses an infrared camera for depth image. In addition, as the frame rate for Kinect sensor output is 30Hz, the process of gesture recognition can be considered as finished in real-time. A practical sensing range for Kinect is 1.2 - 3.5m when the raw data is processed by the Xbox software. For the purpose of HGR, the hands have to be closer than that in order to resolve the details of fingers. Therefore, the effective range of detecting hands and gestures is set to be between 0.5m and 0.8m.

1.3 Organization of the Thesis

Chapter 2 reviews previous work and useful algorithms on HGR using videos, data gloves, and depth cameras; it also surveys the latest related work on Kinect. Chapter 3 describes the architecture and implementation of the system. Chapter 4 contains analyses of performance and accuracy, as well as some future directions to improve the system.

CHAPTER 2

LITERATURE SURVEY

Sign Language Recognition (SLR) has been studied for decades since human-computer interaction stepped into people's lives. Computational interpretation and translation systems can facilitate daily communication for speech and hearing challenged people. Generally, a sign language consists of three parts: finger-spelling, word level sign vocabulary, and non-manual features. Therefore, HGR forms the basis in translating sign languages. That is to say, understanding the structure of sign languages is the starting point to further solve the problems in developing hand motion tracking method. Literature on SLR, especially about 2D video-based recognition, is reviewed in Section 2.1.

Studies aiming at tracking information of hands, including shape, position, and motion, can be traced back to the 1970's. Optical, magnetic, or acoustic sensing devices were attached to hands to report their positions. Later on, a glove-based system was described and implemented, which became a common and matured approach in the field of hand tracking. Throughout the past 20 years, a large variety of glove devices as input media for HCI have been built, of which some have remained in research labs and others have reached the marketplace. Literature about recent glove-based hand tracking research is reviewed in Section 2.2.

A more recent technology for gesture recognition is to incorporate the information of object distances, or depths, normally captured by a 3D camera, or a set of cameras that produce 3D images. It is also a contact-less user interface, in contrast to glove-based devices or handheld remote sensors such as Wiimote mentioned in Chapter 1. Different algorithms are proposed to separate objects from the background and to extract features of the objects, such as the body skeleton or hands, in order to track their motion. Literature related to depth-based recognition is reviewed in Section 2.3.

Some researches were done with Kinect since its first launch in 2010. Among them there are several systems focusing on hand gesture recognition, most of which were making use of depth-based techniques. Literature about most related work on HGR using Kinect is reviewed in Section 2.4.

2.1 Sign Language Recognition

Starner *et al.* [18] presented two vision-based SLR systems using hidden Markov models (HMM): one used a second-person view with a desk mounted camera and the other was the first-person view with a camera mounted on a hat worn by the user. HMM was used for training and continuous motion tracking. Both systems used a skin color matching algorithm for hand tracking. Once a pixel of skin color was found, they checked the eight nearest pixels to search for similar color areas. The facial area was discounted based on the assumption that its position is almost fixed while hands are always moving. They were not able to separate two hands when they overlap each other due to the 2D video limitation. Therefore, they simply assigned the whole area to each hand whenever occlusion happened. Both systems were trained to recognize American Sign Language (ASL) sentences randomly chosen from the form of "personal pronoun, verb, noun, adjective, (the same) personal pronoun" for a 40-word lexicon; four hundred sentences were used for training and one hundred sentences were used for testing. In comparison, the second-person view system had a word accuracy of 92% while the first-person view system had a word accuracy of

98%. The high accuracy indicated that HMM is good for the purpose of continuous motion tracking. However, neither of these two systems have provided a feedback view for the signer himself. In addition, there is no good solution to compensate for head and hand rotations, especially for the first-person view system, in which the rotation of the head may significantly affect the quality of recognition. As lexicon grows large, it requires defining and extracting additional features to maintain the accuracy. Furthermore, no finger-spelling recognition was attempted in this project.

Using color-based recognition on hand tracking may involve some variability in the results due to the difference in people's skin color, not to mention that it needs to distinguish hands from face. Besides, it requires signers to wear clothes of contrasting colors with long sleeves to cover arms. Ong *et al.* [15] provided another approach in which they used a boosted cascade of classifiers that detected hand shapes in grey scale images in order to track hands. First, they collected numerous images of hand shapes. These images were grouped by applying the K-medoid clustering algorithm. Then a tree structure was formed to contain two layers of "weak" hand detectors. At the first layer, each group of hand images was summarized by one representative image. The classifiers at the first layer chose all candidate image blocks that may contain hand shapes, and then passed them to the second layer to be compared to all the images in the corresponding clusters. The FloatBoost algorithm was used to perform boosting in finding weak classifiers. In an experiment of 5,013 hand images, the first layer achieved 98% accuracy in detecting the existence of hands, and the second layer achieved 97.4% accuracy in hand shape detection. These high levels of accuracy can be explained as the choices of unsupervised K-medoid clustering method and the FloatBoost algorithm. Yet the system was not tested on any streams of images containing hand motions; furthermore, the chosen images all had similar and simple background.

Cooper *et al.* [4] employed both of the aforementioned methods to realize an SLR system of a large lexicon. The first layer had three types of classifiers that detected *tab* (placement), *sig* (movement) and *ha* (arrangement) from visemes, which are visual representations of speech sounds in sign languages. First it performed preprocessing of skin segmentation with a Gaussian skin color model to identify skin and a normalized histogram to identify background. After the three features (*tab*, *sig*, and *ha*) were extracted, it chose from two different types of classifiers acting on 2D features in boosting: local binary pattern and additive classifiers. The boosted visemes were put together to generate a binary feature vector and sent to the second layer, where high level classifiers performed word recognition using Markov chains. In an experiment of 164 signs with 1,640 examples, the first layer achieved around 30% accuracy; when combined with the second layer, the accuracy was increased to 72.6%, a significant improvement by using two levels of classifiers for a large lexicon.

It is not robust enough to consider only hands when doing SLR, especially when processing one stream of images because hands often overlap. Buehler *et al.* [2] overcame this by detecting hands and arms at the same time. During the recognition, every pixel was assigned to either the limb model or the background model to avoid the ambiguity from hand, forearm, and upper arm occlusion. A sampling based method for single frames was used to propose arm locations. Distinctive, unambiguous frames were identified and linked together by tracking the arm configurations. This system was tested with BBC broadcast footage using a continuous sequence of 6,000 frames performed by three signers. The result accuracy was above 91% when there was little arm overlap, and it was at least 59% when there was much overlap. When pure color features with Histogram of Oriented Gradients, and distinctive frames tracking framework were combined together, the accuracy was boosted significantly.

Most previous researches on SLR focused on isolated videos recorded under laboratory conditions. They had issues such as small lexicons, and there was no fair way to compare and measure their performance directly. What is more, some results suffered the signer-dependent problem and some results relied on having simple background. Dreuw *et al.* [7] collected a large publicly available database called RWTH-BOSTON-400, which becomes a benchmark for evaluating automatic SLR systems and statistical machine translation systems. The database contains the subsets of ASL published by the National Center for Sign Language and Gesture Resources at Boston University. This benchmark dataset consists of 843 sentences, two male signers and two female signers. With different clothing and different setup poses, there are nine speaker setups. Dreuw *et al.* [6] reviewed this database with some other video benchmark databases, providing criteria to evaluate hand and head tracking algorithms; the proposed criteria were tracking error rate (TER) and word error rate (WER). Dynamic programming tracking (DPT) in [5] was cited to perform hand tracking, which is a model-free and signer-independent tracking method. DPT avoided taking possibly wrong local decisions by tracking back from the last frame of the sequence to the first frame to get a best path with the highest score given by a score function, which was calculated for every frame starting from the very beginning. By integrating into an HMM, this model allowed simultaneous tracking and recognition. In [6] it was tested on the RWTH-BOSTON-104 dataset and the Corpus-NGT dataset. In the best case, the DPT method had a TER as low as 8.37% for the full RWTH-BOSTON-104 dataset. However, it was not robust enough to perform on the Corpus-NGT dataset due to the high number of near-profile images in this database, while DPT was only trained on near-frontal images.

2.2 Glove-based Gesture Recognition

At the finger-spelling level of ASL, several signs of letters are similar to each other. Figure 1 shows the gestures of the ASL alphabet. As an example, letters 'A', 'E', 'M', 'N', 'S', and 'T' are all formed by a closed fist with a little variation in thumb placements; as another example, 'K' and 'V' both use index finger and middle finger with the same angle, and the only difference is again in the thumb placements. The overlaps of fingertips make gesture differentiation a difficult task for 2D video-based recognition systems. Accurate data of individual finger information would be needed. Therefore, glove-based sensing systems have been studied for decades to solve this problem. Although it seems inconvenient for users to wear extra things for the purpose of recognition, glove-based systems do have the advantage to make up the 'cumbersomeness' by largely increasing the accuracy in finger-spelling recognition.

Hernandez-Rebollar *et al.* [11] implemented a glove-based system for ASL finger-spelling recognition, in which the 26 ASL letters, as well as two additional signs 'space' and 'enter', were recognized in real-time. It provided an interface for users to form words or phrases and to send to a speech synthesizer. The system consisted of the Accele Glove, a micro controller, and five MEMS (Micro Electro Mechanical System) dual-axis accelerometers. During training, the micro controller read finger positions provided by the Accele Glove and sent packages of 10 bytes to a PC, where a set of posture features was extracted, and a three-level hierarchical classifier was built. Programmed with the trained classifier, the micro controller recognized the finger-spellings, and then sent the corresponding ASCII to a voice synthesizer so that the system actually spoke out the letters. This system reached an accuracy of 100% for 21 out of the 26 letters, and the worst case was the letter 'U', with an accuracy of 78%. Based on the same concept, Bui *et al.* [3] developed a new SLR system for the Vietnamese Sign Language. They added one more sensor on the back of the hand

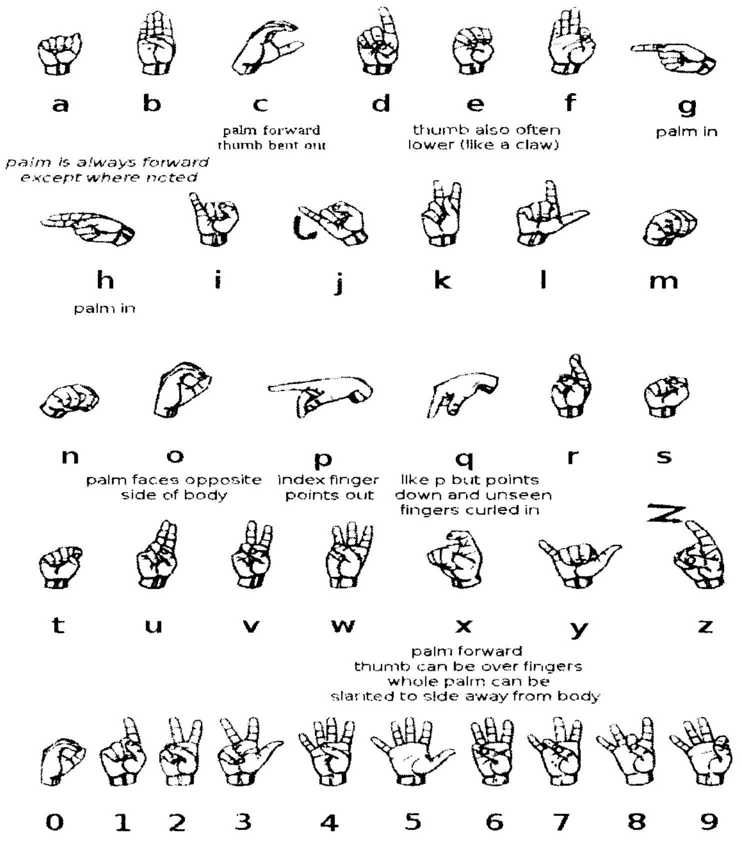


Figure 1. ASL Alphabet; source: Wikipedia

to improve the process. A different classification for letters was introduced, in which all letters were divided into three clusters according to the X-axis value of the palm, and three fuzzy rule-based systems were constructed for further classification without having to collect training samples. Bending or flexing of fingers were measured in five levels: Very-Low, Low, Medium, High, Very-High. Thus a set of "if-then-else" statements were used in programming. It showed promising results in recognition accuracy: Twenty out of 23 letters were tested with an accuracy of 100%, and the worst case was on the letter 'U' with an accuracy of 79%.

Another data glove device was presented in [20], providing not only finger positions, but also the force that fingers applied on an object. The glove was made of cotton and coated by rubber. Flexsensors as finger position sensors were attached to the glove by cyanoacrilic glue. The data taken by position sensors were sent to a PC via two micro controllers: the "root" micro controller and the "slave" micro controller. The PC used a program to decode the position information. The strain gauge force sensor was attached to the thumb; its output voltage was proportional to the applied force. A current source and an A/D converter were used to power the bridge on the force sensor. The PC ran a calibration program for both position sensors and force sensor, and then it was ready to decode the results. This data glove provided a good approach for virtual reality applications, with interesting results of how much force one would use to pick up or squeeze things. Unfortunately, it did not perform hand motion tracking and precise recognition.

One may doubt that despite the advantages of glove-based systems, the glove devices are impractical because they are usually ugly and heavy. Because different sensors and controller chips are needed, the price is another problem for widespread use. Kenn *et al.* [12] came up with a way to integrate glove-based devices into multiple applications with the help of a context framework. The textile glove with integrated electronics device was cool in appearance, and was able to perform in at least three applications as demonstrated: to move/zoom/select parts of a map, to navigate to a remote control in presentation, and to direct a toy robot to move left/right, forward/backward. Gestures were chosen to be simple but sufficient in interacting with applications. One problem was that this device can only detect gestures in the X and Y axes, without the ability to detect motion in the Z axis such as the so-called "yaw". In addition, to achieve wearability, light weight, and cool appearance, the accuracy of recognition was sacrificed.

The glove systems described so far mainly work with 2D information. However, real life applications involve gestures in the 3D space. More recently, the trend of glove-based systems is pointing to 3D recognition, and the reduction of price without sacrificing accuracy.

Kim *et al.* [13] developed a 3D hand motion tracking and gesture recognition system using a data glove called KHU-1. The data glove interacted with a PC via a bluetooth device. It successfully performed hand motion tracking such as fist clenching, hand stretching and bending. A rule-based algorithm was used in simple HGR at two different angular positions: horizontal and vertical. Three gestures (scissor, rock, and paper) were tested with an accuracy of 100% for 50 trials each. The recognition in 3D and the wireless transmission were good improvements, but they incurred time delay. However, the tested gestures were too simple and too few to prove the robustness of this system.

An inexpensive visual motion data glove with high recognition accuracy was presented by Han [9]. The glove device used a single-channel video instead of commonly used motion-sensing fibers or multi-channel videos, with a reconstruction algorithm to compensate for the shortcomings of single-channel videos. Thin-var-type optical indicators were attached to the glove, to avoid high frequency of optical occlusions. A monocular camera was used to capture hand motion, and then a visual analyzer algorithm detected those optical indicators and recovered 3D positions of fingers and joints. Three classes of scenarios (left/right clicks, numbers, and the OK sign) were computed and reconstructed into 3D images in MATLAB. The computing time was robust to image noise and error ratios. The advantage of this approach was that it had consistent computing time and low estimation errors, and the materials for equipment were affordable. However, it was not a real-time system.

2.3 Gesture Recognition Using Depth Data

As stated before, using 2D video-based algorithms in tracking hands leads to confusion when there is occlusion. Instead, if color-based segmentation is used, the hands are not easy to be separated from the face due to their similar skin colors; furthermore, its performance suffers when the lighting conditions change or the background is nonuniform. Data glove devices that provide hand positions may be a good way to augment color-based segmentation, but they cause uncomfortable user experience (UE) because of the physical contact. Fortunately, a new contact-less approach came out in the past ten years: using depth discontinuities to separate hands from the background. In this way, depth-based systems avoid all these problems mentioned above.

Nanda *et al.* [14] presented a method to track hands in highly cluttered environments using 3D depth data provided by a sensor that was based on the time-of-flight (ToF) principle to capture depth and color information simultaneously using the same optical axis. The potential fields of possible hands or face contour were calculated by three algorithms: getting potential fields by using distance transform, k-components-based potential fields with weights, and basin of attraction. The system was tested in head tracking and hand tracking on ten people with good results. It was able to track hands with occlusion, and to recognize simple motion like "step back" and "stop". However, it is not suitable for real-time systems because hand shapes are varied so that a lot of contours are needed for recognition.

Argyros *et al.* [1] proposed a 3D hand tracking method using a stereoscopic system of two video streams. It combined color-based tracking and 3D hand tracking, and it could run in real time. Images were captured by each camera; hand blobs were marked by 2D color-based hand trackers in both video streams, and then matched together by a calibration calculation. After this, hand contours were aligned and reconstructed in the 3D space. This method was applied to several applications. One experiment was to operate a CD player by hand gesture, in which the contour of the hand and index finger were detected. The ability of processing in real time is an advantage, yet the estimated depth data using the stereoscopic system is rather noisy, and the need for calibration increases system complexity.

In [21], Van den Bergh *et al.* used a ToF camera instead of stereoscopic cameras to enhance recognition. A ToF camera with a low resolution (176×144 pixels) was used to get depth image for segmentation, and it was paired with an RGB camera with a high resolution (640×480 pixels) for hand detection. The ToF camera and RGB camera were calibrated at the beginning, and a threshold distance was defined to discard background images according to the depth data. The remaining pixels were passed through a skin color detection to get hand data. The skin color used for detection was decided by a pre-trained, adaptive skin color model, which was updated with color information taken from the face. Three situations were evaluated in hand detection: the hand was next to the face, the hand overlapped with the face, and a second person was behind the tester. Depth-based detection achieved more than 98% accuracy in all three situations, while the accuracy of color-based detection decreased dramatically from 92% of the first situation to 19.8% of the third situation. Average

Neighborhood Margin Maximization transformation was used to build the classifier for gesture recognition, where the Haarlet coefficients were calculated to match hand gestures stored in a database. The RGB-based recognition showed an accuracy of 99.54%, and the depth-based recognition showed an accuracy of 99.07%, while the combination of the two methods showed 99.54%. This suggests that depth-based recognition may be good enough to form the basis of a recognition system especially in hand segmentation.

2.4 Most Related Work Using Kinect

After the launch of Microsoft Kinect in November, 2010, several exciting recognition systems based on this device were developed in less than 18 months. The resolution of its RGB camera and depth camera are both 640 × 480 pixels, which are fairly sufficient under many situations.

Yang *et al.* [22] proposed a gesture recognition system using depth information provided by Kinect, and implemented in a media player application. It was able to recognize eight gestures to control the media player, with a maximum confusion rate of 8.0%. The algorithm for hand tracking is to first find the handwaving motion based on the assumption that a user tends to start an interaction session with such a motion. A continuously adaptive mean shift algorithm was applied to track the hand by using the depth probability and updating the depth histogram at each frame. The hand trajectory was examined by a 3D feature vector, and the gesture was recognized by an HMM. This system demonstrates the applicability of using Kinect for gesture recognition in a contact-less UI.

The previous system was not able to find fingertips; thus it was limited to recognize only motion gestures like wave, and move up/down, left/right, and forward/backward. A method to track fingertips and the centers of palms using Kinect was presented by Raheja *et al.* [17]. It applied thresholding on the depth of hand regions for segmentation. Then the palm was filtered and subtracted from the hand, so that only the fingers were left in the image. Under most situations when the hand was in front of the user, the fingers should be closest to the Kinect with the shallowest depth. Therefore, by determining the minimum depth, fingertips were found. The center of the palm was determined by finding the maximum of distance within the image of the hand. When fingers were extended, the accuracy of detecting fingertips was nearly 100% accuracy, and that of palm centers was around 90%. However this method did not attempt at gesture recognition. He *et al.* [10] proposed another approach using depth data provided by Kinect to detect fingertips. First, it found hand points by thresholding on depth data, and then generated the convex hull containing the hand by Graham Scan [8]. Fingertips were detected by calculating the angle between candidate points. After fingertips were found, the mouse clicking motion was recognized and tested on the popular game Angry Bird; that is, it recognized only one gesture.

CHAPTER 3

METHODS

The system of the present thesis consists of three main components: Hand Detection, Finger Identification, and Gesture Recognition. This system is built upon the Candescent NUI [19] project, which is freely available online. The OpenNI framework is used to extract depth data from the Kinect sensor. From the depth data, it is easy to distinguish the user's hands from the background. When both hands are present and at some distance apart, two hand clusters will be found; otherwise, there is only one cluster. The convex hull containing the hand is computed, and then the contour of the hand is traced. By examining the convex hull and the contour, fingertips are detected and identified according to their relative distances. Gestures are recognized by matching feature vectors with different sets of predefined classifiers.

3.1 Working Environment Introduction

There are plenty of open-source frameworks developed for HCI, especially for communication between physical devices like hearing or motion sensors and UIs. OpenNI is one of them that provides APIs for NI applications in multi-language and for cross-platform utilization. Currently it supports three major platforms: Windows 7, Mac OSX, and Linux. Applications built on OpenNI are usually portable and easy to be deployed to other middleware.

Figure 2 describes the three abstract layers of the OpenNI concept [16]. Each of the layers represents an integral element:

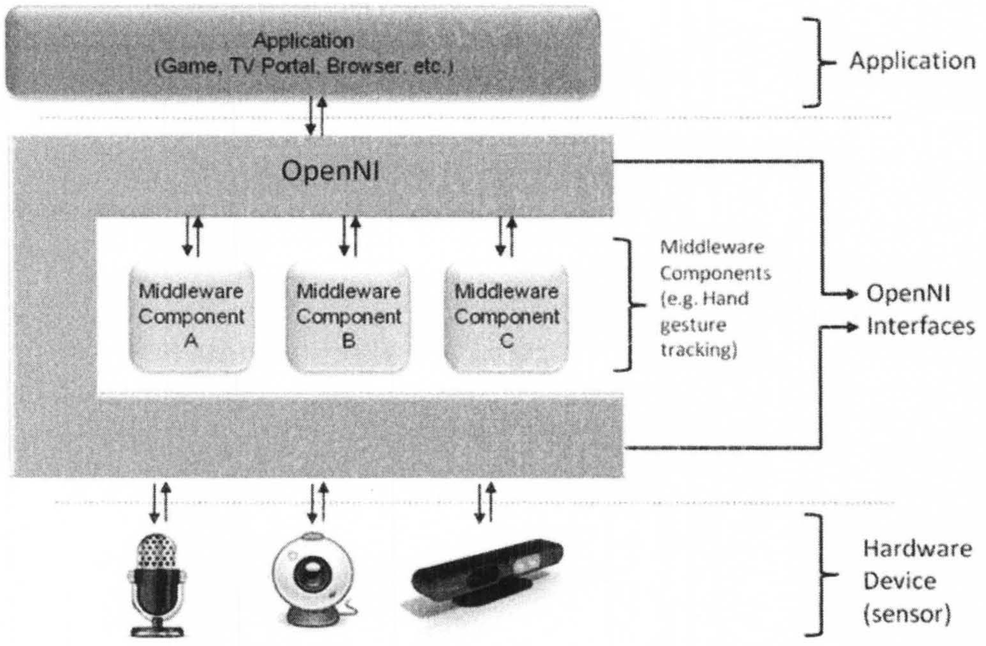


Figure 2. OpenNI Abstract Layered View; source: [16]

- The top layer represents the applications implemented for natural interactions.
- The middle layer represents the OpenNI framework, in which it not only interacts with physical sensing devices as well as software, but also communicates with middleware components.
- The bottom layer represents all kinds of sensing devices, including visual and audio sensors.

OpenNI provides two kinds of production nodes: sensor-related production nodes and middleware-related production nodes. Each node is a set of components that generate a specific type of data for NI based applications. There are five sensor-related production nodes currently supported in OpenNI: Device, Depth Generator, Image Generator, IR Generator, and Audio generator. The system of the present thesis makes use of the first two of them, where the Device node enables device configuration, and Depth Generator generates a depth-map.

The first generation of Microsoft Kinect is designed for the XBox 360 game console. It is made to be compatible with other operating systems via the SensorKinect driver published by PrimeSense Company. The driver is also an open-source module which supports multiple platforms (Windows 7, Mac OSX, and Linux). By working with the SensorKinect driver, the OpenNI is able to initialize its production nodes and to start extracting data from Kinect.

The system of the present thesis is written in C# using Windows Form of Microsoft Visual Studio 2010.

3.2 Hand Detection

In order to detect hands, the first step is to separate the hands from the background. A depth threshold is set manually in advance to specify the depth range where the clustering will take place later. In this system, the detection range, *zRange*, is set to be between 0.5m and 0.8m; up to two hands in this range can be detected. Pixels with depths outside *zRange* are ignored in the rest of the gesture detection process. Henceforth, a hand pixel may also be referred to as a point. These hand pixels are projected to a 2D space for subsequent analysis. The distance between two pixels $p_1(x_1, y_1)$ and $p_2(x_2, y_2)$ in the 2D space is defined as:

$$D(p_1, p_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}. \tag{1}$$

Within *zRange*, the K-means clustering algorithm is applied to partition all the pixels into two groups according to their positions in the 2D space. K-means clustering is a method to partition n observations into k clusters, $C_1, C_2, \ldots, C_k$; each observation is identified with the nearest mean, $\mu_i(x, y)$, which is calculated as the mean of points in C_i. K-means clustering minimizes the within-cluster sum of squares:

$$\arg\min_{C} \sum_{i=1}^{k} \sum_{p_j(x,y) \in C_i} \left\| p_j(x, y) - \mu_i(x, y) \right\|^2. \tag{2}$$

This hand detection system is running in real-time, and K-means clustering is continuously applied whenever there is change in the input data source, and the value of k is 2. At the beginning of every iteration, each empty cluster is assigned a random center point located within the *zRange*. An iteration consists of two steps. In the assignment step, distances between points and the center point of each cluster are calculated, and the points are assigned to the nearest cluster; in the update step, the new means are calculated to be the centroids in the clusters. The assignment step and the update step are executed alternately. After some iterations, the computation converges when the number of points in each cluster becomes stable. Thus, the points belonging to each hand are clustered. If the distance between the two centroids of hands is less than a pre-defined value, the two clusters are merged into one.

After the points of the hands are clustered, we are ready to detect the convex hull of hands as well as the contour of hands. A set is convex if, for every pair of points in the set, the line segment joining the pair is also contained in the set. Let S be a set of points, $\{p_0, p_1, p_2, \ldots, p_n\}$, in the 2D plane, the convex hull of S is the smallest convex polygon that contains all points in S. The Graham Scan algorithm [8] is used to compute the convex hull of the detected hand clusters. The method is demonstrated in Figure 3, and it works as follows.

Graham Scan Algorithm

1. Find the point $p_0(x, y)$ with the largest Y-axis value. If there are more than one point with the same largest Y-axis value, the one with the smallest X-axis value will be chosen.
2. An imaginary line is constructed between $p_0(x, y)$ and each of the other points. The angles between these lines and the X-axis are measured. The points are then sorted in the ascending order with respect to these angles. In the implementation, the angles being sorted are replaced by the corresponding cosine of

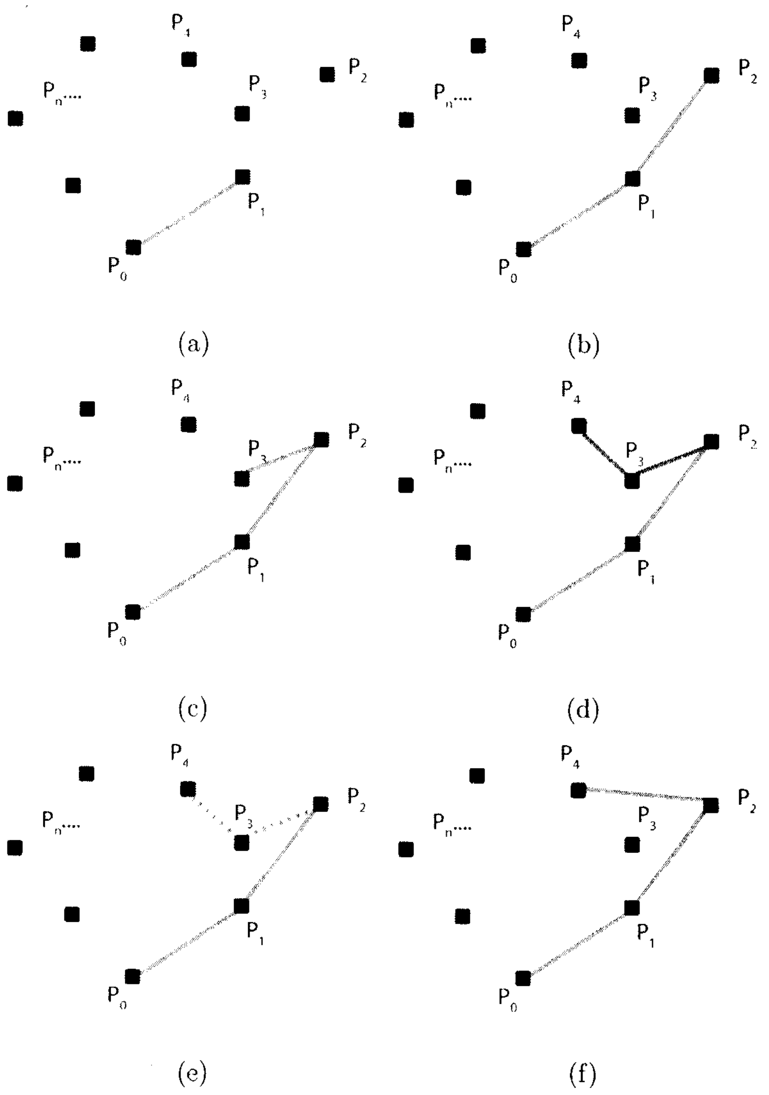


Figure 3. Graham Scan

the angles, for the reason that cosine is a monotonically decreasing function in the domain of 0 to 180 degrees.

3. Form the line segment between $p_0(x, y)$ and the first point in the sorted sequence, $p_1(x, y)$. These two points are on the convex hull.
4. Let $p_0, p_1, \ldots, p_s$ be the partial convex hull constructed. For each of the remaining points in the sorted sequence, p_t, if the line segment $\overline{p_s p_t}$ makes a counterclockwise turn from the the line segment $\overline{p_{s-1} p_s}$, p_t is added to the partial hull, and this step is repeated. If $\overline{p_s p_t}$ makes a clockwise turn from the line segment $\overline{p_{s-1} p_s}$, p_s is removed from the partial convex hull, and the process backtracks to the partial hull $p_0, p_1, \ldots, p_{s-1}$, and this step is repeated.

To implement Step 4, the direction from p_{s-1} and p_s to p_t is determined by the cross product of the two vectors defined by the three points: $\vec{V_1} = (p_{s-1}, p_s)$ and $\vec{V_2} = (p_s, p_t)$.

$$CroVec = \vec{V_1} \times \vec{V_2} = (x_s - x_{s-1})(y_t - y_{s-1}) - (y_s - y_{s-1})(x_t - x_{s-1}) \qquad (3)$$

If $CroVec$ is positive, the direction is counterclockwise; if it is negative, the direction is clockwise; otherwise the three points are collinear.

Hand contours are detected by a modified Moore-Neighbor Tracing algorithm. After the clustering, a group of hand points are found and stored. Figure 4 shows an example of implementing the algorithm to detect contour.

Define $\mathbf{N}(a)$ to be the eight-point neighborhood of a pixel a. Let p denote the current contour pixel. Let q denote the starting pixel of current neighborhood checking. Let $\mathbf{C}$ denote the set of detected contour points, which is initialized to be the empty set.

Contour Tracing Algorithm:

1. From top to bottom, and left to right, scan all pixels on the screen until a pixel s being a hand point is found, which is set as the starting point.

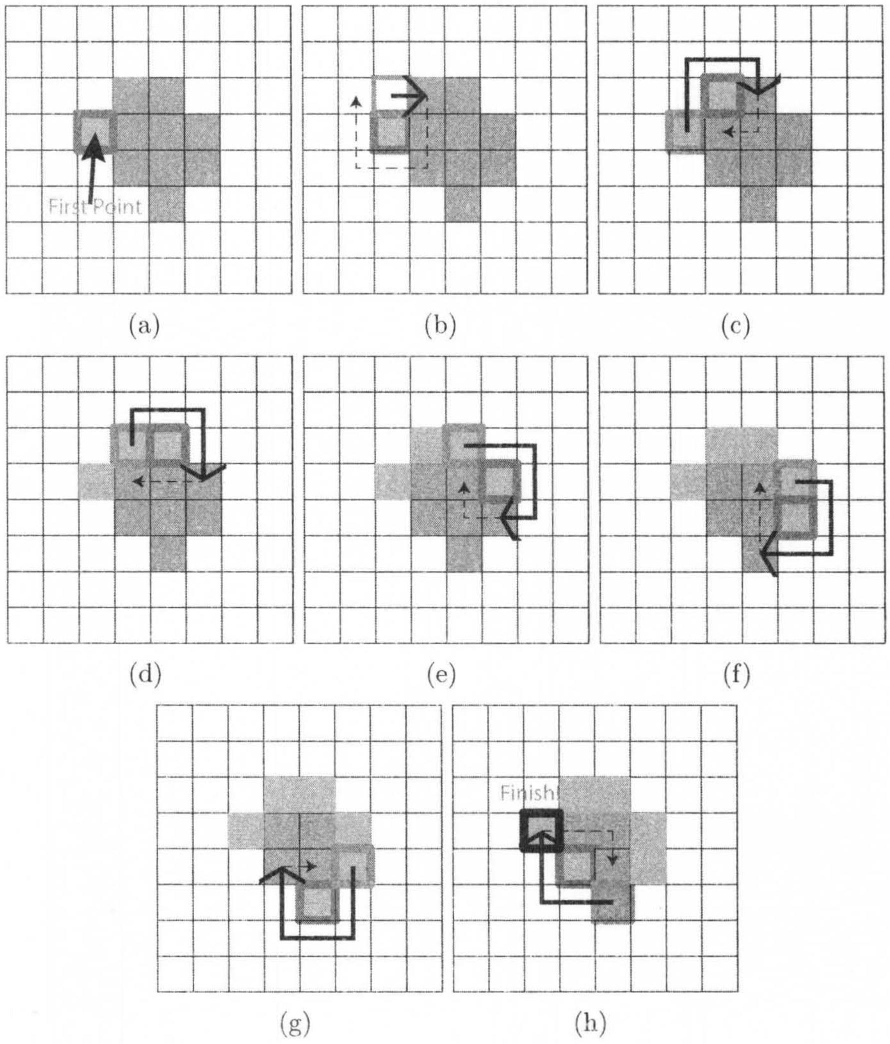


Figure 4. Contour Tracing. To trace the contour of the blue pixels: Let green pixels represent detected contour pixels. The pixel with red boundary denotes the current contour pixel; the pixel with green boundary denotes the starting pixel of neighborhood checking, namely the last contour pixel except for the starting point. The black arrow shows the clockwise path of neighborhood checking; the dashed arrow shows the path that the algorithm does not need to finish, because the next contour pixel is already detected.

2. Set the current contour pixel p to be s. Set the starting pixel of neighborhood checking q to be the point to the immediately north of s.
3. Insert p into $\mathbf{C}$, and calculate the neighborhood $\mathbf{N}(p)$.
4. Start from q, go around the neighborhood $\mathbf{N}(p)$ in the clockwise direction until the next hand pixel r is found.
5. Set q to be p, and p to be the new contour pixel r. Then repeat Step 3 until the starting point s is reached again, or the number of detected points exceeds a maximum value.

Once the hand contours are detected, the centers of the palms are calculated as the centers of the inscribing circles of the hand contours. The positions of palm centers are used to calculate the directions of fingers, which are described in the next section. The centers of the inscribing circles are much more stable than the centroids of hand clusters, because the latter vary a lot when opening/closing hands or bending fingers.

Afterwards, the shapes of hand contours are detected and stored in a hand shape collector.

3.3 Finger Identification

Fingertips are detected by checking the three-point alignment relationship. Candidate points are the points that are on both the convex hull and the hand contour. To speed up the computation, every fourth point is checked by the method shown in Figure 5(a).

Three-point Alignment Algorithm

1. Let $\mathbf{C}$ be the set of all candidate points that are both on the convex hull and the hand contour.
2. For each point p_0 in $\mathbf{C}$, take the two points p_1 and p_2 in the two opposite

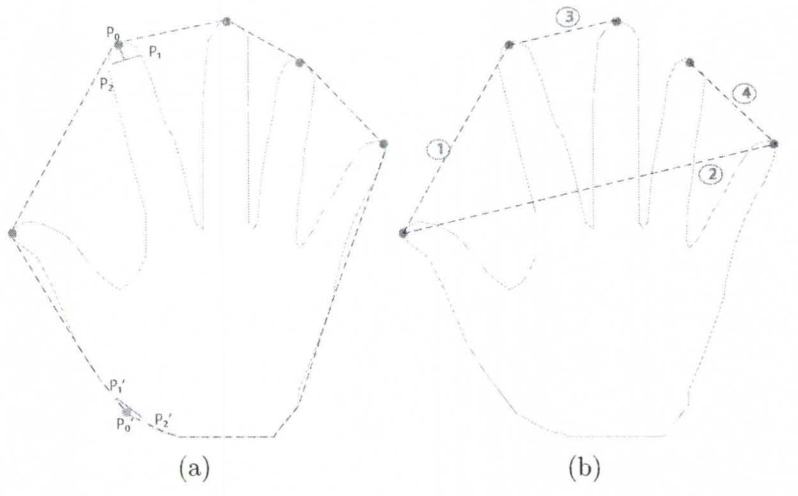


Figure 5. 5(a) Fingertip Detection. The red dots are real fingertips, and the green dot is not a fingertip. The yellow dots are used to check for three-point alignment. The distance between p_0 and the line made by p_1 and p_2 is apparently larger than that of p'_0 and the line made by p'_1 and p'_2. Thus a threshold distance could be used to differentiate fingertip points and non-fingertip points.
5(b) Finger Identification.

directions along the contour that are at a given distance to p_0.

3. Find the midpoint of p_1 and p_2 and calculate the distance between this midpoint and p_0. If the distance is larger than a specific value, the three points are not aligned, i.e., far from being collinear. Then this p_0 is identified as a fingertip. Otherwise go back to Step 2 and check the next candidate point.

After fingertips are detected, their direction vectors are easy to calculate by subtracting the position of the palm center $P_c(x_c, y_c)$ (which is calculated in section 3.2): $\vec{V} = (x - x_c, y - y_c)$. The vectors are pointing from the palm center to fingers.

Finger names are determined according to their relative distances, which require the user to open palm and to extend fingers. Four procedures to identify the names of all fingers are shown in Figure 5(b). The Fingertip class has a string variable to store the corresponding names. The identification of all individual fingers has never been done by anyone with the Kinect platform to the best of my knowledge.

Finger Identification Algorithm

1. The first and easiest step is to identify the thumb and the index finger, since the distance between them is the largest among all neighboring fingers.
2. The little finger is identified as the farthest finger away from the thumb, meanwhile the middle finger is identified as the closest one to the index finger.
3. The remaining one is the ring finger.

The process of detecting hands and identifying fingers are performed every time when the input data source changes. If the same object still exists in the current frame with some transformation compared to the previous frame, all properties of this object is mapped from the previous frame to the current frame; otherwise the disappeared object is collected by an unmapped item collector. The results of finger identification is shown in Figure 6.

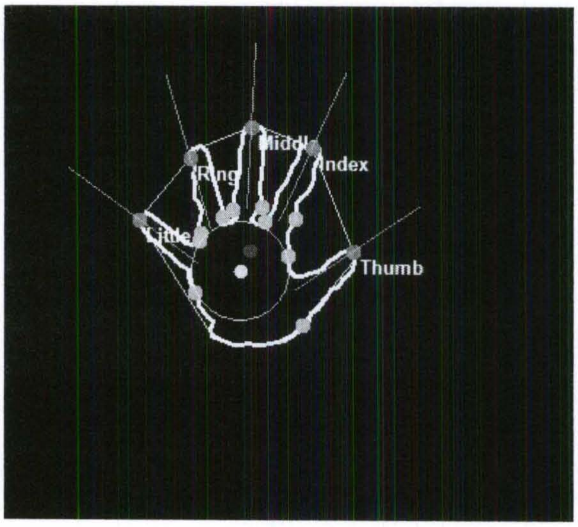
3.4 Gesture Recognition

Once the fingers are identified with their names, we are ready for gesture recognition. Gestures are recognized by passing them through a set of classifiers. The user first chooses one out of several scenarios, such as Popular Gesture and Numbers, to indicate the gesture vocabulary he/she wants to perform. Thus the chosen set of the classifiers is used to perform recognition.

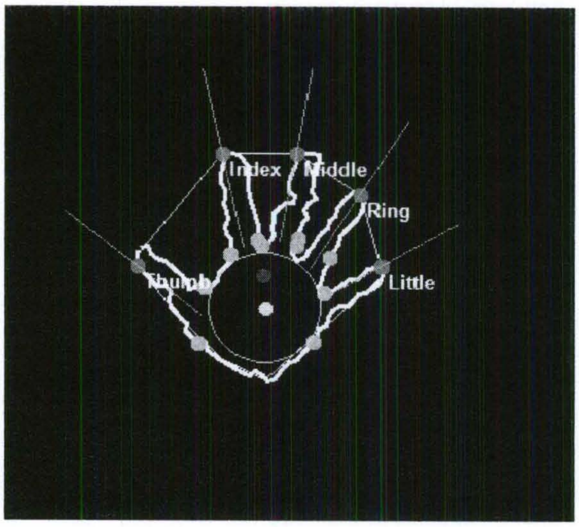
The classifiers are organized in three layers: finger counting, finger name collecting, and vector matching. The angle A between two vectors $\vec{V_1}(x_1, y_1)$ and $\vec{V_2}(x_2, y_2)$ is calculated as:

$$A = \arccos \frac{\vec{V_1} \cdot \vec{V_2}}{\left\|\vec{V_1}\right\| \left\|\vec{V_2}\right\|}. \tag{4}$$

Three layers of classifiers:



(a)



(b)

Figure 6. Fingertip Identification Results.

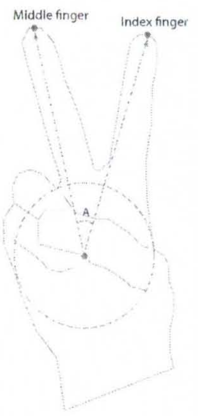


Figure 7. Recognition Example. In the first layer, the system determines that the number of fingers in this gesture is two. Then in the second layer, the finger names of "index finger" and "middle finger" are found. The classifier in the third layer calculates the angle of two finger vectors are within a range of 30 degree to 60 degree. Thus the gesture is recognized as "Victory" in the Popular Gesture scenario, while in the Numbers scenario it is recognized as the number "Two" through a similar process.

- Finger counting classifier: Gestures are first classified by the number of extended fingers, and then sent to the corresponding second layer of classifiers.
- Finger name collecting classifier: In this layer, gestures are further classified by the names of the extended fingers. If the combination of the extended fingers in one gesture is unique among all gestures in the current scenario, the recognition process terminates, and the meaning of the gesture is displayed; otherwise the gesture is sent to the corresponding third layer of classifiers.
- Vector matching classifier: The direction vectors of all extended fingers are measured, and all pairwise angles between extended fingers are calculated. The meaning of the gesture is now classified according to these angles, and then it is assigned to the hand data.

An example of recognizing the "Victory" gesture is shown in Figure 7. The meaning of the gesture and a picture are displayed on screen in the hand drawing layer.

Two scenarios are constructed so far: Popular Gesture and Numbers. The Popular Gesture scenario consists of nine gestures: "Start Gesture", "Thumb-up", "Thumb-down", "Victory", "Okay", "I love you", the letter "L", the letter "Y", "Live long and prosper"; the Numbers scenario consists of numbers from "One" to "Nine". Due to the requirement of finger name collecting and vector matching, the user needs to perform the calibration gesture, namely the "Start Gesture", each time before performing a gesture.

CHAPTER 4

RESULTS AND CONCLUSION

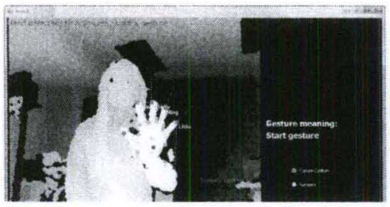
4.1 Results

The HGR results of Popular Gesture are shown in Figure 8.

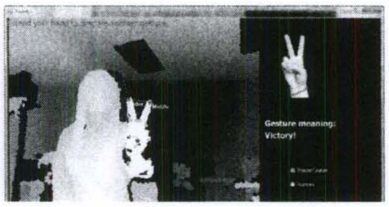
To test this system, four people were asked to perform each of the nine gestures in the Popular Gesture scenario for 100 times: 50 times with the left hand and 50 times with the right hand. The average accuracy of each gesture is calculated and shown in Table 1. For "Start Gesture", the accuracy is nearly 100%, and the names of the fingers are perfectly identified. The best case is the letter "L" with an accuracy of 91%, while the worst case is "Live long and prosper" with an accuracy of 84%. Due to the difficulty of the *Star Trek* gesture, only two of the four people are able to perform "Live long and prosper". Furthermore, if the signer uses both hands to do the same gestures at the same time, the accuracy is significantly increased to more than 90% for all nine gestures.

4.2 Conclusion and Future Work

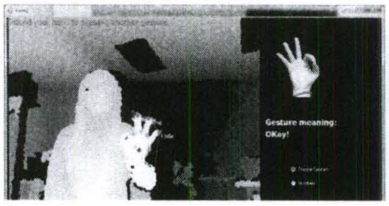
In this thesis, an HGR system based on Microsoft Kinect for Xbox is introduced. The system is motivated by the importance of real-time communication under specific situations, such as chatting with speech and hearing challenged people. It is capable of working in the dark, invariant to signer's skin color, clothing, and background lighting conditions, and it can be easily transplanted to other applications. The system is robust against clutter in the background. Even when the signer's face


Gesture meaning:
Start gesture

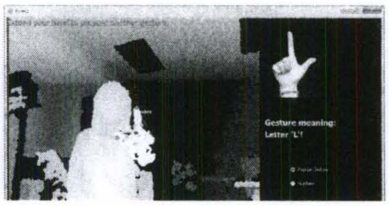
(a)


Gesture meaning:
Victory!

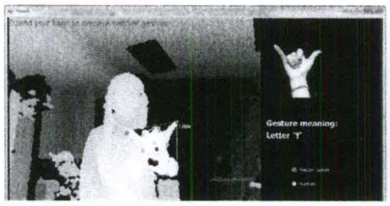
(b)


Gesture meaning:
OKay!

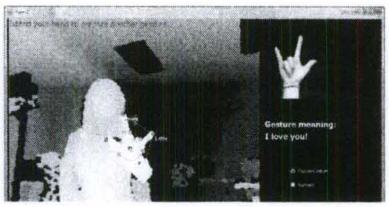
(c)


Gesture meaning:
Letter "L"!

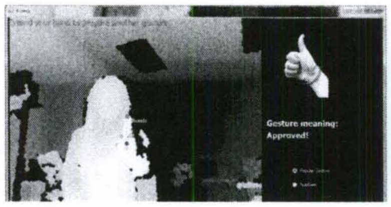
(d)


Gesture meaning:
Letter 'Y'

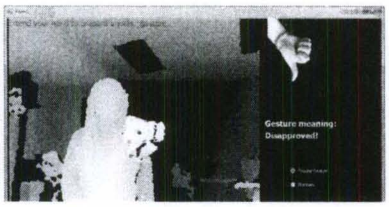
(e)


Gesture meaning:
I love you!

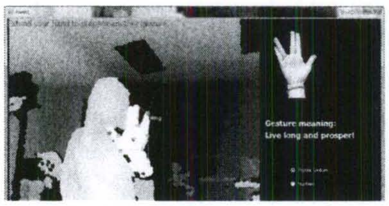
(f)


Gesture meaning:
Approved!

(g)


Gesture meaning:
Disapproved!

(h)


Gesture meaning:
Live long and prosper!

(i)

Figure 8. Recognition Results. 8(a) is the "Start Gesture" for the purpose of calibration; 8(b) is "Victory"; 8(c) is "Okay"; 8(d) is the letter "L"; 8(e) is the letter "Y"; 8(f) is "I love you"; 8(g) is "Thumb-up"; 8(h) is "Thumb-down"; 8(i) is the *Star Trek* gesture, which means "Live long and prosper".

TABLE 1

HGR Accuracy

Gestures	Start	Thumb-up	Thumb-down	L	Y
Accuracy(%)	99	89.5	88.25	91	90.5
Range(%)	98-100	86-96	85-94	88-97	89-94
Gestures	Okay	Victory	Star Trek	I ♡ U	
Accuracy(%)	84.75	89.25	84	85.25	
Range(%)	70-96	85-92	80-89	84-90	

is behind the hands, the hand detection component can separate the hands from the face according to the depth information.

Previous work on HGR systems is reviewed, and these systems are classified into four categories in Chapter 2: 2D video-based SLR systems, glove-based gesture recognition systems, depth-based gesture recognition systems, and Kinect-based HGR systems, which are most related to the work in this thesis. Each group of systems has their own advantages and limitations, which are taken into consideration in the present work.

In the developed system, hands are distinguished from the background by the depth information; specifically, the detection range is between 0.5m to 0.8m. The K-means clustering algorithm is used to obtain two clusters of hand pixels; if the distance between the two cluster means is within a threshold, the two clusters are merged, and only one hand is detected. Then the Graham Scan algorithm is used to determine the convex hulls of hands; a contour tracing algorithm is used to detect hand contours. Afterwards, candidate fingertips are collected by calculating the common pixels that are on both the convex hull and the hand contour, from which fingertips are detected by applying the three-point alignment algorithm. Finger names are determined according to their relative distances, and a direction vector is assigned to each finger. Gestures are recognized by passing them through a set of classifiers that contains three layers: finger counting classifier, finger name collecting, and vector matching. Two scenarios are constructed to be recognized: one is Popular Gesture, which consists of nine popular gestures; the other is Numbers, which consists of nine numbers from "One" to "Nine". The recognition has accuracy of at least 84% for one hand, while it increases to more than 90% when both hands are doing the same gestures.

This present system provides a new approach for HGR at the level of individual fingers. It opens possibilities for HGR systems to be integrated in many practical applications in real life.

Some directions for future work are to further improve the accuracy of recognition, to add more scenarios such as finger-spelling of the alphabet and numerals, and sports referee gestures. Moreover, by incorporating hidden Markov models, the system may allow the recognition of continuous gestures that form words or sentences, and the narrowly defined scenarios may be expanded to discourse contexts. With the recently launched Microsoft Kinect for Windows 7, the sensors are more powerful than the Kinect for Xbox, which makes it more suitable for HGR. It is expected that more applications with excellent recognition results will be developed.

REFERENCES

[1] Antonis A Argyros and Manolis I A Lourakis. Binocular hand tracking and reconstruction based on 2d shape matching. *ICPR*, pages 207–210, 2006.

[2] P. Buehler, M. Everingham, D. P. Huttenlocher, and A. Zisserman. Long term arm and hand tracking for continuous sign language TV broadcasts. In *British Machine Vision Conference*, 2008.

[3] The Duy Bui and Long Thang Nguyen. Recognizing postures in vietnamese sign language with MEMS accelerometers. *Ieee Sensors Journal*, 7(5):707712, 2007.

[4] Helen Cooper and Richard Bowden. Large lexicon detection of sign language. In *ICCV, Workshop Human Comp. Inter*, 2007.

[5] P Dreuw, T Deselaers, D Rybach, D Keysers, and H Ney. Tracking using dynamic programming for appearance-based sign language recognition. *7th International Conference on Automatic Face and Gesture Recognition FGR06*, 62(1):293–298, 2006.

[6] Philippe Dreuw, Jens Forster, and Hermann Ney. Tracking benchmark databases for video-based sign language recognition. *In ECCV International Workshop on Sign Gesture and Activity SGA Crete Greece September 2010*, 2010.

[7] Philippe Dreuw, Carol Neidle, Vassilis Athitsos, Stan Sclaroff, and Hermann Ney. Benchmark databases for video-based automatic sign language recognition. *Pattern Recognition*, pages 6–6, 2008.

[8] R L Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters*, 1(4):132–133, 1972.

[9] Y Han. A low-cost visual motion data glove as an input device to interpret human hand gestures. *Ieee Trans Consum Electron*, 56(2):501–509, 2010.

[10] Guan-Feng He, Sun-Kyung Kang, Won-Chang Song, and Sung-Tae Jung. Real-time gesture recognition using 3D depth camera. In *Software Engineering and Service Science (ICSESS), 2011 IEEE 2nd International Conference on*, pages 187 –190, 2011.

[11] Jos L. Hernndez-rebollar, Robert W. Lindeman, and Nicholas Kyriakopoulos. A multi-class pattern recognition system for practical finger spelling translation. In *In Proceedings of the 4th IEEE International Conference on Multimodal Interfaces*, pages 185–190, 2002.

[12] Holger Kenn, Friedrich Van Megen, and Robert Sugar. A glove-based gesture interface for wearable computing applications. *Applied Wearable Computing (IFAWC), 2007 4th International Forum on*, pages 1 –10, 2007.

[13] Ji-Hwan Kim, Nguyen Duc Thang, and Tae-Seong Kim. 3-D hand motion tracking and gesture recognition using a data glove. In *Industrial Electronics, 2009. ISIE 2009. IEEE International Symposium on*, pages 1013 –1018, 2009.

[14] H Nanda and K Fujimura. Visual tracking using depth data. *2004 Conference on Computer Vision and Pattern Recognition Workshop*, 00(C):37–37, 2004.

[15] Eng-Jon Ong and Richard Bowden. A boosted classifier tree for hand shape detection. In *Proceedings of the Sixth IEEE international conference on Automatic face and gesture recognition*, FGR' 04, pages 889–894, Washington, DC, USA, 2004. IEEE Computer Society.

[16] OpenNI organization. *OpenNI User Guide*, 2010. Last viewed 19-01-2011 11:32.

[17] Jagdish L Raheja, Ankit Chaudhary, and Kunal Singal. Tracking of fingertips and centers of palm using KINECT. *2011 Third International Conference on Computational Intelligence Modelling Simulation*, pages 248–252, 2011.

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

[19] Stefan Stegmueller. Hand and finger tracking with Kinect depth data, 2011. `http://candescentnui.codeplex.com`.

[20] K N Tarchanidis and J N Lygouras. Data glove with a force sensor, 2003.

[21] 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*, pages 66 –72, 2011.

[22] Cheoljong Yang, Yujeong Jang, Jounghoon Beh, David Han, and Hanseok Ko. Gesture recognition using depth-based hand tracking for contactless controller application. In *Consumer Electronics (ICCE), 2012 IEEE International Conference on*, pages 297 –298, 2012.

CURRICULUM VITAE

Yi Li

PERSONAL INFORMATION

Date of Birth	Jul 8, 1989
Citizenship	People's Republic of China
E-mail	`y0li0035@louisville.edu`

EDUCATION

Aug. 2010 - Present	M.S. Computer Engineering and Computer Science, University of Louisville, KY, USA
Sep. 2006 - Jun, 2010	B.S. Optical Information of Science and Technology, Communication University of China, Beijing, China

PROFESSIONAL EXPERIENCE

Aug. 2010 - Present	Graduate Research Assistant, Department of Computer Engineering and Computer Science, University of Louisville
Sep. 2008 - Jun. 2010	Project Leader, Research on Popular New Optical Film, Communication University of China

AWARDS

Aug 2012 - Jul 2014	Provost Fellowship Recipient from University of Louisville (The most prestigious university-wide scholarship at UofL)
Oct. 2009	Third Grade Scholarships for Outstanding Undergraduates, Communication University of China
Oct. 2008; Oct. 2007	Second Grade Scholarships for Outstanding Undergraduates, Communication University of China
Oct. 2008; Oct. 2007	"Three Goods" Student Award, Communication University of China

PUBLICATIONS

Y. Li, *"Hand Gesture Recognition Using Kinect"*, 2012 IEEE 3rd International Conference on Software Engineering and Service Science (ICSESS2012), Beijing, Jun, 2012

Y. Li, *"Digital Image Processing Thesis Preliminary Study on Image Processing Technology and Its Application on Weed Recognition"*, Science and Technology Innovation Herald, China, 2009