RMIT University

Master of Engineering

Vietnamese Sign Language Reader using Intel Creative Senz3D

Student Name:
Nguyen Van Duc

Student Number:
s3411503

Professional Engineering Advanced Project Part B
End of Semester Report
May 2014

Supervisor: Ms. Chew Moi Tin

ACKNOWLEDGEMENTS

From the bottom of my heart, I would like to express my deep appreciation to Mdm. Chew Moitin who has been directing me in doing this project during the last two semesters. Despite of a long distance in geography when she moved to New Zealand, we still keep tracking and working hard on the project via meetings on Skype. Without instructions, advices and encourages from her, I cannot achieve the project completion as today. Other than that, I have learnt from my supervisor skills in analyzing and solving problems arose in the project implementation.

It is impossible to finish the project without the consistently support from people working in Electronic and Computer Engineering department of RMIT Vietnam (Saigon campus), for providing me necessary facilities to carry out the project. Especially, I would like to give my gratitude to Thanh Chi Pham who is mainly in charge of helping us with electronic devices supply.

EXECUTIVE SUMMARY

The advanced technology has enabled human to create certain possibilities which were impossible in the past. Now the interaction between machines and human could not be easier via voice, touch screen or even hand gesture. There are vast of applications in entertainment field with such state-of-the-art devices like Xbox-One or PlayStation 4. Other than that, these new technologies also facilitate the communication barriers for people with hearing and speaking problems. For this project, it proposes an application that using a 3D sensor to bridge the communication between ordinary people and the deaf/mute. The proposed system, called Vietnamese Sign Language (VSL) Reader targets for Vietnamese community only.

The VSL Reader system is basically comprised of a 3D sensor, Intel Creative Senz3D, a processor and a screen for displaying. It aims to recognize 28 static and 7 dynamic gestures taken from the VSL dictionary. Due to the differences in representation of these two gesture types, the implementation is divided into two parts with different approach.

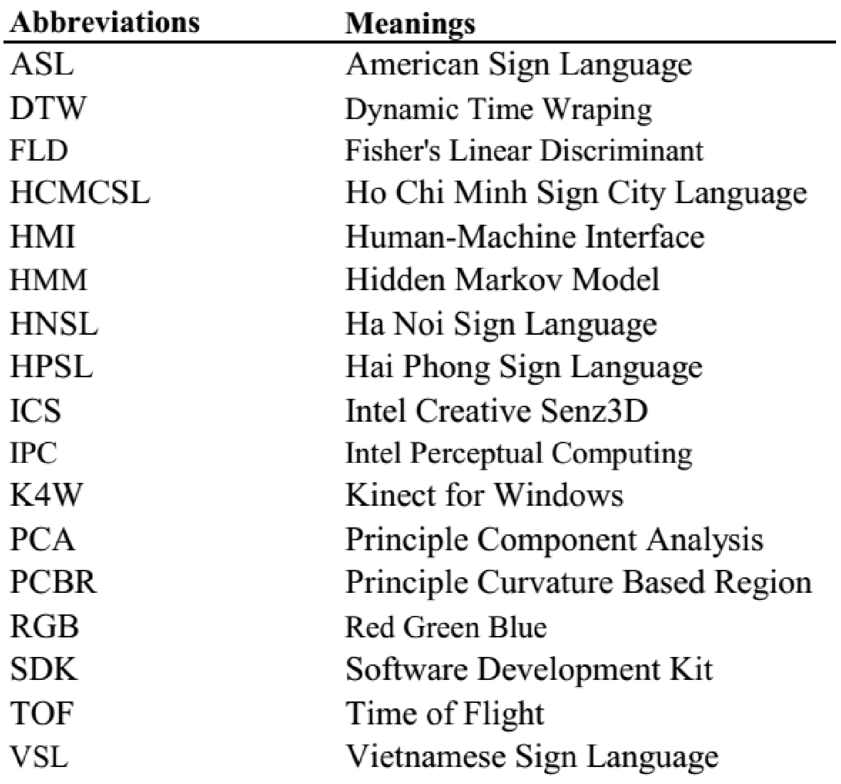
The static gesture is presented by a static hand posture and it mainly describes letters in the alphabet of VSL system. To recognize this gesture type, a sequence of image processing must be carried out. It first extracts the hand location and then eliminate the back ground to enhance the local feature for each gesture. Every sign is classified based on its unique features extracted by Gabor filter. Feature vectors are then projected to Fisher’s Discriminant Analysis to get a reduced dimension for faster comparison in classifier module. The last stage of static recognition implementation is to apply Cosine Metric Distance method for finding the best match from sign database. The proposed technique achieves a good result with 93.89% accuracy and speed of 14 frame per second (fps).

In contrast with static gesture, dynamic gesture is presented by hand movement in different shape and orientation which creates a series of points presented in 2-dimensional format. The used method is based on $1 recognizer algorithm which obtains good accuracy, while it still requires not much computation at computer side. It first locates the index finger tips and does tracking to transfer data for the core processor. This input data, called gesture path is normalized to a uniform format by a sequence of processing including sampling, scaling and translating to new coordinate. The gesture is then classified by a Euclidean distance method. By applying this technique, the system obtains a very good performance with 97.14% of accurate rate and real-time running at 15fps.

CONTENTS

FIGURES AND TABLES

List of abbreviations



Abbreviations	Meanings
ASL	American Sign Language
DTW	Dynamic Time Wraping
FLD	Fisher's Linear Discriminant
HCMCSL	Ho Chi Minh Sign City Language
HMI	Human-Machine Interface
HMM	Hidden Markov Model
HNSL	Ha Noi Sign Language
HPSL	Hai Phong Sign Language
ICS	Intel Creative Senz3D
IPC	Intel Perceptual Computing
K4W	Kinect for Windows
PCA	Principle Component Analysis
PCBR	Principle Curvature Based Region
RGB	Red Green Blue
SDK	Software Development Kit
TOF	Time of Flight
VSL	Vietnamese Sign Language

CHAPTER 1
INTRODUCTION

1.1 Motivation

In Vietnam, there is a large part of disable people with impaired capabilities in hearing and speaking (20% of total disable people, see Figure 1). The community of hearing and speaking impairment often experiences the difficulties in communication with ordinary people. It arises from the evidence that not so many people understand the sign language which is the fundamental method to convey information for the deaf or mute communication [1]. This fact results in a personal motivation to construct a Human-Machine-Interaction system to bridge this communication gap.

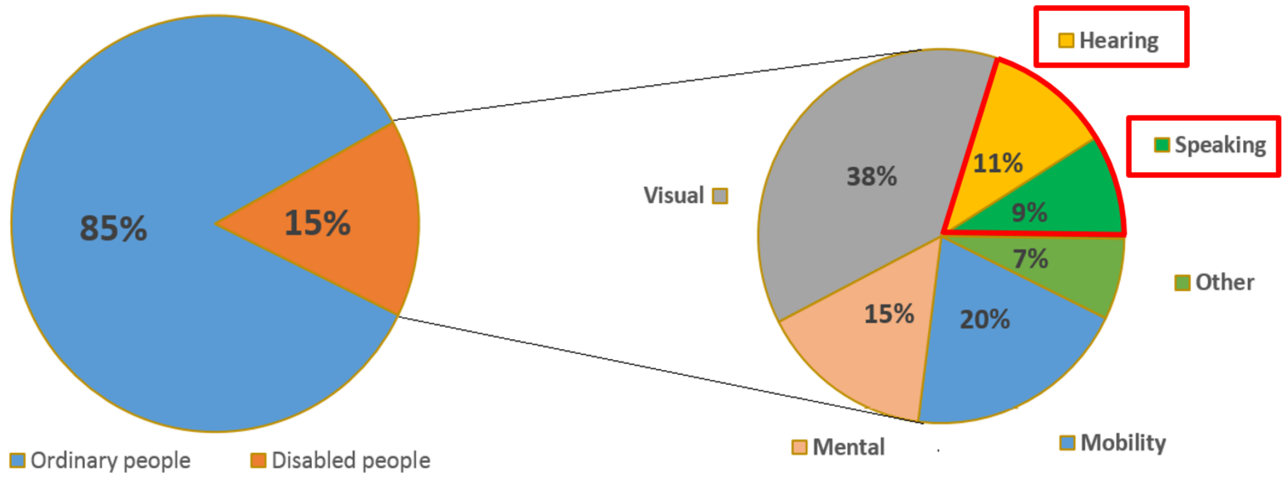


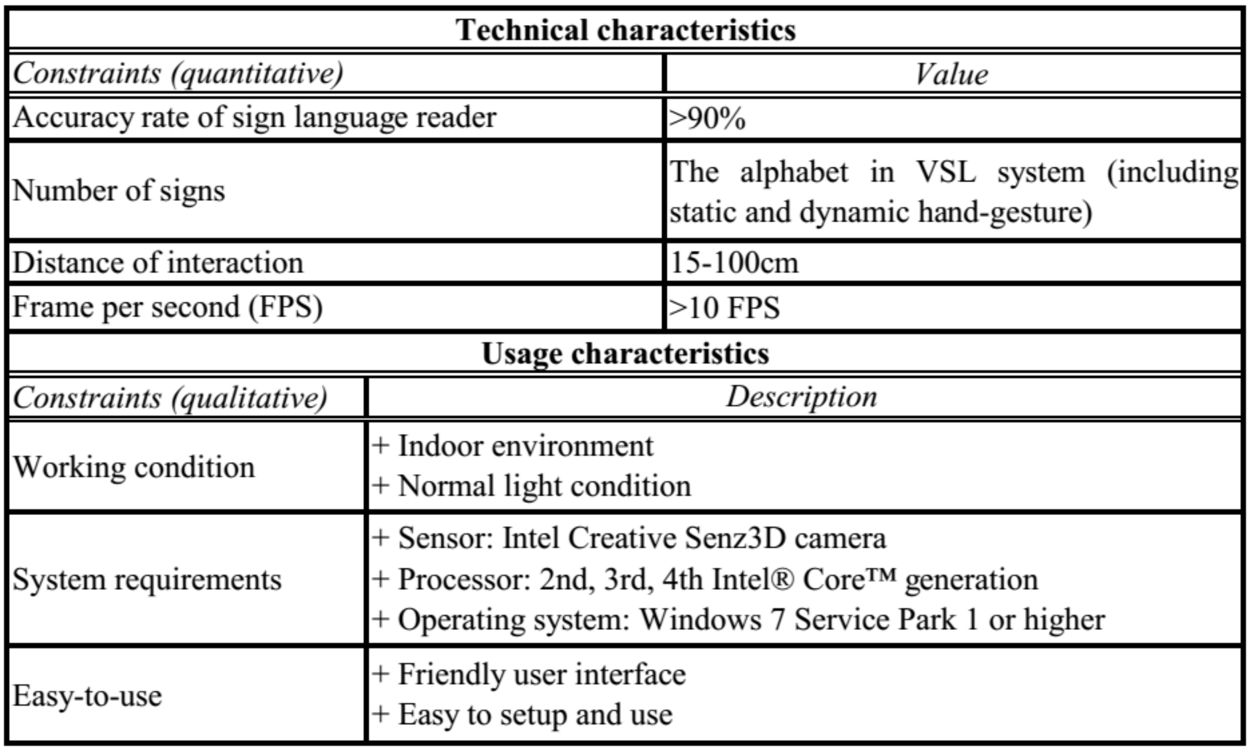
Figure 1: Census of disable people in Vietnam, 2006 (adapted from [2])

With the fast pace of technology development, the interaction between machine and human is not only limited to peripherals like mouse or keyboard, the machine now could recognize on-the-air human hand gestures by a 3D camera. By taking this advantage, this project aims at construct an HMI system that can translate sign language into text format by using a very state-of-the-art 3D camera, Senz3D. And the translator is implemented specified for Vietnamese Sign Language.

1.2 Scope

The objective system is comprised of a 3D camera and an Intel-processor computer. The software core is constructed to recognize signs defined in Alphabet Table of Vietnamese Sign Language system. To get the project done in a duration of six months, number of feasible goals should be set in term of technical and usage characteristics (see Table 1).

Table 1: Project scope



Technical characteristics	
Constraints (quantitative)	*Value*
Accuracy rate of sign language reader	>90%
Number of signs	The alphabet in VSL system (including static and dynamic hand-gesture)
Distance of interaction	15-100cm
Frame per second (FPS)	>10 FPS
Usage characteristics	
Constraints (qualitative)	*Description*
Working condition	+ Indoor environment + Normal light condition
System requirements	+ Sensor: Intel Creative Senz3D camera + Processor: 2nd, 3rd, 4th Intel® Core™ generation + Operating system: Windows 7 Service Park 1 or higher
Easy-to-use	+ Friendly user interface + Easy to setup and use

As for technical characteristic, the translator is required to obtain the accuracy rate of more than 90%. This accuracy is fairly high but it is possible to achieve with the state-of-the-art camera. Because of the heavy workload to cover all signs in VSL, number of signs is limited to the alphabet in the VSL system. The alphabet includes 28 static and 7 dynamic signs that will be descried in detail in in the section 2. The interaction range between end-user and camera is 15-100cm which is an acceptable distance in practice as well as for better accuracy. Since the system is designed for real time application, the operating speed should be over 10 frames per second (FPS, number of acquired pictures per second).

With regards to usage characteristics, the system works best in indoor environment where could provide a normal light condition. It is due to the light sensitivity of the IR sensor from the 3D camera. Beside the working condition, the final product requires to run on specified devices and operating system. The 3D camera type is Intel Creative Senz3D (or Senz3D) that runs compatibly only on latest Intel's processor generations including 2nd, 3rd and 4th Intel® Core™ and on Windows 7 or higher version of the operating system.

1.3 Thesis outline

The main content of this thesis is presented by 7 following chapters:

Chapter 2: gives an introduction of sign langue term in general and the Vietnamese Sign Language in particular. The alphabet representation in VSL system which is classified into static and dynamic signs will be the focus of this chapter.

Chapter 3: presents the literature review of current solutions in building a sign language recognition system.

Chapter 4: gives a general view about the proposed system. The components used for the project are also evaluated and selected appropriately.

Chapter 5: implements static gesture recognition which includes three main modules: pre-process, descriptor and classifier module. The test plan and the obtained results based on applied technique are also given.

Chapter 6: implements dynamic gesture recognition which also includes three modules: input data, normalization and classifier module. The methodology based on $1 recognizer algorithm.

Chapter 7: presents the project management in term of budget plan and timeline. The two main milestones should be delivered are static and dynamic gesture recognition. The time requires to finish this project is two semesters.

Chapter 8: concludes all works have been carried out so far. The intended works to improve the program in the future also are also presented.

CHAPTER 2
VIETNAMESE SIGN LANGUAGE

2.1 Introduction to sign language

Sign language has been developing for a long time and considered as a primary media for conveying information of the deaf and mute community [3]. “Sign language” differs from “spoken language” or “written language” that it is represented by human gestures. Although sign language definition varies throughout the world, it mainly consists of static and dynamic gestures. Both type of these gestures are presented in specific hand postures to illustrate particular information. In dynamic gesture, it requires to add some movements while the hand must be stayed still in the static gesture. Figure 2 shows an example of how to say “I love Vietnam” in sign language. To express this phrase, it requires a combination of two static signs (Figure 2a, 2b) and one dynamic sign (Figure 2c).

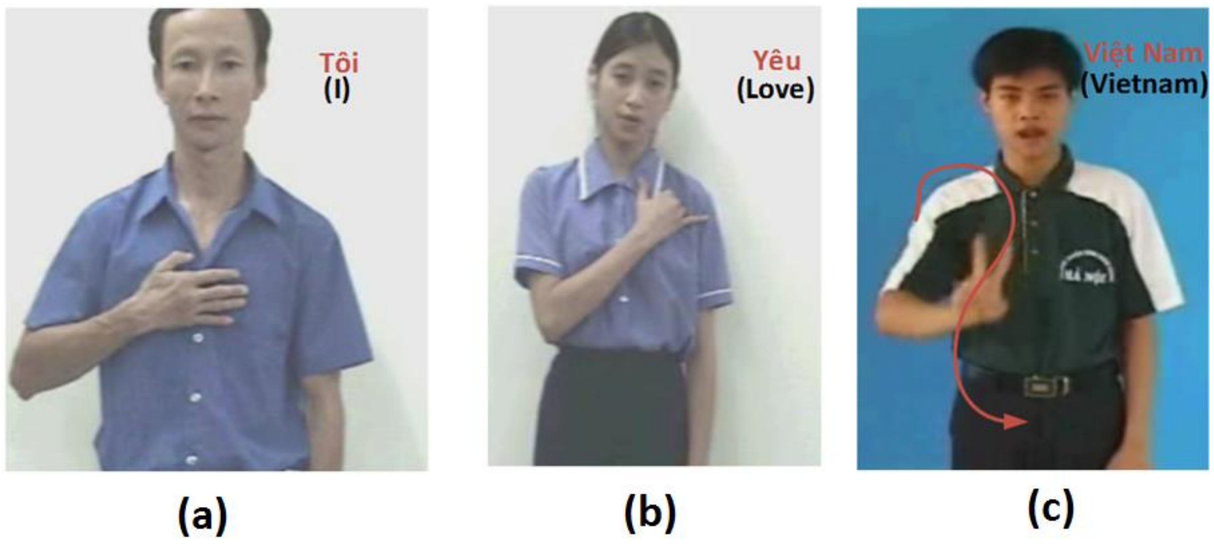


Figure 2: Expression of "I Love Vietnam" in sign language [4]

Vietnam is attempting to develop a national standard sign language since there are vast of definitions from different geographic areas. It can be found that there are three main dictionaries being used: Ho Chi Minh Sign Language (HCMSL), Ha Noi Sign Language (HNSL) and Hai Phong Sign Language (HPSL) [5]. This factor makes confusion for learners with hearing impairment. Thanks to that, Ho Chi Minh City University of Education has implemented electronic dictionary that including most sign language definitions over the country [6]. For the most part of this project, the signs are taken from this electronic dictionary.

2.2 Alphabet, markers and tones in Vietnamese sign language

Figure 3 shows twenty-four based letters and three loan letters: J, W and Z in VSL system. In this alphabet table, only J and K require to add some movements therefore they are considered as dynamic gestures. The remaining letters are presented in fixed-position hand postures and thus they are recognized as static gestures.

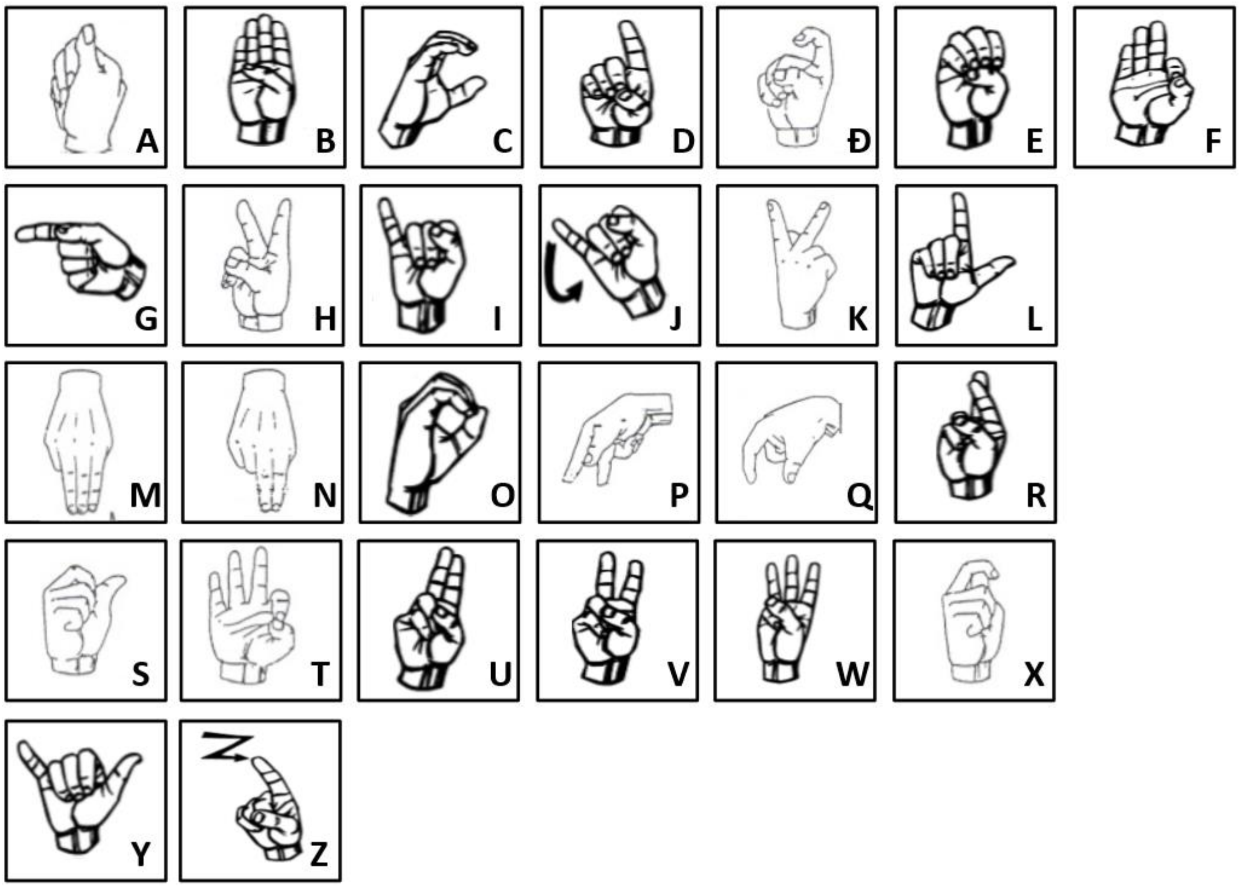


Figure 3: Alphabet hand sign representation in VSL system [4]

Apart from twenty-four base letters, in Vietnamese writing system, there are six extended vowels: Ă, Â, Ê, Ô, Ơ, Ư which are created by combining four letters: A, E, O, U with three different markers (˘, ˆ, ʾ) as described in Figure 4a. In addition to markers, five special symbols called tones are also presented in Figure 4b. Depending on the tone added, it gives different meanings for a word. For example, the word "ma" has specific meaning with particular tone as below:

- "ma" without tone = *"ma"* means *"ghost"*
- "ma" + rising tone = *"má"* means *"mother"*
- "ma" + falling tone = *"mà"* means *"but"*
- "ma" + falling rising tone = *"mả"* means *"tomb"*
- "ma" + high rising tone = *"mã"* means *"horse"*
- "ma" + low constricted tone = *"mạ"* means *"rice seeding"*

Based on the illustration for makers and tones described in Figure 4, it can be seen that there are five dynamic gestures representing maker "˘" and rising, falling, falling rising, high rising tone. The maker "ˆ", "ʾ" and low constricted tone are considered as static gestures.

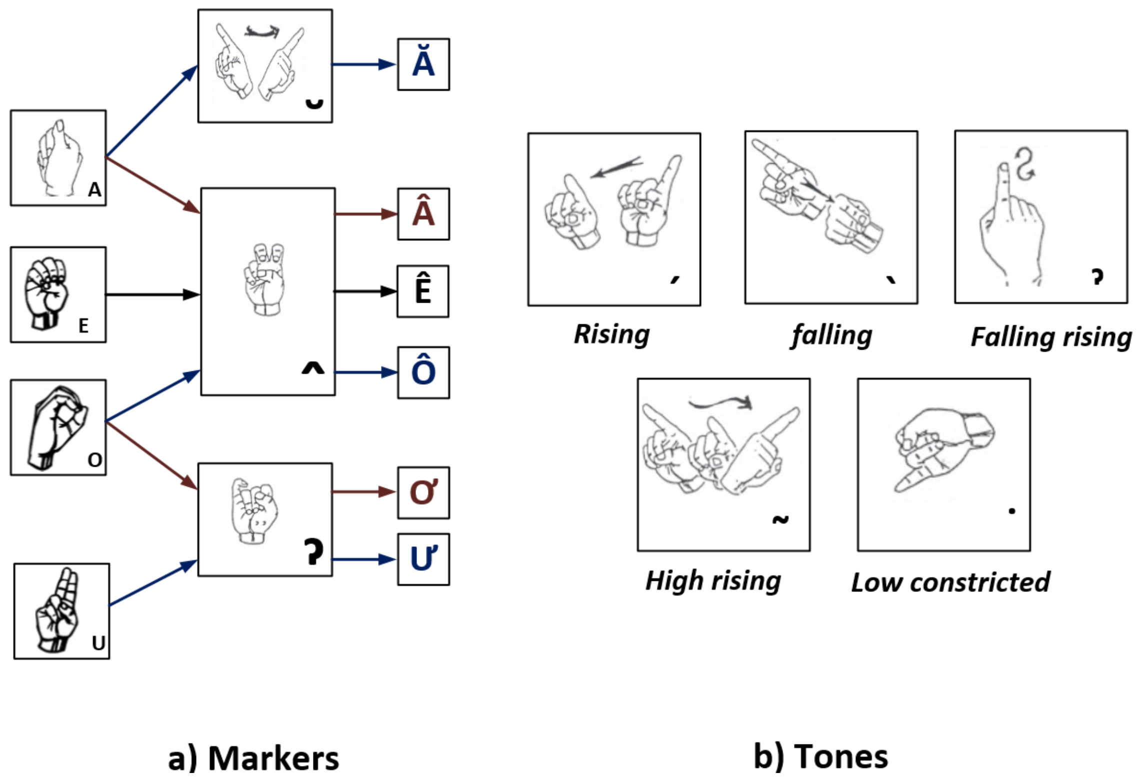


Figure 4: Six extended vowels with markers and tones in VSL [4]

As remarked in the scope, the focus of project is to identify twenty-seven signs in Figure 3, three markers and five tones in Figure 4. Consequentially, 35 signs in total including 28 static signs and 7 dynamic signs require to be recognized.

CHAPTER 3
LITERATURE REVIEW

In building a gesture recognition system, there are numerous techniques being applied, but it is mainly characterized in two approaches: glove-based and vision-based method.

3.1 Glove-based sign language recognition

Glove-based method refers to the use of an electronic gloves to analyze the hand's posture in such human-machine applications [7]. It could measure finger joint angles, detect bending fingers or thumbs-up posture, and these captured signal are sent back to computer for processing.

Gesture recognition with electronic gloves has been researching for a long time. The first electronic gloves appeared in the late 1970's [8]. With advanced technology, the electronic gloves are currently developed with high accuracy for hand gesture detection. Though glove-based technique helps reducing system computational burden, it finds difficult to popularize the product due to its high price and cumbersome to users [9].

Figure 5: Two electronic gloves on the market [10]

Figure 5a shows one of electronic gloves of 5DT with 14 sensors placed on five fingers and $5,495 of the price [10]. Figure 5b shows the CyberGloves III product with 22 robust sensors for capturing the complexities of hand and finger movement which costs $17,795 [11]. Both type of gloves are developed for sign language recognition with good results [12, 13], but they are still not approaches to popularize for practical use due to not only the expensiveness but also the encumbrance and easy-to-be-damaged.

3.2 Vision-based sign language recognition

In vision-based method, the acquiring frames from camera will be analyzed and processed at computer side. Each sign is described by a specific hand posture will be used to compare with database based on their unique features. From that, the system can recognize what gesture being presented at input. The input data could be acquired from two types of camera: 2D and 3D. The ability of providing depth information makes 3D camera differ from 2D one.

Using 2D camera to extract skin color for gesture recognition is one of the common methods [14, 15] (see Figure 6a). To eliminate the background and take the interested hand region, it requires users to wear a long sleeves. Hence, it is difficult to apply this method for practical use. To enhance the quality for extracting hand region, using color gloves has been applied as shown in Figure 6b. But the performance will be affected in the case that the background has same color with the glove. When the hand is located, numerous of algorithms could be applied to extract the features and do recognizing. By using Hidden Markov Model (HMM) algorithm, Starner et al. achieved the accuracy rate of 92% in recognizing 40 different signs in ASL [16]. Rekha [17] used Principle Curvature Based Region (PCBR) detector to extract the shape, finger features with 91.3% detecting correctly. The well-known object recognition algorithm: Principle Component Analysis (PCA) also has been used for taking out features like position, shape, and orientation of the finger. Lamar [18] proved that PCA helped sign language recognition system obtain 93% of accuracy rate. Although using a 2D camera could achieve a fairly good result for recognizing hand gestures, it is very dependent on illumination condition and background color which are common problems of using 2D camera.

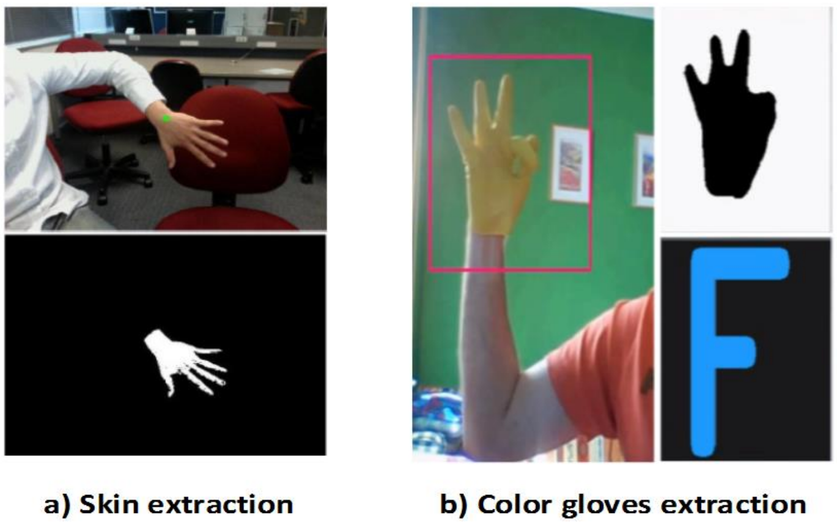


Figure 6: Technique applied in extracting hand region

In order to eliminate limitations in gesture recognition processing including light sensitivity, background noise; using devices which are able to provide 3D data or depth data (see Figure 7) would be a good approach. Vogler et al. [19] constructed an ASL recognition system that used 3D data as the input parallel HMM-based method as the processing algorithm. They successfully obtained the results for 22 ASL signs with the best accuracy of 95.83%. Lee et al. [20] designed a system for recognizing static gesture for Taiwan sign language that used eight-camera 3D opto-electronic and neuron network. This system achieved an average accuracy of 96.58% on 20 hand gestures in changing background. The typical limitations of recent approach using 3D data are the costly 3D camera system and small number of gestures detected due to heavy process.

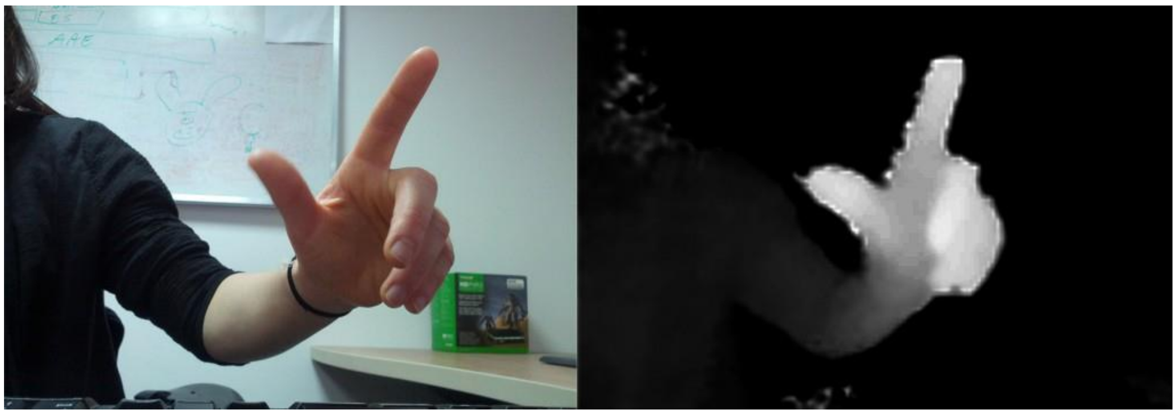


Figure 7: Color and depth images

Fortunately, the advances in 3D cameras have been widely applied especially in gaming with low cost. Typical examples are Microsoft Kinect for Xbox 360 ($100) and Asus Xtion Pro ($120) which could provide RGB image and depth image as shown in Figure 7 [21, 22]. With better accuracy rate and robustness against variant background color as well as light conditions; this project aims to build a Vietnamese Sign Language recognition system by using 3D sensor.

CHAPTER 4
SYSTEM OVERVIEW

4.1 System description

To bridge to communication between ordinary people and the deaf/mute, the proposed system is to provide a 3D sensor, a processor working as a brain to analyze gesture and a screen for displaying translated text as shown in Figure 8. The 3D sensor is responsible for capturing human gestures and transferring raw data to the processor for processing. The input gesture basically is compared with the database to find the most similar one, and the corresponding text will be shown on the screen to help ordinary understand what the deaf/mute is saying.

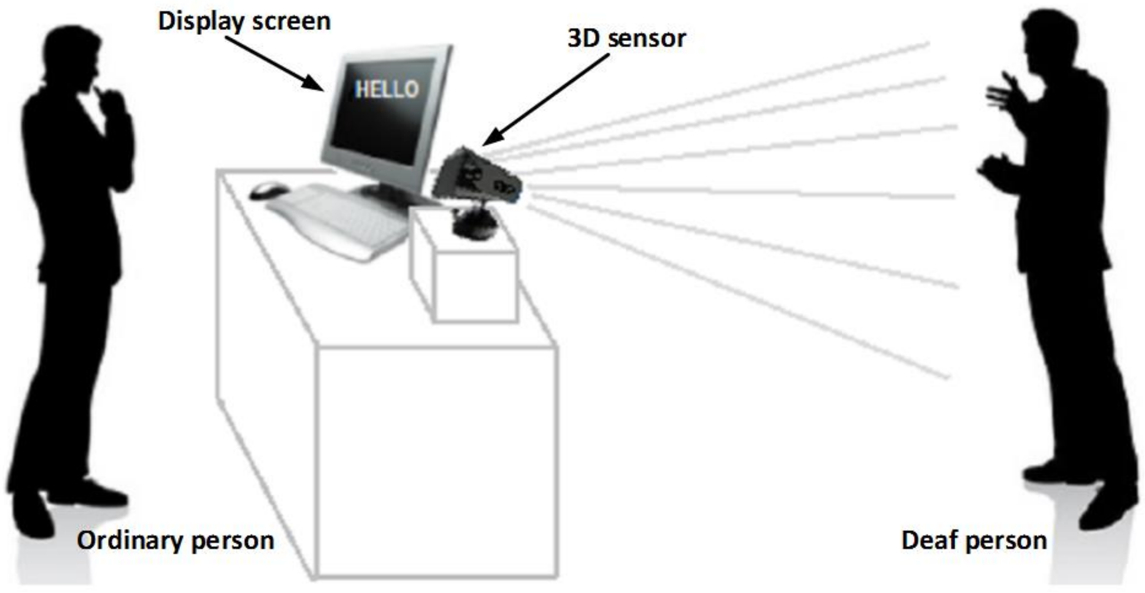


Figure 8: System description

The following sections are to analyze required components in term of hardware and software required for the project implementation.

4.2 Hardware

In term of hardware, the overall system basically requires one 3D sensor and one central processor. Because the acquired data from the 3D sensor includes not only RGB image, it also provides depth information, the central processor should be strong for better performance. In this project, a laptop with Intel processor Core i5 is used for the implementation.

Besides, 3D sensor is considered as a most important component in this project, so choosing a suitable one for the project implementation is critical. Kinect is the most well-known one available on the market. When Kinect for Xbox 360 was launched in 2010, Kinect has been revolutionized the game market with as well as depth sensor that could construct 3D space using Time of Flight (TOF) technology [23]. In order to make their 3D sensor commercially not only for game, Microsoft has released the Kinect for Windows which is particularly designed for developers on Windows Operating System. There are two products similar to Kinect which came to market at the same time including Asus XTionPRO and PrimeSense PSDK (see Figure 9). Yet Kinect is still outstanding due to huge support from developer community [24].

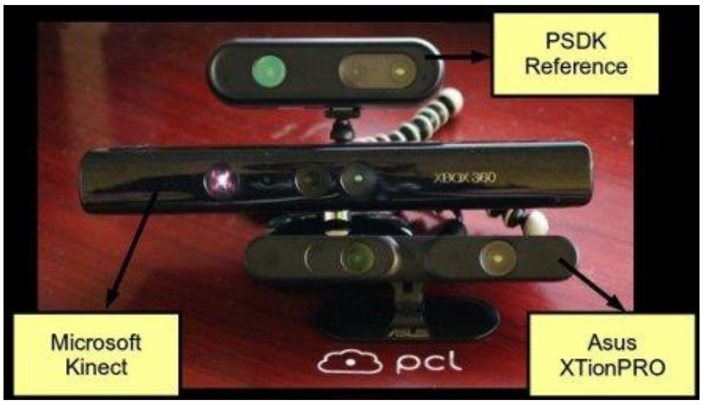


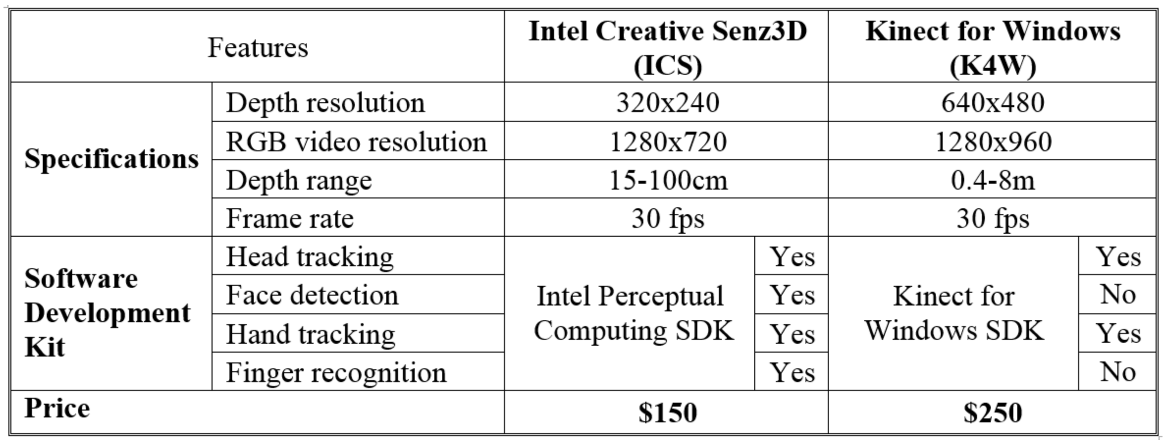
Figure 9: Available 3D sensor on market

Although it is very new on the market, Intel Creative Senz3D still gives an impressiveness due to its design particularly for short range interaction (see Figure 10). This sensor was introduced in June 2013 with the aim of creating new experience in human-machine interaction [25].

Figure 10: Kinect for Windows versus Intel Creative Senz3D

After a carry fully evaluation, the best 3D sensors used for this project are Kinect for Windows and Intel Creative Senz3D. To select the suitable one, Table 2 has been built up for comparison.

Table 2: Intel Creative Senz3D versus Kinect for Windows [26-29]



Features		Intel Creative Senz3D (ICS)		Kinect for Windows (K4W)	
Specifications	Depth resolution	320x240		640x480	
	RGB video resolution	1280x720		1280x960	
	Depth range	15-100cm		0.4-8m	
	Frame rate	30 fps		30 fps	
Software Development Kit	Head tracking	Intel Perceptual Computing SDK	Yes	Kinect for Windows SDK	Yes
	Face detection		Yes		No
	Hand tracking		Yes		Yes
	Finger recognition		Yes		No
Price		**$150**		**$250**	

As presented in Table 2, depth resolution and depth range are significantly different between the two cameras. The Senz3D support lower depth resolution and shorter depth range in compared to K4W due to the fact that ICS is designed for short range interaction. Regarding Software Development Kit (SDK), only the four features are listed including head tracking, face detection, hand tracking and finger recognition which provide meaningful information for sign language recognition. With respect to this issue, Intel Perceptual Computing SDK supports its sensors better than Kinect for Windows SDK. Additionally, ICS is much cheaper than K4W. Consequently, ICS advances over K4W for the sign language translator application which only requires a short distance from sensor to user, and also needs detailed information of the hand, fingers and even the face for future improvement.

4.3 Software

It cannot make hardware work without software, hence choosing a suitable software could contribute significantly for the final product. For this project, there are three types of software needed: driver for Senz3D, the C++ compiler, and image processing libraries. The driver coming along with Senz3D is Intel Perceptual Computing SDK (IPC SDK) which provides not only driver but also a small library of basic functionalities for accessing the device. Visual Studio 2012 compiler is used due to its well-support from developer community. We also need external library for image processing including OpenCV and CLAPACK library. These two libraries are open source for real time computer vision application [30, 31]. While OpenCV library is well-known for image processing, the CLAPACK is built for particular purpose in Eigenvectors calculation which will be mentioned in section 5.4.2. The software pack including four items is shown in Figure 11.

Figure 11: Software package used in this project

Because the application will be implemented on the provided interface or driver for Senz3D, the IPC SDK architecture is discussed in detail in this section.

IPC SDK is a tool for developers to create applications for Intel Creative Senz3D camera. The IPC SDK provides driver for the communication between ICS and computer, and it only supports 2nd, 3rd and 4th generation Intel core processors. As presented in Figure 12, the SDK architecture lists all layers from top (application) to bottom (core functionalities). The explanation about each layer's operating principle will be presented from bottom to top as follows.

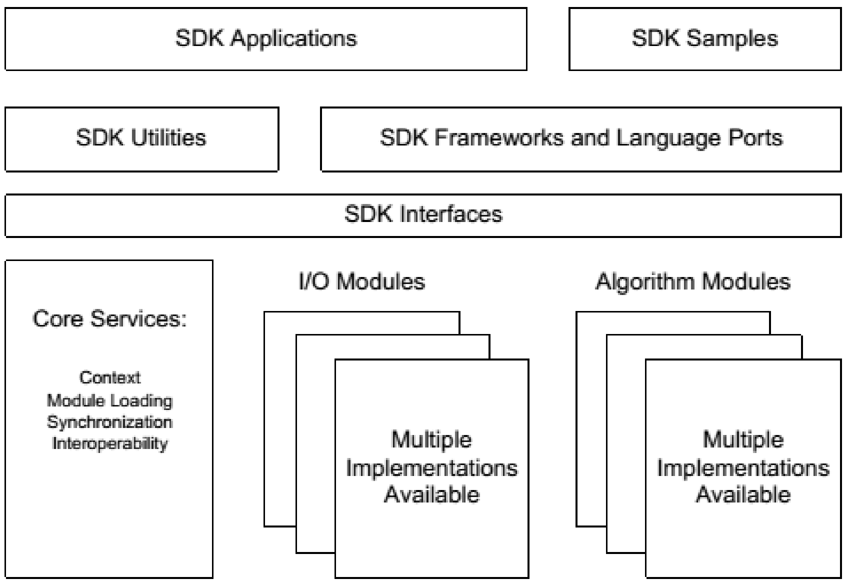


Figure 12: Intel Perceptual Computing SDK architecture [32]

The main functionalities of the SDK are located in I/O and Algorithm modules. The I/O modules retrieve sensor data including RGB, depth image and audio stream from the input and sink that data to output devices after being processed. In this project, RGB and depth image stream are interest data for the application implementation. The algorithm modules include some patterns detection such as: face detection, hand detection, voice recognition that enable human-computer interactions. Multiple implementations at this layer may occur at the same time independently or link to each other, thus the core services appear hear to manage the task of module loading, synchronization and interoperability.

The SDK interfaces are provided for the access to functionalities in I/O and algorithm module of the upper layer without be concerned about the underlying implementations. From this layer, developers could build a wrapper of basic functions for regular usage themselves for their applications.

On the top of SDK interfaces, the SDK provides classes of functionalities for developer to facilitate application programing including SDK Utilities and SDK Frameworks and Language Ports. These classes include raw functionalities such as sensor data (depth, RGB image, audio stream) retrieving or basic gestures (thumbs up, big five or waving gesture) recognition. The interface is originally built in C++ and also ported to C#. The C++ programming language is selected for implementation in this project because of the flexibility and better performance compared with C# in term of processing speed.

At the top of the SDK architecture, there are some samples provided by the SDK for getting familiar in using the library. Additionally, the developers now could do programing for their application at this layer which links to all below levels of the SDK architecture.

CHAPTER 5
STATIC GESTURE RECOGNITION

5.1 Methodology

Figure 13 shows the software flow of the static gesture recognition designed for two operation phases: training and recognizing phase. These two phases run separately; the training phase is considered as an offline task, only active when users want to add new gesture to sign database, while the recognizing phase operates most of the time to detect hand gesture. The system core is comprised of three main modules: Pre-process, Descriptor and Classifier module. In the training phase, a set of training images will be pre-processed and descripted to save to sign database with its corresponding text. While in the recognizing phase, it requires one more step is to classify the input gesture and select the most similar one from database. The corresponding "**Text**" of the most similar gesture referred from database will be displayed at output.

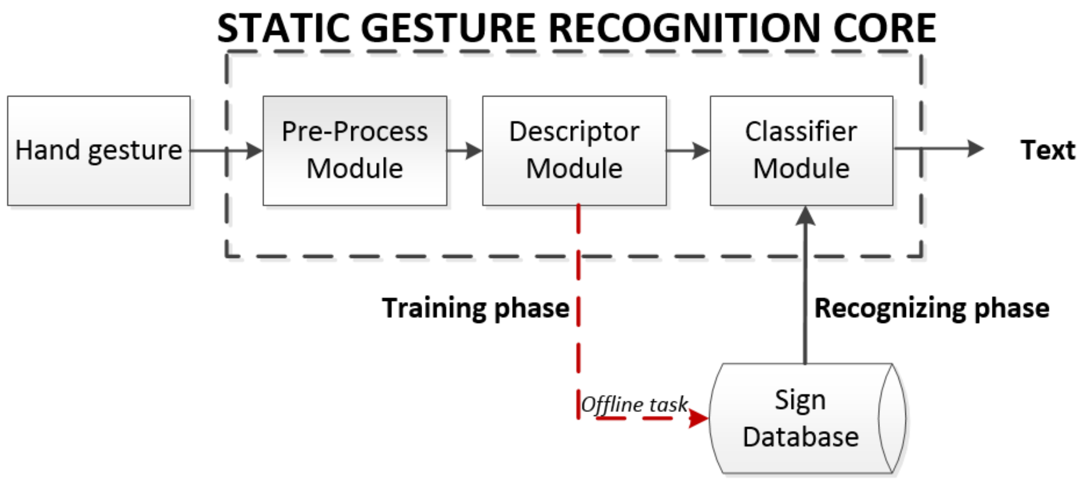


Figure 13: Sign language recognition flow chart simplified

The brief description of each module is presented as follows:

- *Hand gesture*: the static hand posture will be grabbed by Senz3D. With support of hand tracking from IPC SDK, the hand is easily located and extracted interested region for further process.
- *Sign database*: is an xml file that contains feature vector of each sign with its corresponding text. The sign database is referred to get the most similar gesture in recognizing mode.
- *Pre-process module*: segments only hand region from the background by using depth information. In order to enhance the local hand feature and reduce illumination effect, a sequence of image processing is applied.
- *Descriptor module*: describes each gesture in a way that it is unique and different from other signs. The feature on each gesture data will be extracted by using Gabor filter. For the dimension reduction, the Fisher Linear Discriminant (FLD) is applied.
- *Classifier module*: is responsible for recognizing the input gesture by comparing with signs stored in database. The cosine distance method is used to find the most similar sign from database and display its **"Text"** definition at output.

5.2 RGB and depth data of Senz3D

Not like a normal single camera, Senz3D is equipped with not only a 2D sensor to record RGB image, but also an infrared camera. The chip built inside Senz3D could generate depth map information based on stream data acquired from these two sensors. The RGB image has a maximum resolution of 1280x720 while the depth image resolution is limited to 320x240 as illustrated in Figure 14. Regarding streaming speed, both type of data can run at speed of 30 frames per second. The light auto-balance function feature built in the Senz3D helps reducing illumination effect.

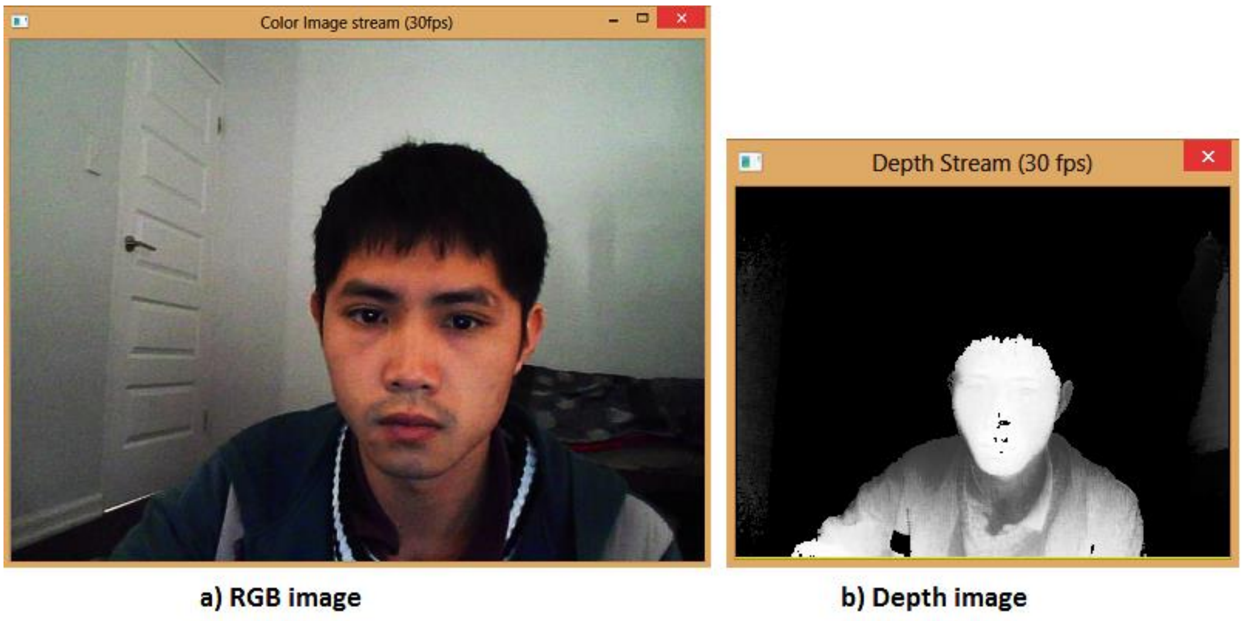


Figure 14: RGB and Depth image

With respect to depth data as shown in Figure 14b, each pixel represents distance to camera. These pixels are presented in 16 bit format. As illustrated in Figure 14b, pixels describing closer distance are brighter than the further ones. Therefore, we could see clearly user's body modeled in depth when he/she sit in front of the camera. This 3D sensor only constructs depth map within the distance of 1.5 meters, and with further distance, Senz3D considers it as infinite, hence black color dominates at background side in Figure 14b.

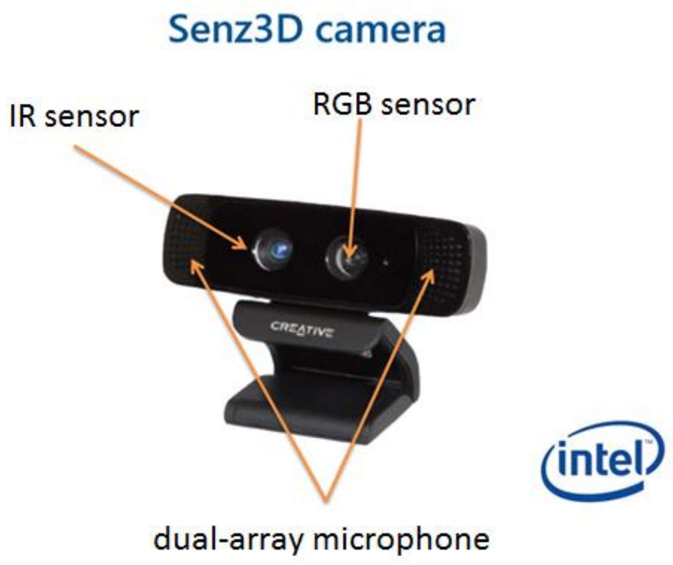


Figure 15: Brief look of Senz3D camera

As shown in Figure 15, two main sensors infrared (IR) and RGB. The two sensors are placed in different positions of the camera. Moreover, they also provides different resolution as mentioned previously. These factors result in a non-aligned grabbed images. Therefore, it requires to calibrate Senz3D before overlapping the two data acquired from the two sensors. Overlapping is necessary because we need to subtract the background based on depth and RGB data. Fortunately, IPC SDK provides two functions to do alignment or mapping, called *UVMap* and *ProjectionMap*. Figure 16 shows the mapping result when applying these two methods.

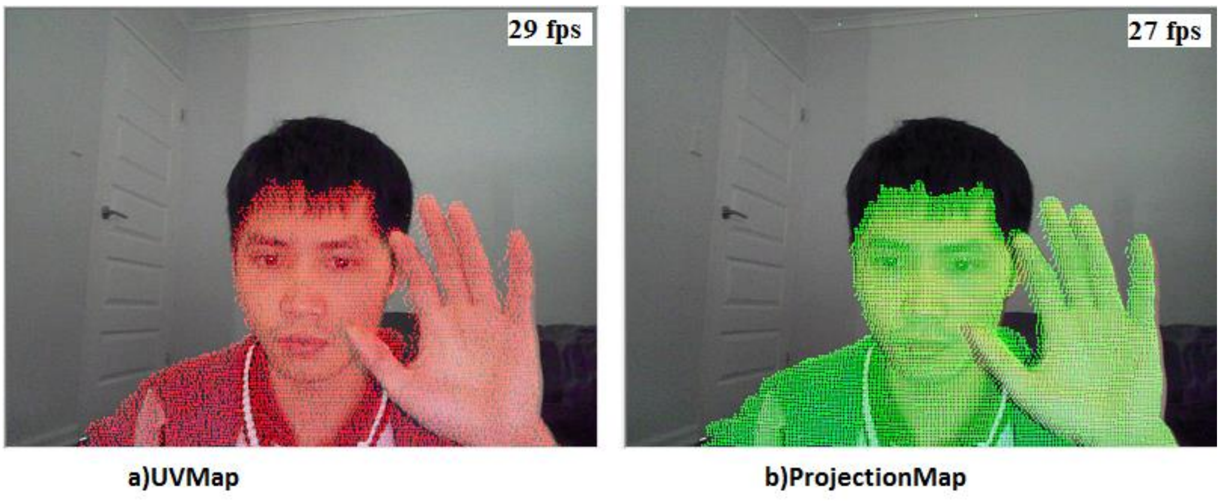


Figure 16: Alignment techniques supported by IPC SDK

In Figure 16, the color dots (red and green) presents pixels taken from the depth image. Because RGB sensor and depth sensor have different resolutions, the color dots or depth pixels are placed on RGB image discretely. In these two maps, the depth range for mapping is defined from 0÷1 meter, so that there are no color dots on the out of this region (the wall on the left is still observable in Figure 14b, but it is eliminated in Figure 16b). After experiment, we found that: although using UVMap method is faster, the result of a pixel mapped from depth image to RGB image by using ProjectionMap method is more accurate and stable. From the fact that the project requires to have stable hand tracking and correct hand segmentation result, *ProjectionMap* method is selected for the implementation of the Pre-process module, where the hand sign is extracted.

5.3 Pre-process module

Pre-process is responsible for normalizing the raw input data by the following two steps:

- Hand segmentation: it first locates the hand region, and then extract only interested hand gesture based on two streaming data from two sensors: IR and RGB.
- Hand normalization: the segmented hand is scale to a fixed square to make the task of recognition easier. To enhance the local hand features, a sequence of image processing is employed for reducing illumination effects.

5.3.1 Hand segmentation

As mentioned in section 4.3, the IPC SDK implements some algorithm modules for pattern detection. In such sign language recognition, hand detection is the most interested one. Hence, we mainly focus hand detection module. In this module, the SDK supports functionalities to do hand tracking, finger tips detection and some basic gestures on depth image stream.

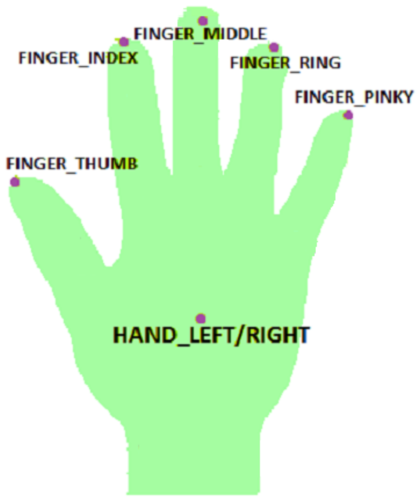


Figure 17: BIG5 gesture representation

When **"BIG 5"** sign is recognized by hand detection module as shown in Figure 17, the palm center and all the finger tips' position are detected and tracked when the hand moves. These information are meaningful for hand segmentation. Based on the geometric nodes of hand, the hand region could be estimated easily.

To extract the hand sign, we first do **"BIG 5"** sign detection to find the hand region. The **"BIG 5**" sign is used thanks to its biggest region so that region is also used for other hand shapes at particular distance range. When **"BIG 5"** sign is detected, we use middle finger tip and palm center position to estimate the hand region in Cartesian coordinate as presented in Figure 18.With other hand shapes, fingers' position may change but the palm center keeps being in fixed location referencing to finger geometric nodes. Therefore, the blue square indicating the hand region in Figure 18 will ensure covering entire the hand when it alters posture and moves.

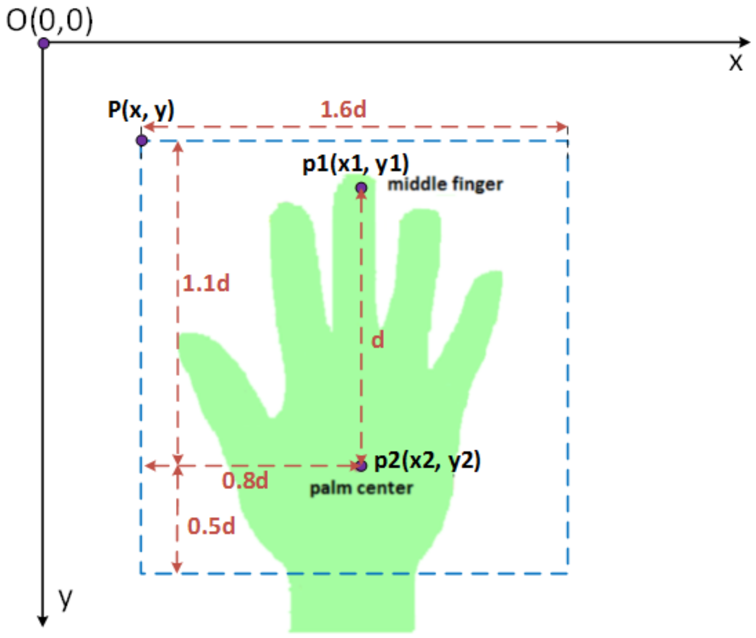


Figure 18: Hand location estimation based on BIG5 gesture

Figure 18 shows how the hand region in image coordinate (original point is located on the top left of the image) is estimated. Based on the vertical distance (d = y2 –y1) between middle finger tip (p1) and palm center (p2), the hand region could be determined as illustrated in Figure 18. From experiments, the optimized interested region which covers all the fingers is defined as a square with the origin at P(x, y) and the dimension size is 1.6d (x = x2 – 0.8d, y = y2 – 1.1d). The result when applying the proposed hand region estimation method is described in Figure 19a which is for depth image. The corresponding hand region on RGB image is mapped from depth image. Up to now, for usage, users first need to shows BIG5 sign in detection range (see Table 1) for the program calculate hand region. After that, the Vietnamese Sign Language Reader program is activated.

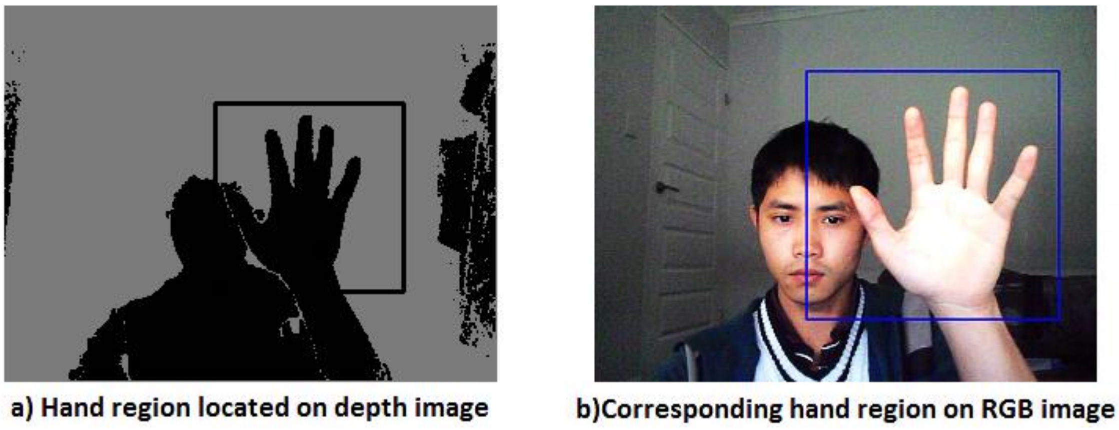

a) Hand region located on depth image
b)Corresponding hand region on RGB image

Figure 19: Hand region estimation on depth and RGB image

When hand region is located, ProjectMap method will be used to extract the hand in the RGB image. The mapped hand region is acquired as presented in Figure 20a with the discrete black dots on the white background. The dots lied near each other are then filled by using `cvMorphologyEx()` function in OpenCV library to create the mask for hand region (see Figure 20b).

Hand segmentation result shown in Figure 20d is obtained by lapping the mask over the RGB hand region.

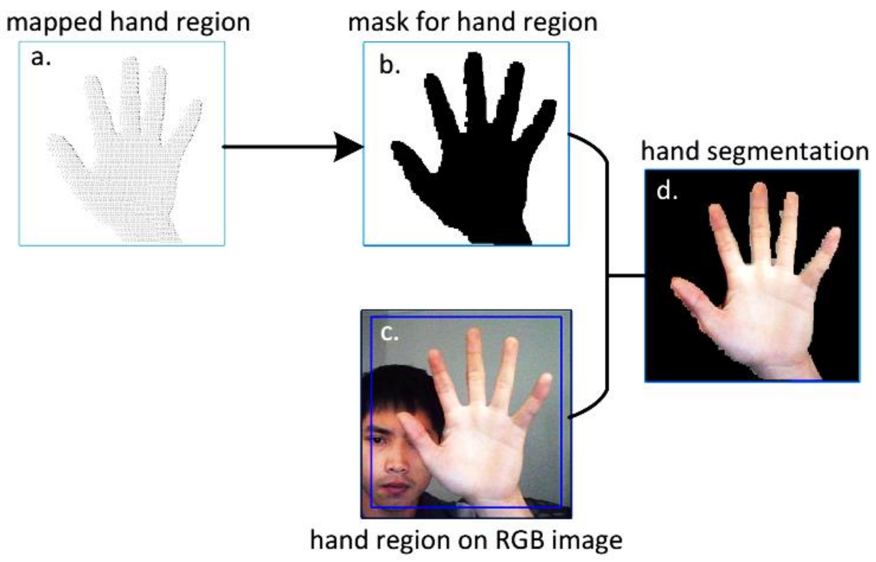


Figure 20: Steps in segmenting hand region

5.3.2 Hand normalization

The segmented hand image is first scaled to reduce computation cost and speed up the program. The size is empirically chosen as 64x64 pixels. Next, a series of image processing suggested by Tan and Triggs are applied for illumination normalization, including: gamma adjustment, Gaussian filter and contrast balance [33].

Gamma adjustment. This is a non-linear transformation that replacing the pixel intensity P by P^γ, with γ varying from 0÷1. This method enhances the local textures level of dark areas where normally cannot be obtained. But the bigger γ is, the more noise is amplified. As suggested in [33], the range of [0÷0.5] for γ is a good compromise. The chosen value of γ in the project is 0.4 by practical experiment.

Gaussian Filter. Although the gamma adjustment can enhance feature in dark regions, it cannot remove all the shadow effects which is induced by the surface structure of the hand. Gaussian filter will increase the contrast in the shadowed regions. The method is easy to obtain by using the function: `cvFilter2D()` in OpenCV library.

Contrast balance. This is the last one in the chain of image processing in pre-process module, which will standardize the overall contrast as well as intensity variation. Its contributions to the pre-process module is to eliminate the extreme values resulting from the bright regions and noise at the edge of objects. There are two steps for this:

$$P(x,y) \leftarrow \frac{P(\mathrm{x},\mathrm{y})}{(\mathrm{mean}(|P(\mathrm{x},\mathrm{y})|^{\alpha}))^{1/\alpha}}$$

$$P(x,y) \leftarrow \frac{P(\mathrm{x},\mathrm{y})}{(\mathrm{mean}(\min(\tau,|P(\mathrm{x},\mathrm{y})|^{\alpha})))^{1/\alpha}}$$

Where P(x, y) represents the value of the pixel at (x, y) coordinate in the image. α and τ are compressive and threshold component respectively that used for decreasing the effect of extreme values. In this normalization stage, we use: $\alpha = 0.3$ and $\tau = 13.3$ empirically.

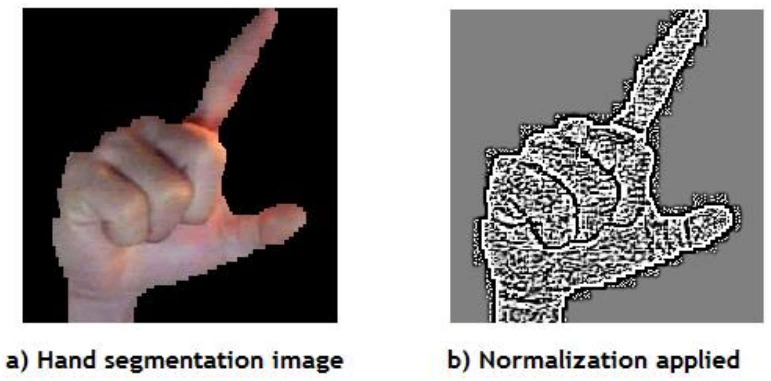


Figure 21: Hand gesture normalization

Figure 21 describes the segmented hand image and normalization applied by the Pre-process module. The features of each sign could be visually observed which are mainly the shapes, pixel's intensity and local textures as illustrated in Figure 21b. The outcome of pre-process module is to make hand gesture more discriminative which is very important for the next module in the project flow, descriptor module.

5.4 Descriptor module

Descriptor module is responsible for descripting gestures by extracting features in three steps. Firstly, the image after processed by Pre-process module is convolved with Gabor filters to produce Gabor-coded images. Secondly, these Gabor-coded images are concatenated to form one dimension array of features. Finally, the FDA algorithm is applied for dimension reduction since the number of elements in feature array is too big.

5.4.1 The Gabor filter

The Gabor filter is good approach to acquire features in different scales as well as orientations of an input image. This method obtains good results in face and hand gesture recognition due to its providing discriminant features ability [34-36]. The Gabor filter equation used in the module is formed by multiplying a Gaussian envelope with a complex exponential function (see equation (1)).

$$\psi_{\mu,\nu}(z) = \frac{\left\|k_{\mu,\nu}\right\|^2}{\sigma^2} e^{\frac{-\left\|k_{\mu,\nu}\right\|^2 \left\|z\right\|^2}{2\sigma^2}} \left[e^{ik_{\mu,\nu^z}} - e^{\frac{-\sigma^2}{2}} \right] \qquad (1)$$

Where:

- μ : The orientation of the filter
- ν : The scale of the filter
- z: The pixel coordinate, z = (x, y)
- $k_{\mu,\nu}$: The wave vector, $k_{\mu,\nu} = k_\nu e^{i\phi_\mu} = \frac{k_{\max}}{\lambda^\nu} e^{i\phi_\mu}$; λ is the wavelength which is the spacing between two Gabor filters, $\phi_\mu = \frac{\pi\mu}{8}$

- $\|\square\|$: The norm operator
- σ: The ratio of Gaussian window width to wave length.

Figure 22 shows an example of how Gabor filter is visually presented in varied orientation, scale and wavelength which are three main parameters used for feature extraction.

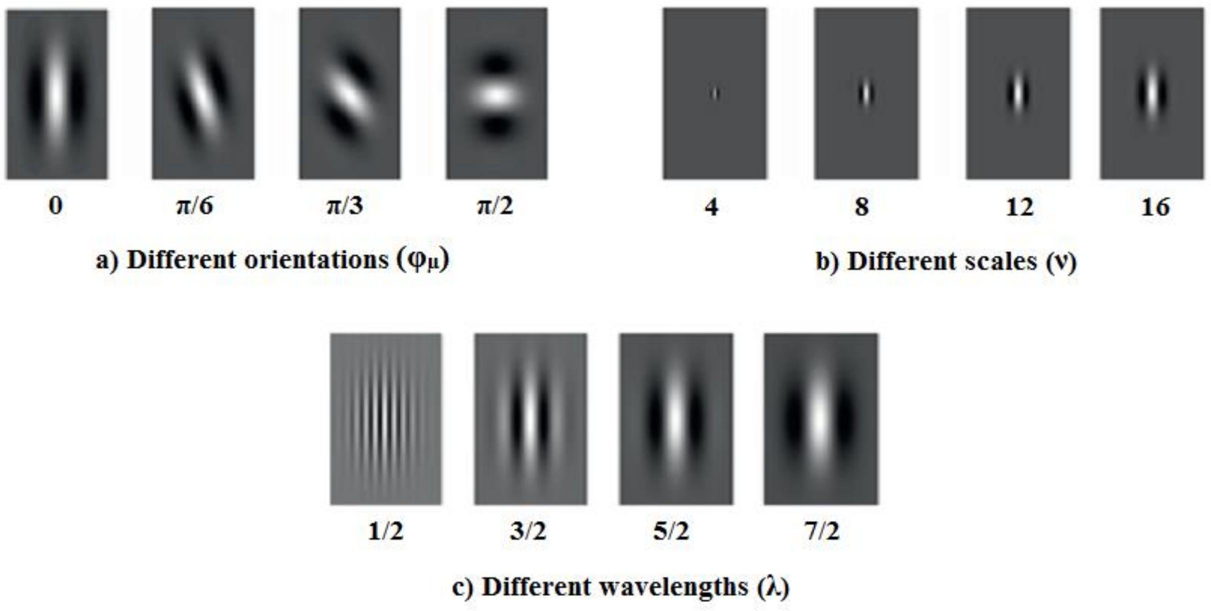


Figure 22: Example of Gabor filter with four different orientations, scales and wavelengths

In this project, the only two parameters: orientation and scale are employed due to its providing acceptable discriminant features at a specific wavelength value. Besides, the computation for processing image is also reduced significantly. As suggested in [37], the used Gabor filter's parameters are defined as: $k_{\max} = \pi/2$, $\sigma = 2\pi$, $\lambda = \sqrt{2}$, eight orientations $\phi_{\mu} = \pi\mu/8$ with $\mu = 0 \div 7$ and five scales $\nu = 0 \div 4$. Total of 40 Gabor filters are generated as illustrated in Figure 23.

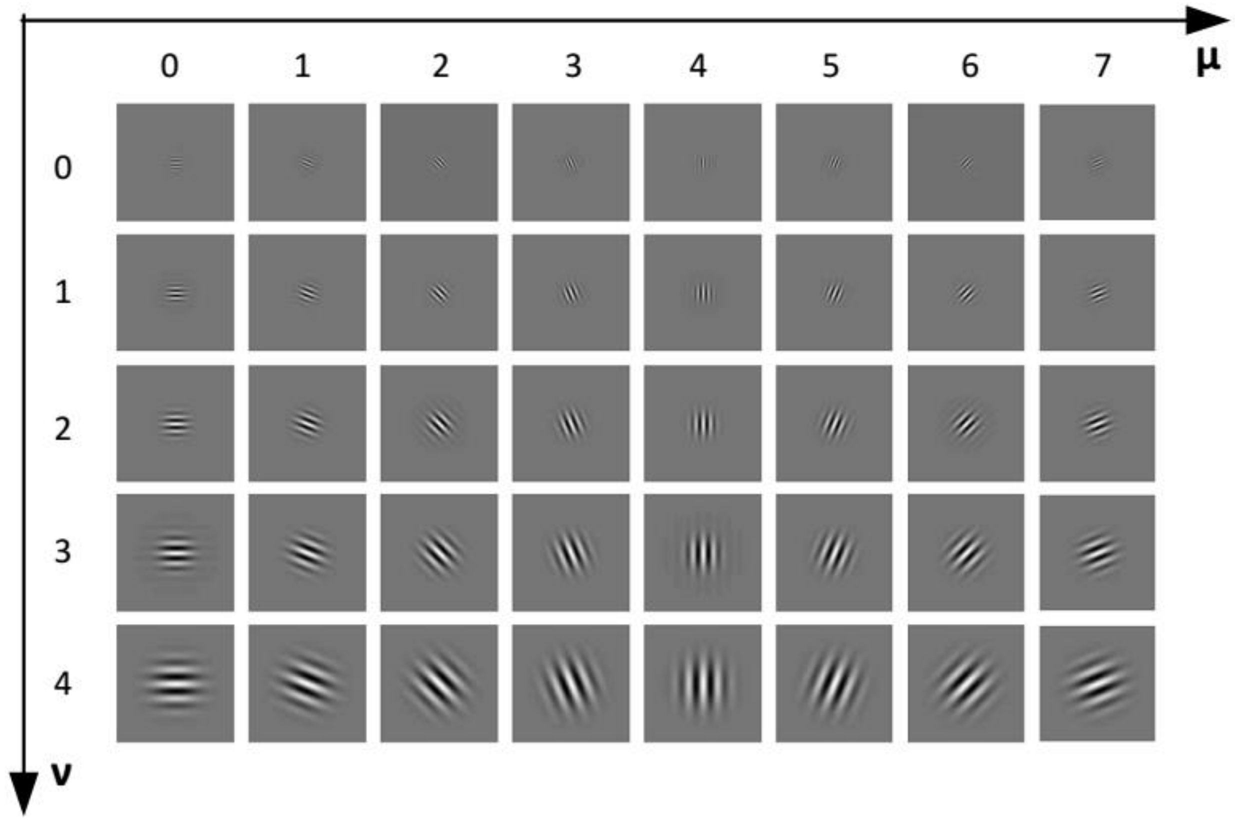


Figure 23: Forty Gabor filters applied

By convoluting the input image with Gabor filters, the Gabor-coded images are deduced for feature extraction. The equation represents this step is shown as following:

$$G_{\psi f}(\mathrm{x}, \mathrm{y}, \mu, \nu) = \mathrm{f}(\mathrm{x}, \mathrm{y}) * \psi_{\mu,\nu}(\mathrm{z}) \qquad (2)$$

Where:

- $G_{\psi f}(\mathrm{x}, \mathrm{y}, \mu, \nu)$: the Gabor-coded image
- f(x, y): the input image
- $\psi_{\mu,\nu}(\mathrm{z})$: the Gabor filter

Forty corresponding Gabor-coded images are shown in Figure 24 when applying equation (1) and (2) for the normalization image in Figure 21b.

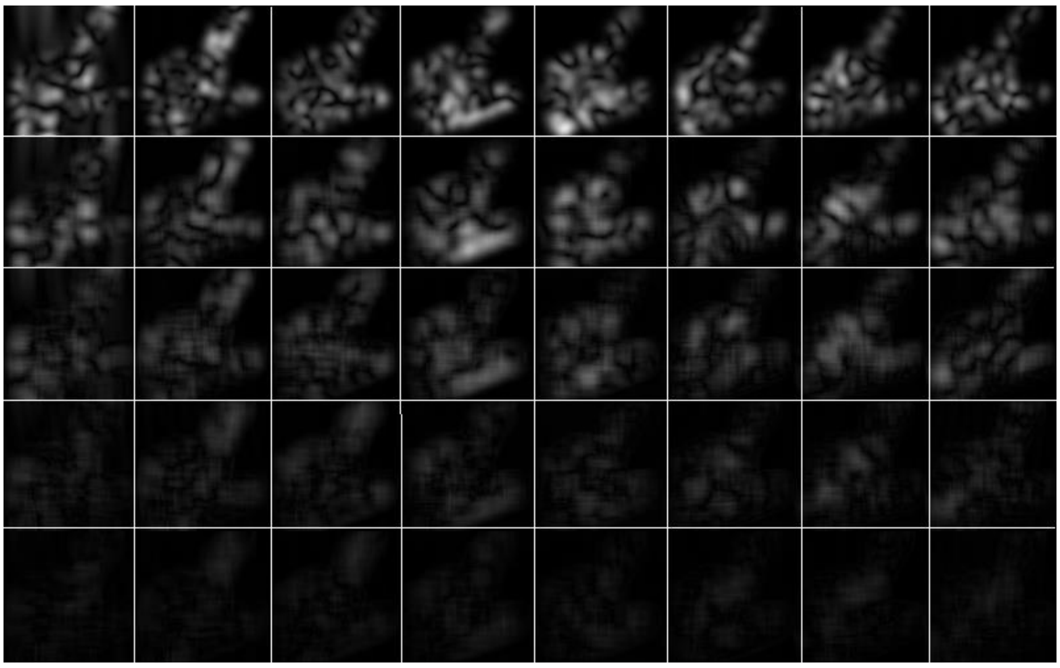


Figure 24: Gabor-coded images of image taken from Figure 21b

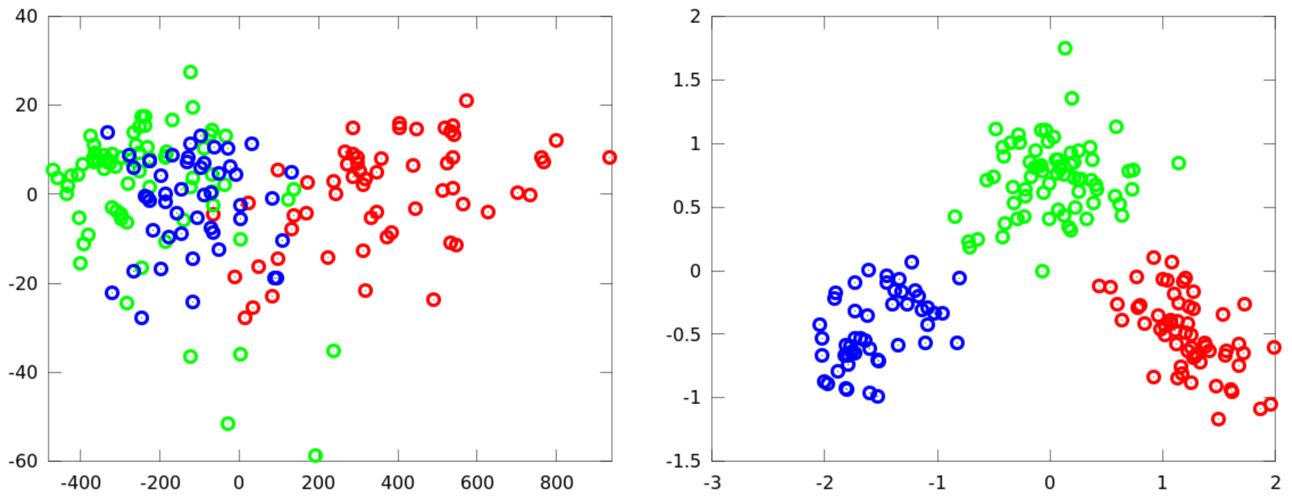
As mentioned in section 5.3.2, the chosen image size is 64x64, so a convolution with 40 Gabor filters will produce an array of 64x64x40 = 163840 elements, which is too big. Therefore, it requires to carry out the task of dimension reduction for faster classifying in classifier module.

5.4.2 Fisher’s Linear Discriminant for training purpose

The Fisher’s Linear Discriminant (FLD) algorithm keeps a same procedure in training or recognizing phase. The only difference is the number of input images. While it requires only one input image in recognizing phase, the training phase needs a set of images representing all the signs. This section will present the FLD algorithm in detail for training purpose.

In term of dimensionality reduction, the Principle Component Analysis (PCA) algorithm is an well-known technique by projecting an original dataset with high dimension into lower dimension space [38, 39]. Although PCA is optimal and useful for data compression, its performance in pattern recognition is not desirable [37]. The FLD which is PCA-based algorithm is proposed for overcoming this drawback [40].

The FLD enhances the classification capability of pattern recognition while still able to reduce dimension of the input features. Figure 25 shows an example that FLD performs better than PCA regarding discriminant ability with the same input data. The examined input data is taken from [41], algorithm applied and displayed by GNU Octave software [42].



a) PCA projection of 3 classes on two dimensions　　b) FLD projection of 3 classes on two dimensions

Figure 25: Discriminant capability of the two methods: PCA and FLD

The technique in the FLD algorithm is to maximize distance between the means (μ) of each class based on the calculation of the two importance information: the "within-class" (Sb) and "between-class" (Sw) matrix. Figure 26 visually describes these parameters for the three different classes as mentioned in Figure 25b.

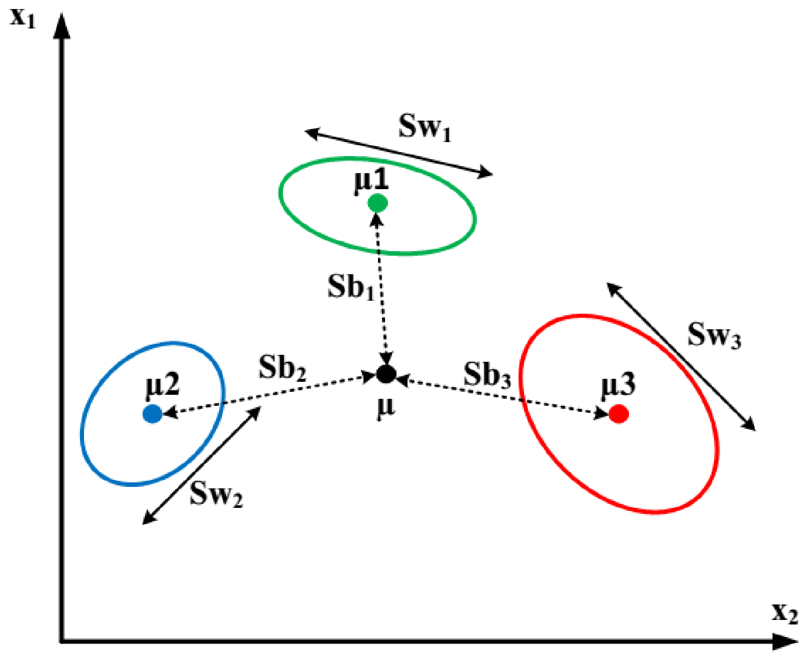


Figure 26: FLD projection of three classes onto two dimensions x_1, x_2

Assume that we have K classes (number of static hand gestures, K = 30) $C_1, C_2, \ldots, C_K$; in each class, we have $N_1, N_2, \ldots, N_K$ feature vectors respectively. In our project, each feature vector contains 163840 elements. The goal for applying FLD algorithm is to reduce number of these elements in each vector. Five steps required to be carried out are presented as follows:

<u>Step 1:</u> Each feature vector is presented as a matrix with the size of 1×N (N = 163840). These feature vectors are illustrated as below:

$$\begin{aligned}
&\text{Class } C_1: \ I_{11}, I_{12}, \ldots, I_{1N_1}\\
&\text{Class } C_2: \ I_{21}, I_{22}, \ldots, I_{2N_2}\\
&\quad\vdots\\
&\text{Class } C_K: \ I_{K1}, I_{K2}, \ldots, I_{KN_K}
\end{aligned}$$

<u>Step 2:</u> Calculate the mean for each class and the dataset

$$\begin{aligned}
&\text{Class } C_1: M_1 = \frac{1}{N_1}(I_{11} + I_{12} + \ldots + I_{1N_1})\\
&\text{Class } C_2: M_2 = \frac{1}{N_2}(I_{21} + I_{22} + \ldots + I_{2N_2})\\
&\quad\vdots\\
&\text{Class } C_K: M_K = \frac{1}{N_K}(I_{K1} + I_{K2} + \ldots + I_{KN_K})\\
&\text{DataSet}: M_{total} = \frac{N_1M_1 + N_2M_2 + \ldots + N_KM_K}{N_1 + N_2 + \ldots + N_K}
\end{aligned}$$

The size of M_i ($i = 1 \div K$) and M_{total} matrix is $1\times N$.

<u>Step 3:</u> Calculate the difference between the feature vectors and the mean in each class (Γ_i); between the mean in each class and the total mean (Φ_i).

$$\Gamma_i = \begin{pmatrix} I_{i1} - Mi \\ I_{i2} - Mi \\ \vdots \\ I_{iN_i} - Mi \end{pmatrix}, \quad \Phi_i = M_i - M_{total} \quad \text{with } i = \overline{0 \div K}$$

The size of Γ_i and Φ_i is $N_i\times N$ and $1\times N$ respectively.

<u>Step 4:</u> Calculate the within-class (Sw) and between-class (Sb) matrix:

$$Sw = \sum_{i=1}^{K} \Gamma_i^T \times \Gamma_i$$

$$Sb = \sum_{i=1}^{K} N_i \Phi_i^T \Phi_i$$

The size of Sw and Sb is $N\times N$. Eigenvectors and Eigenvalues of the matrix $S = Sw^{-1}\times Sb$ (size $N\times N$) which are used to present a new space called FLD space, will be computed by a in the next step.

<u>Step 5:</u> Calculate the Eigenvectors and Eigenvalues for training or recognizing purpose.

The Eigenvectors and Eigenvalues of the matrix S are calculated by using `GeneralEig()` function of the CLAPACK library[31]. The behind mathematical computation could be found at [43]. Each Eigenvector comes along with an Eigenvalue. In general, an Eigenvector presents a dimension in a new space of the input dataset while the Eigenvalue defines its Eigenvector's critical level to the dataset representation. It means that with higher Eigenvalue, the Eigenvector-pair contributes much more on reconstructing the dataset.

With the given matrix S, after processed by `GeneralEig()`, we have N Eigenvectors (E_i, i = 1÷N) with the size of N×1. We call a matrix of these Eigenvectors is $P_{full} = \{ E_1, E_2, \ldots, E_N \}$ (size N×N) where E_i is sorted in a descending order of corresponding Eigenvalue. P_{full} is also called a full projection matrix. A new space called FLD space formed by N Eigenvectors is not necessary due to its very big dimension (N = 163840). Hence, in this project, we empirically use the first K-1 Eigenvectors in P_{full}. K is always smaller than N because K is number of signs (K = 30) while N is number of feature's dimension (N = 163840). Consequently, the projection matrix used for this project is $P = \{ E_1, E_2, \ldots, E_{K-1} \}$.

At this stage, for training purpose, the total mean (M_{total}), projection (P, size N×(K-1)) and model matrix are stored for recognition. The model matrices (Υ_{ij}, size 1×(K-1)) are formed by projecting all input images into FLD space:

$$\Upsilon_{ij} = (I_{ij} - M_{total}) \times P \quad \text{with } i = \overline{1 \div K},\ j = \overline{1 \div N_i}$$

5.4.3 Input sign image projection for recognizing purpose

For recognizing purpose, we assume that the query sign image is Q_{raw}. After being processed by pre-process module and Gabor filter, we get the feature vector $Q_{feature}$ (size 1×N). This vector is then projected into the FLD space by the following equation:

$$\Upsilon_{query} = (Q_{feature} - M_{total}) \times P$$

Υ_{query} is called model vector of query sign which is used to compare with all model vectors in the stored database to find the best match. This procedure will be executed in the next module, classifier module.

5.5 Classifier module

This module is to compare the model vector of query sign with all model vectors stored in database to output the corresponding text. For carrying out this task, the cosine metric distance is used. The equation 3 defines how the cosine metric distance value is calculated:

$$d_{ij} = \frac{1 - \dfrac{\Upsilon_{ij} \bullet \Upsilon_{query}}{\|\Upsilon_{ij}\| \times \|\Upsilon_{query}\|}}{2} \qquad \text{with } i = \overline{1 \div K},\ j = \overline{1 \div N_i}$$

Where:

- ✓ d_{ij}: the cosine metric distance
- ✓ Υ_{ij}: the model vector taken from database
- ✓ Υ_{query} : the model vector of query sign
- ✓ $\bullet$: dot product
- ✓ $\|\square\|$: the norm operator

By finding the smallest d_{ij} when comparing Υ_{query} with all Υ_{ij} in sign database, the most similar sign will be presented with corresponding text. In this case, it will be a letter, tone or marker in alphabet of VSL system.

5.6 Project flowchart

The project flowchart shown in Figure 27 describes the overall implementation based on the proposed methodology. The project application is designed for operation modes: image collecting, training and recognizing mode. The image collecting operation generates a set of segmented hand images for training. In training operation mode, the FLD space is calculated and saved in sign database. By referencing this sign database, the text is displayed when its corresponding hand gesture appears in the recognizing operation mode.

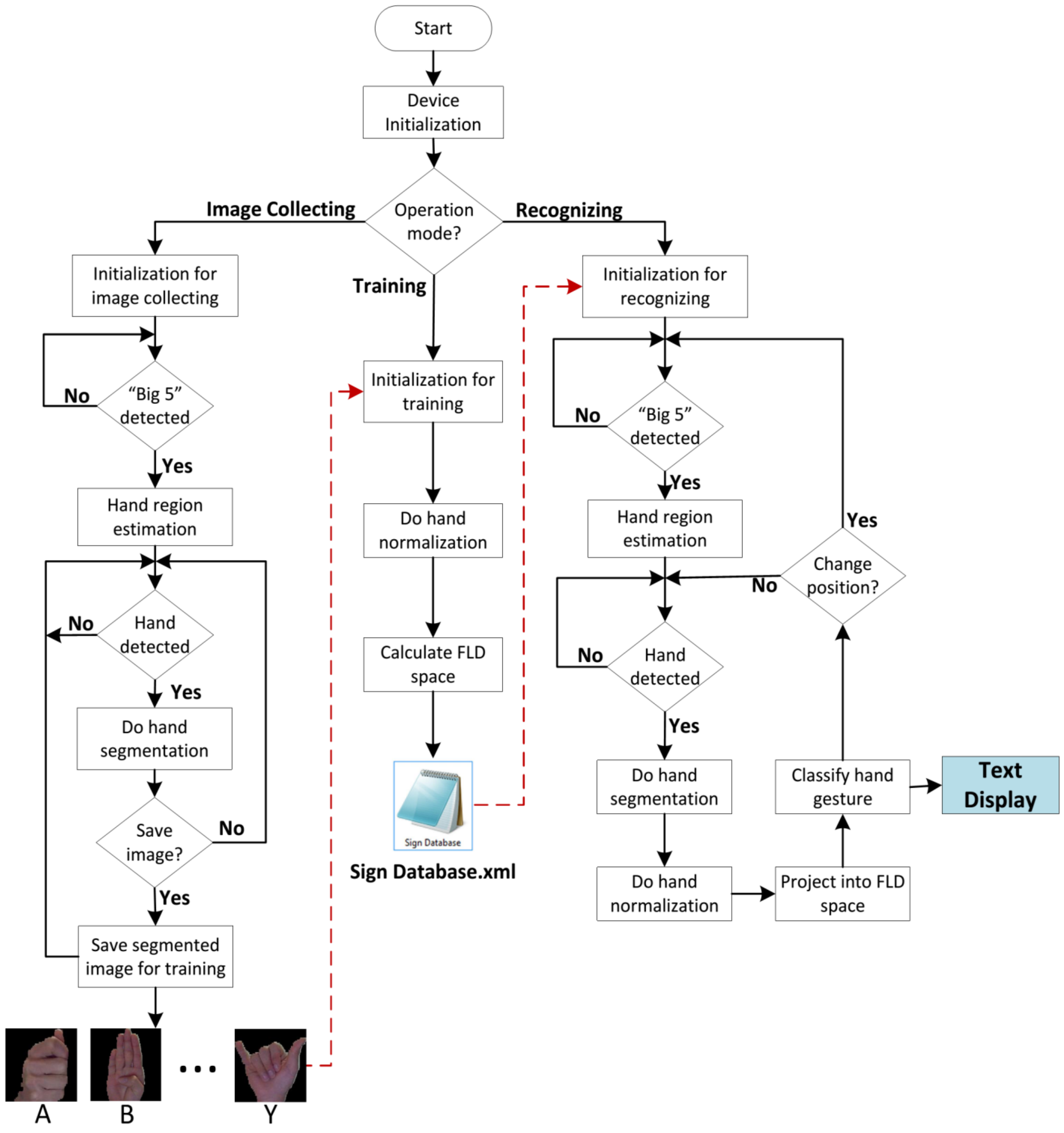


Figure 27: Project flow chart in detail

- **Image collecting mode:**
  This operation mode repairs input data for training mode by capturing images for each static hand-gesture. The working flow basically derives from section 5.3.1 which is hand segmentation. Number of images representing for each hand gesture will be save in a format as "*ID number_Hand gesture name_Ordinal number.bmp"*, for example: 01_A_1.bmp and 02_B_1.bmp. The chosen format for the image is Windows bitmaps due to its lossless characteristic [44].

  To make the program work in practice, the database of static signs should be built. The sign database includes 28 static signs with corresponding definitions (letter or marker or tone) as shown in Figure 28:

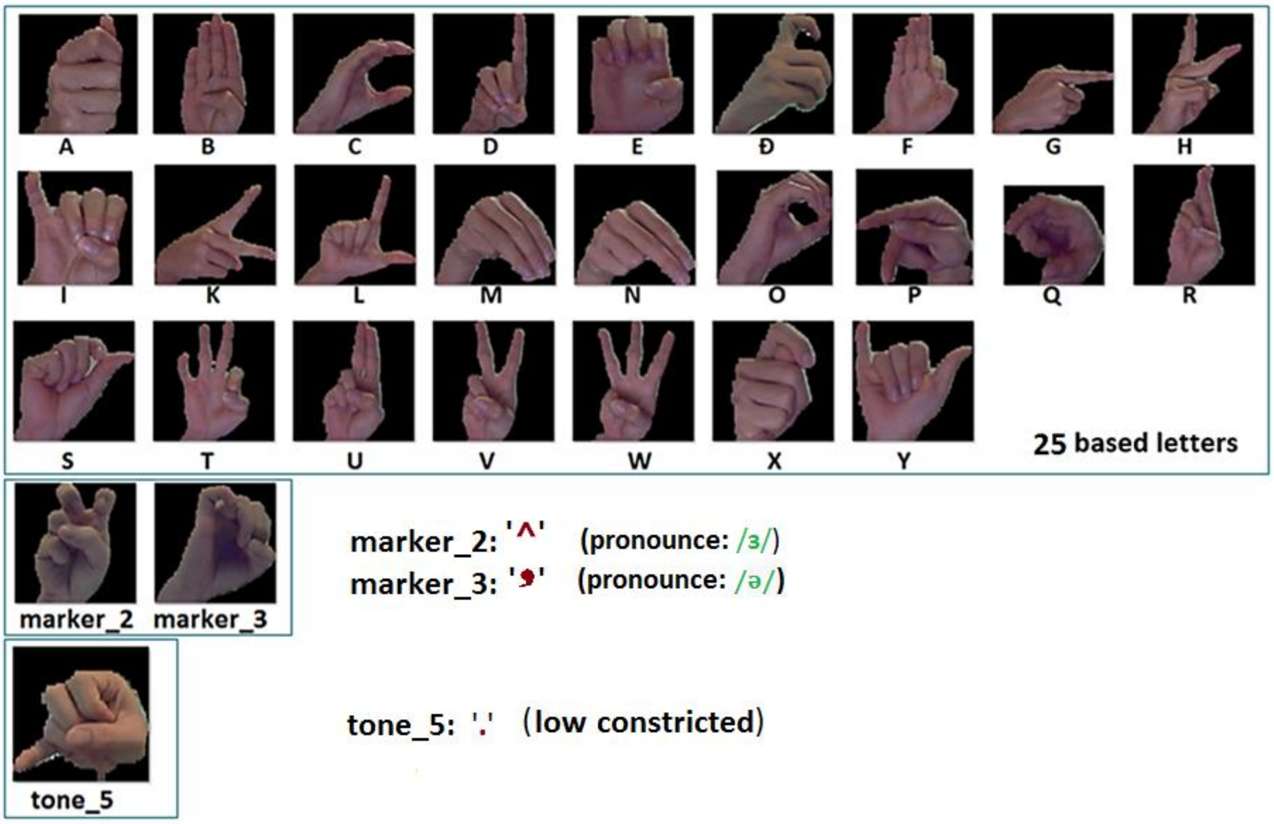

A
B
C
D
E
Đ
F
G
H
I
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
25 based letters
marker_2
marker_3
marker_2: '^' (pronounce: /ɜ/)
marker_3: '̉' (pronounce: /ə/)
tone_5
tone_5: '.' (low constricted)

Figure 28: Twenty-four static hand-gesture images

Each sign requires fifteen training images, hence there are 28*15 = 420 training images in total. To enhance the recognizing performance, training images of each sign are captured in different angles and distances to the Senz3D as illustrated in Table 3.

Table 3: Six different hand postures of letter ‘D’ are recorded for training

angle / distance	Normal posture	To left	To right
Further from camera			
Closer to camera			

- **Training mode:**
  With a set of images presenting for all signs, the training mode will establish a new space named FLD by applying hand normalization and FLD space calculation method as described in section 5.2.3 and 5.4. This new space information is stored in database of signs for later use in recognizing operation.

- **Recognizing mode:**
  At this mode, the database is first loaded for initialization. BIG5 sign is considered as a user’s position signal. Specifically, to use the program when it starts, the user must show BIG5 sign for hand region estimation. After that, whenever a hand-gesture is encountered, a series of processes including hand segmentation, hand normalization, FLD projecting and gesture classification is executed to find the most similar gesture from sign database for displaying. In the case that user wants to move to another position for interacting with the camera, BIG5 sign signal is required to recalculate the hand region. This results from the fact that at a particular distance of staring at the camera, when enabling position signal, the obtained hand region is different.

5.7 Test plan

In this section, the test plan will be taken place for the verification of static gesture recognition. The test is carried out on devices with respective specifications shown in Table 4:

Table 4: Devices' specifications used for system verification

Devices	Specifications	
Intel Creative Senz3D	Depth resolution	320x240
	RGB video resolution	1280x720
	Depth range	15-100cm
	Frame rate	30 fps
Laptop ASUS N56VZ	• Operating system: Windows 8.1 Pro 64-bit (Build 9600) • Processor: Intel Core i5-3210M CPU @2.5GHz (4 CPUs)	

	• Memory: 8192MB RAM • Graphic: NVidia GeForce GT 650M

The test is performed in the normal light condition and the camera is put on the laptop screen with adjustable angle. And the way of calculating the system accuracy is illustrated in Figure 29:

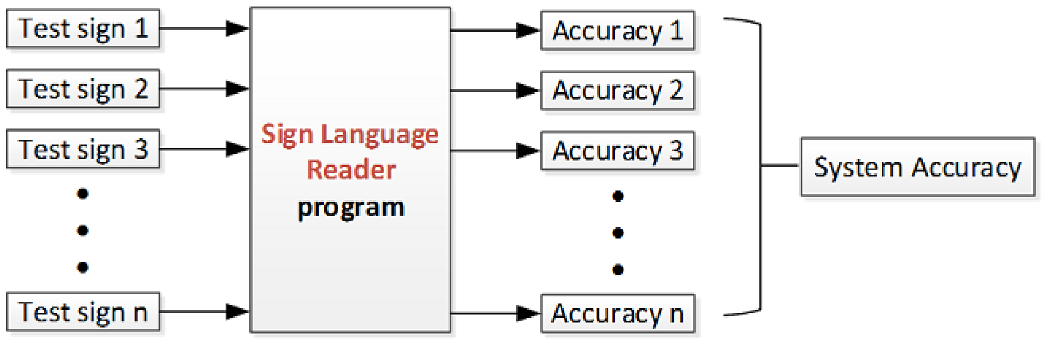


Figure 29: System accuracy measurement

The test signs will be transferred to the VSL Reader, and the corresponding output of this system will be observed and compared with the database to give out the most similar gesture. In the test plan, we apply m times of each input test sign. In total, we have n test signs as shown in Figure 29. The overall system accuracy is calculated as follows:

$$\text{Accuracy}_i = \frac{\text{Possitives}_i}{\text{Possitives}_i + \text{Negatives}_i} \times 100 = \frac{\text{Possitives}_i}{\text{m}} \times 100 \qquad \text{with i} = \overline{1 \div n}$$

$$\text{System accuracy} = \frac{\sum_{i=1}^{n} \text{Accuracy}_i}{\text{n}}$$

Where:

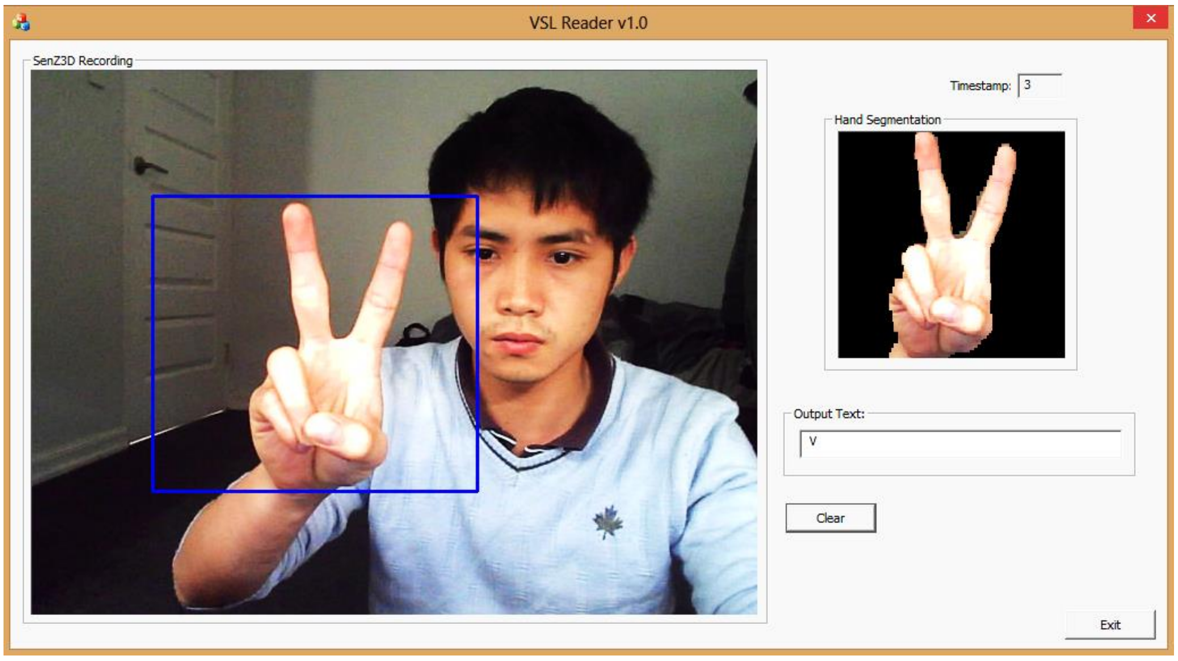
- Accuracy$_i$: is the accurate rate for recognizing "Sign i"
- Positives$_i$: is number of times when "Sign i" is recognized correctly
- Negatives$_i$: is number of times when "Sign i" is recognized incorrectly
- m: number of tests for each sign
- n: total signs

For the static gesture recognition, there are total 28 signs that will be tested on three different people. And each people perform their testing five times for each sign. Hence, we have:

- m = 5×3 = 15 testing times for each test sign
- n = 15 × 28 = 420 testing times for all test signs

5.8 Result and discussion

The program interface is designed friendly as shown in Figure 30. Whenever user's hand is presented, the segmented hand region and its text definition will be displayed at the Output Text.



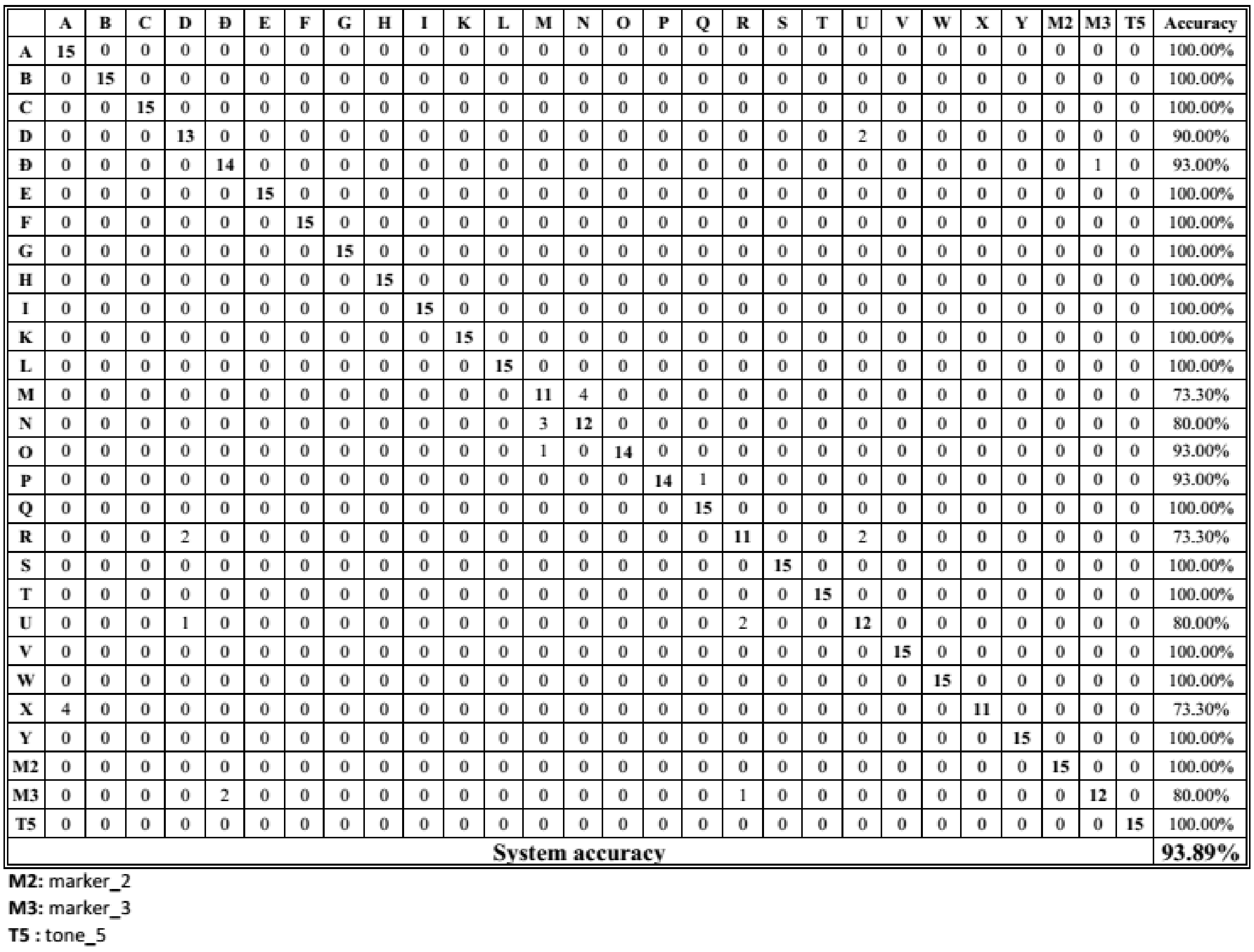
Timestamp: 3

Output Text: V

Figure 30: The Vietnamese Sign Language GUI in Static Gesture mode

For testing purpose, three people are invited to examine the system by showing their hand posture five times per one gesture with different distance and angle to the camera. Table 5 shows the result of static hand gesture recognition. The obtained accuracy rate is 93.89% which is quite good in a normal light condition. Although the accurate rate is not outstanding in comparison with some methods mentioned in literature review section, the improvement is potential because the background of hand region is mostly eliminated.

Table 5: System accuracy measurement for static hand gesture recognition



	A	B	C	D	Đ	E	F	G	H	I	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	M2	M3	T5	Accuracy
A	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100.00%
B	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100.00%
C	0	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100.00%
D	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	90.00%
Đ	0	0	0	0	14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	93.00%
E	0	0	0	0	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100.00%
F	0	0	0	0	0	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100.00%
G	0	0	0	0	0	0	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100.00%
H	0	0	0	0	0	0	0	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100.00%
I	0	0	0	0	0	0	0	0	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100.00%
K	0	0	0	0	0	0	0	0	0	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100.00%
L	0	0	0	0	0	0	0	0	0	0	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100.00%
M	0	0	0	0	0	0	0	0	0	0	0	0	11	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	73.30%
N	0	0	0	0	0	0	0	0	0	0	0	0	3	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	80.00%
O	0	0	0	0	0	0	0	0	0	0	0	0	1	0	14	0	0	0	0	0	0	0	0	0	0	0	0	0	93.00%
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14	1	0	0	0	0	0	0	0	0	0	0	0	93.00%
Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	0	0	0	0	0	0	0	0	0	0	0	100.00%
R	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0	2	0	0	0	0	0	0	0	73.30%
S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	0	0	0	0	0	0	0	0	0	100.00%
T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	0	0	0	0	0	0	0	0	100.00%
U	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	12	0	0	0	0	0	0	0	80.00%
V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	0	0	0	0	0	0	100.00%
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	0	0	0	0	0	100.00%
X	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0	0	73.30%
Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	0	0	0	100.00%
M2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	0	0	100.00%
M3	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	12	0	80.00%
T5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	100.00%
System accuracy																													**93.89%**

M2: marker_2
M3: marker_3
T5 : tone_5

As shown in Table 3, there are three letters got lowest accuracy of 73.3%: M, R and X. Where M gets confused with letter N; R is recognized as D and U sometimes; X gets confused with A. The reason results from the fact that these gestures have similar posture (see Figure 28).

Regarding the running speed, the applied technique for a database of 28 gestures performs in real-time with 14fps.

In summary, the implementation of static gesture recognition has achieved a good results that satisfies the project scope in term of accuracy, operational speed and interface design.

CHAPTER 6
DYNAMIC GESTURE RECOGNITION

In the previous section, we have discussed how the system recognizes static hand-gesture and translates it into text. However, the dynamic gestures are used more often in reality hence the task of recognizing this gesture type critically contributes to the entire gesture recognition system.

6.1 Methodology

In constructing a dynamic gesture recognition, there are vast of methods suggested including Hidden Markov Models (HMM) [45], dynamic programing [46] and neural networks [47]. But these methods requires expensive computation which results in non-real-time application. In order to facilitate the task of building a gesture recognition with less processing effort but still keeping good accuracy, Jacob et al. have proposed a method, called *$1 recognizer* algorithm that basically involving trigonometry and geometric computation [48]. The goal of this algorithm is for touch screen devices like smart phone or tablet. In this project, a modification of *$1 recognizer* is applied for dynamic gesture recognition on 3D camera device.

Figure 31 shows the suggested modules in implementing dynamic gesture recognition based on [48]. It basically includes three main modules: input data, normalization and classifier. The given diagram targets for two working phases: recognizing and training phase. In the training phase, the input data will be processed and saved into database. Referring to this database, the dynamic gesture recognition core will select the best matching sign as well as display corresponding text at output.

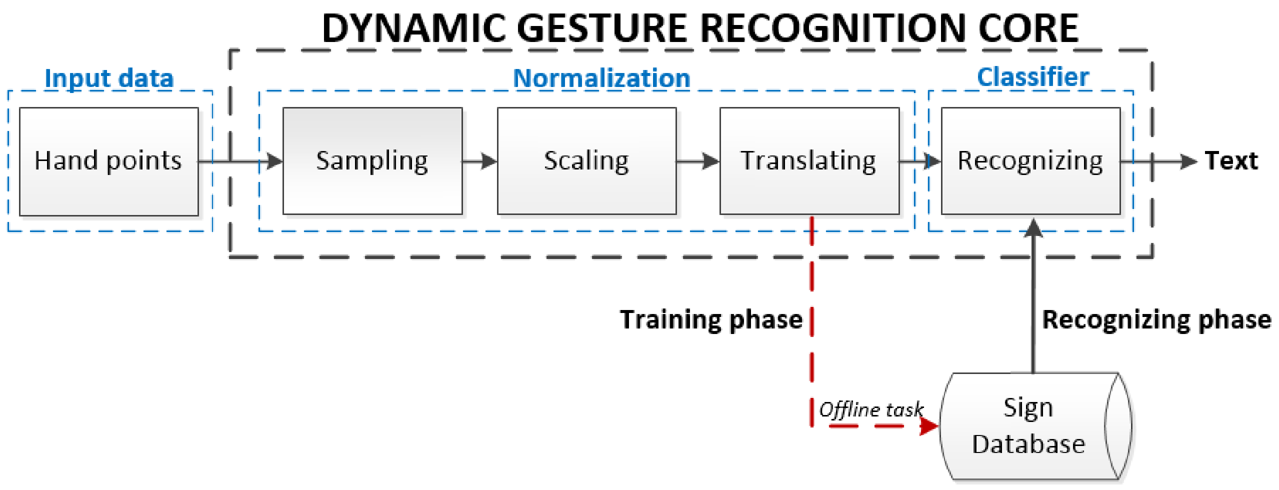


Figure 31: Dynamic gesture recognition flow

The function of each module is briefly described as follows:

- *Input data module*: tracks the finger's tip movement which creates a series of points in two dimensional format. This data will be fed into the dynamic gesture recognition core to continue processing.
- *Normalization module*: converts input data into a standard form such that it could be used to compare with each other. It includes three sub-modules: sampling, scaling and translating.

- *Classifier*: is responsible for comparing the input data with gestures stored in database to find the most similar one.

6.2 Input data module

With the support of Intel Perceptual Computing SDK, Senz3D could detect the appearance of hand as well as track hand movement. Furthermore, Senz3D is able to locate hand geometric nodes including position of fingers, palm center and elbow as shown in Figure 32. In addition to this, there are three relative positions: high, medium and low of the hand finger. The hand geometric nodes detection algorithm is embedded in the Senz3D's chip hence the data stream speed still keeps at maximum value, 30 frame per second (fps).

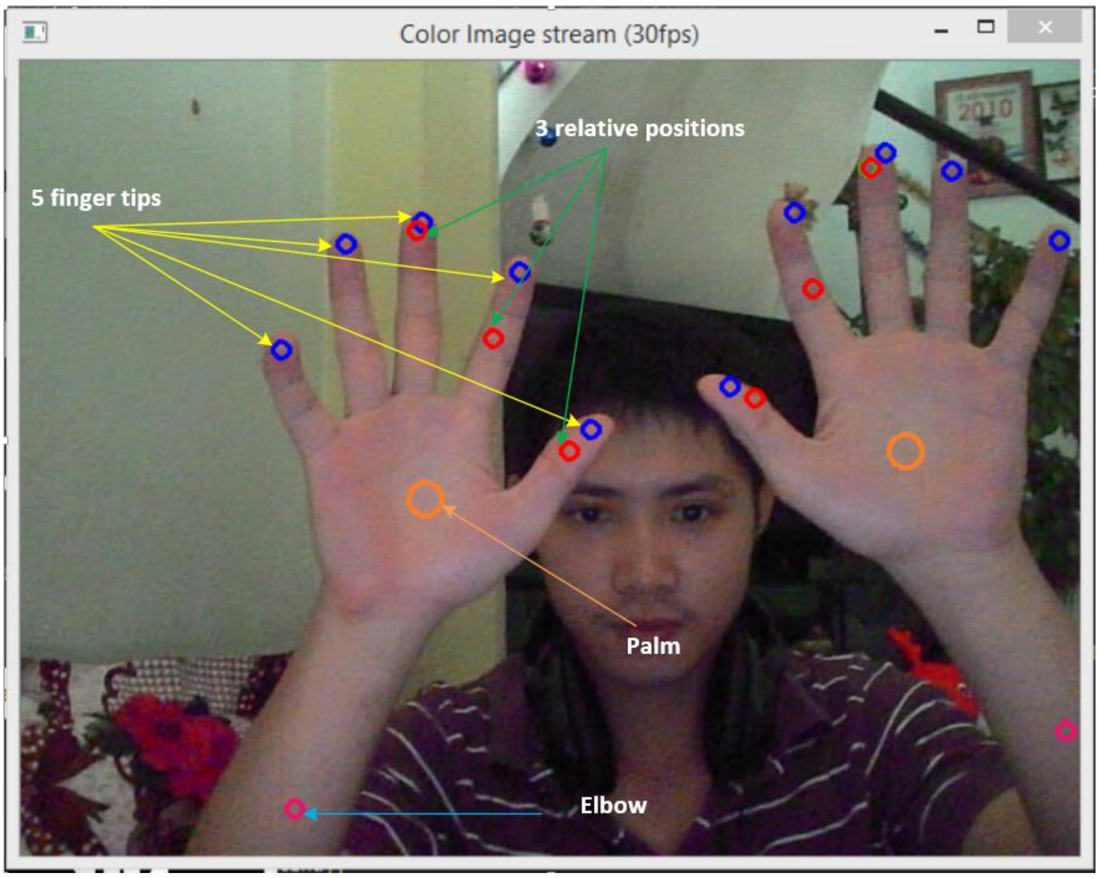


Figure 32: Hand geometric nodes detected by Senz3D

In this project, the index finger tip position is preferred due to its easy-to-track and popular-in-use in sign language. When the hand moves, a sequence of finger tip's points is tracked and saved simultaneously into an array for later process. But the problem is that how to detect the start and end signal for the camera to understand when the gesture starts as well as when it finishes. In dealing with this, these two signals are defined as below:

- **Start signal**: is detected when the hand in the posture that its index finger tip points upward as illustrated in Figure 33.

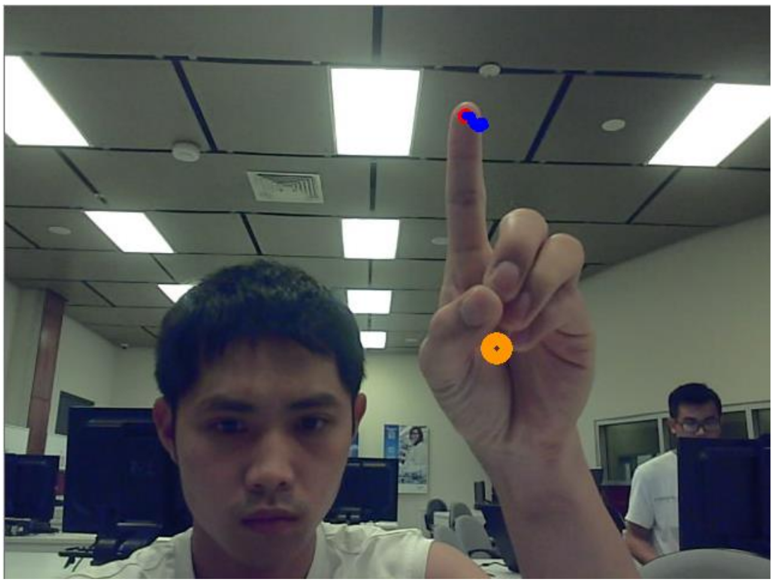


Figure 33: Start signal: asserted only when index finger points upward

- **End signal**: is asserted when both hands (left and right) appear concurrently in front of the camera (see Figure 34).

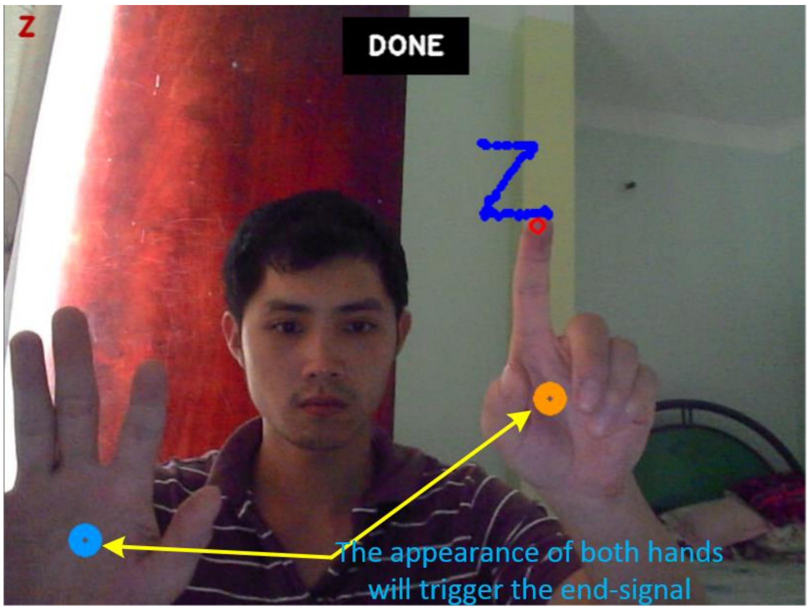


Figure 34: End signal: asserted when both hands show up the Senz3D

6.3 Normalization module

6.3.1 Sampling hand-point data

Data recorded from previous module contains a big amount of points representing the fingertip position in 2D space which create gesture path. To reduce the computation cost of the system core, it requires to sample the input data.

Because number of points of each gesture depends on the user's movement speed as described in Figure 35 for gesture 'J', sequence of input points will be sampled in pixel length domain to make gesture paths comparable at different speeds.

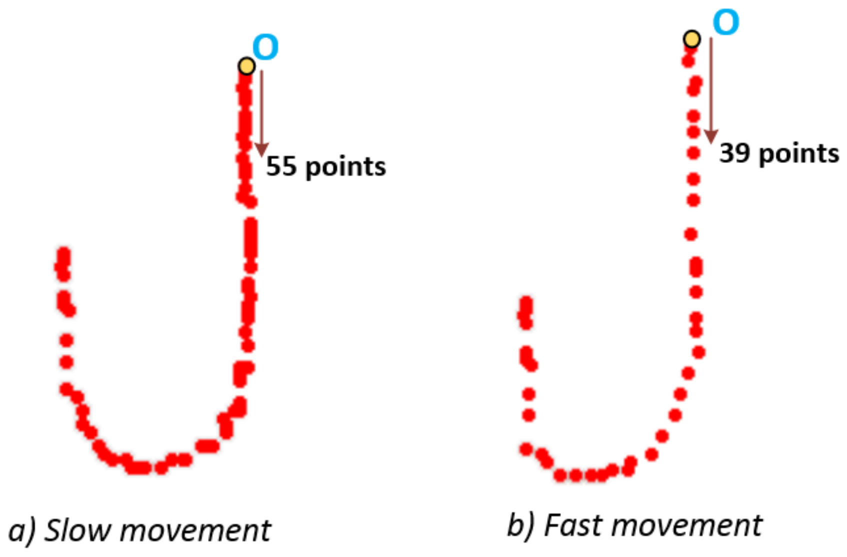


Figure 35: The number of input points varies depending on movement speed

Each gesture path starts at origin O and it is sampled such that we get L points with equal spaced pixels. Choosing L will make a tradeoff between the accuracy and operational speed. The precision of gesture path recognizing is deducted with the too small L, while it costs the processing time when L is too big. The L = 40 (pixels) is found empirically which compensates for the speed and accuracy.

To sample, based on the input points, L new points will be generated and tracked closely the original path. It first calculates the length of a gesture path beginning at O by accumulating pixel deviation between two consecutive points. Then the path is divided into (L-1) equal parts between L new points. We assume that the total length is N, therefore the each equal part will be N/(L-1). In the case that number of input points is less than 40, the sampling is still working fine because each new generated point is computed through linear interpolation. At the end of this module, the sampled gesture is obtained with the fix number of points, L.

For illustration, Figure 36 shows an example of sampling 'J' gesture with L = 7. The *length* is a sum of all pixel deviation between two consecutive original points from *d1÷d10*. This *length* is then divided into six equidistantly spaced points, *k*.

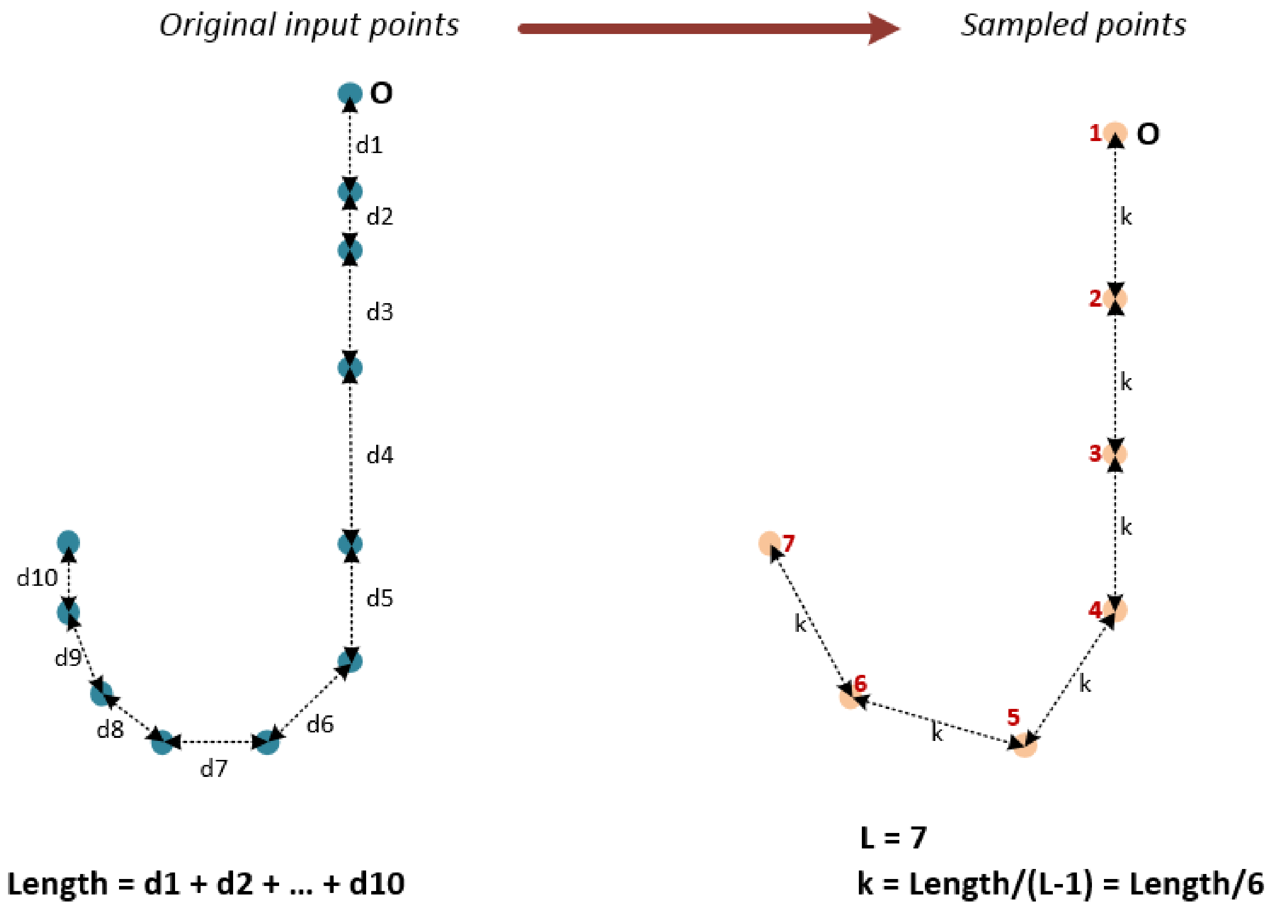


Figure 36: Sampling process example with L = 7

6.3.2 Scaling region of hand-point data

Before scaling, it first requires to locate the sampled gesture area. As shown in Figure 37, the bounding area is determined by a top left point and a minimum square size R that covers all the gesture path. Next, the located area is scaled to a fixed square with the size of 80 in order to make all gestures comparable. The square size of 80x80 pixel is found through practical experiment.

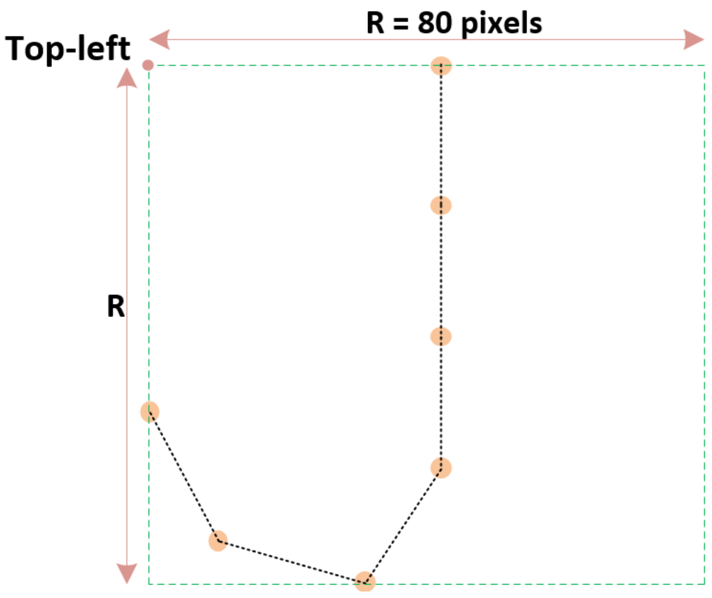


Figure 37: Gesture scaled a fix square size of 80 pixels

6.3.3 Translating hand-point data to the local-centroid coordinate

All the gesture points' value are defined as 2-dimensonal format and presented on the image coordinate which the origin is at the top left as shown in Figure 38. In reality, users could perform their gestures on any location of the image coordinate. And this results in the need of coordinate transformation such that all gestures could be compared with each other without depending on different gesture locations.

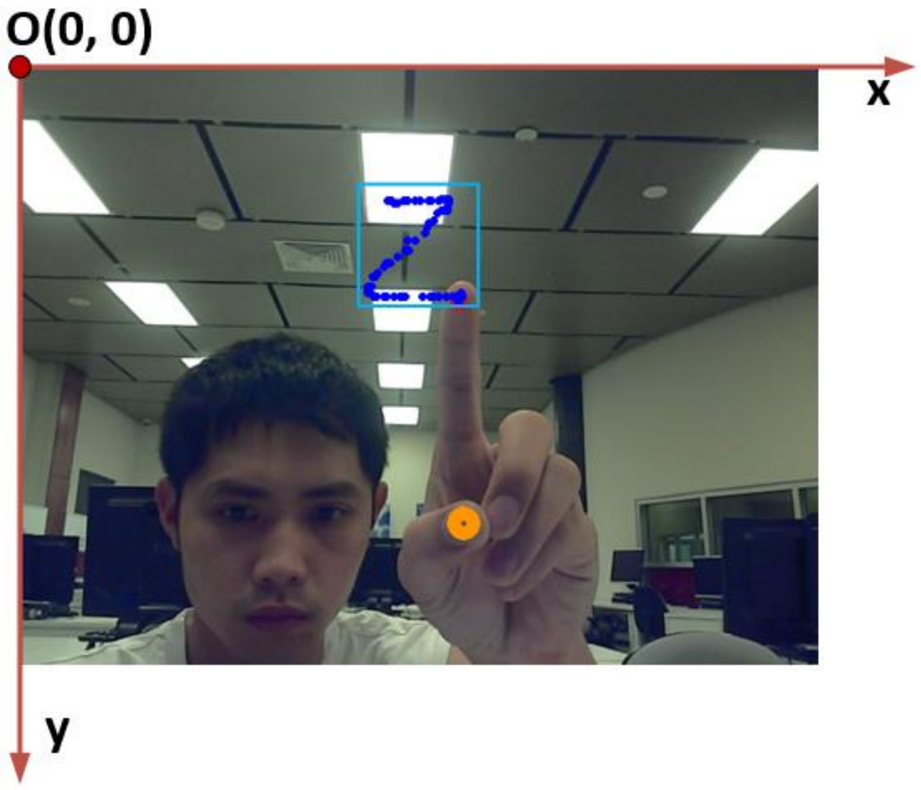


Figure 38: Image coordinate

The new origin is set at the center of the bounding box constructing in scaling sub-module. By configuring like this, after processed by the normalization module, the input gestures are now presented in uniform way which make the task of classifier easier.

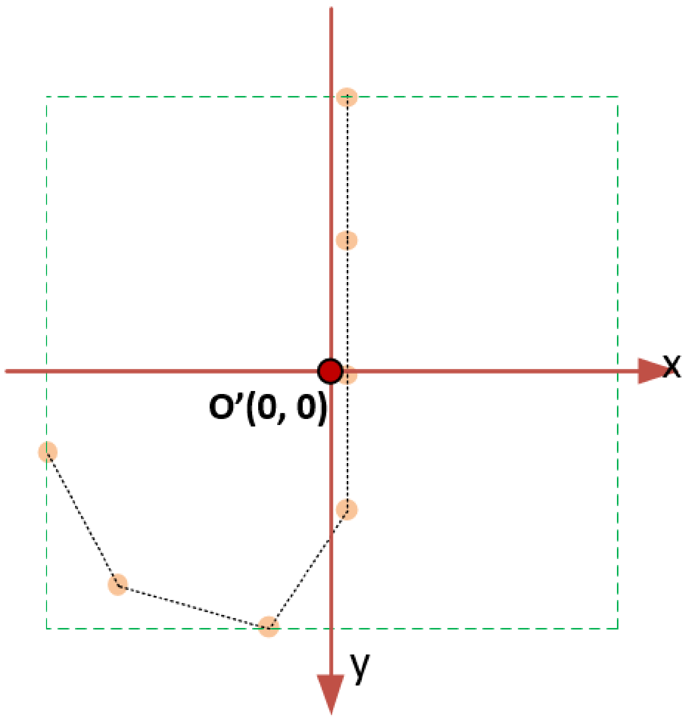


Figure 39: Origin O(0, 0) is translated to the center of the bounding square

At this point, depending on working mode, the normalization module perform different tasks:

- *Training mode*: gesture is stored into sign database with it corresponding text definition.
- *Recognizing mode*: it pass the normalized gesture to the classifier for recognizing.

6.4 Classifier module

In this module, the Euclidean distance is applied to find the difference between the input gestures and database. The formula is described as below:

$$d_i = \frac{\sum_{k=1}^{N}\sqrt{(C[k]_x - T_i[k]_x)^2 + (C[k]_y - T_i[k]_y)^2}}{N}$$

Where:

- C is a set of candidate points
- T_i is a set of template points of gesture i in the database
- N is number of points of one gesture
- d_i is the Euclidean distance between candidate C with template points T_i

The minimum d_i gives the most similar gesture from database.

6.5 Database building

Similar to the static database building, the dynamic ones are built based on the VSL system. We note that the dynamic database only includes seven different signs and these signs are so significant different. Additionally, the original $1 recognizer algorithm still performs well with the database of one sample for each sign [48]. Therefore, the dynamic database only contains seven samples for seven dynamic gestures as illustrated in Figure 40. These seven dynamic gestures includes two letters (J, Z), one marker (marker_1) and four tones (tone_1, tone_2, tone_3 and tone_4).

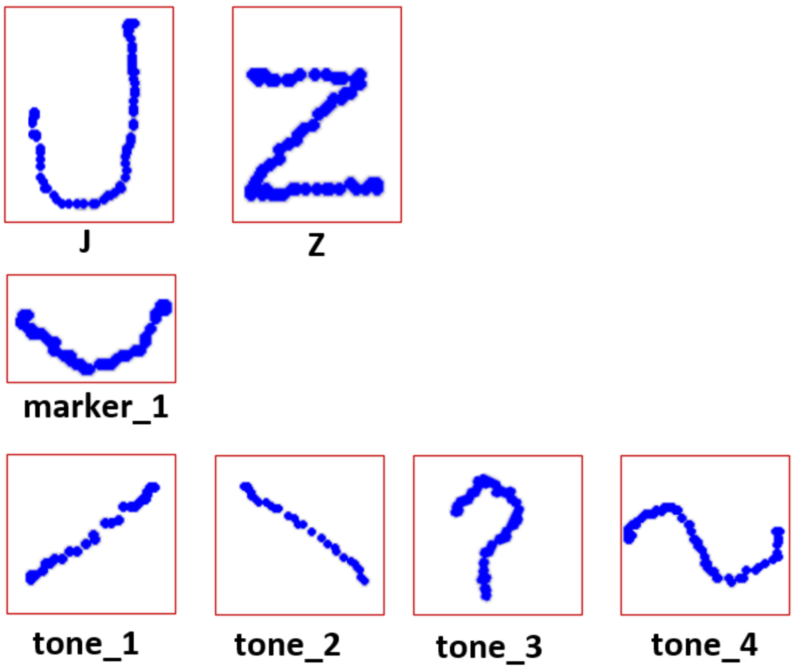


Figure 40: Seven dynamic signs definition

6.6 Result and discussion

Figure 41 shows the program GUI designed for dynamic recognition mode. Users simply present their gestures in front of camera and the corresponding text will be displayed on screen whenever end signal is detected. In contrast with static one, dynamic gesture recognition still performs well in low light condition. It results from the fact that the dynamic one is working on geometric points, while the static one requires to extract local features of the hand.

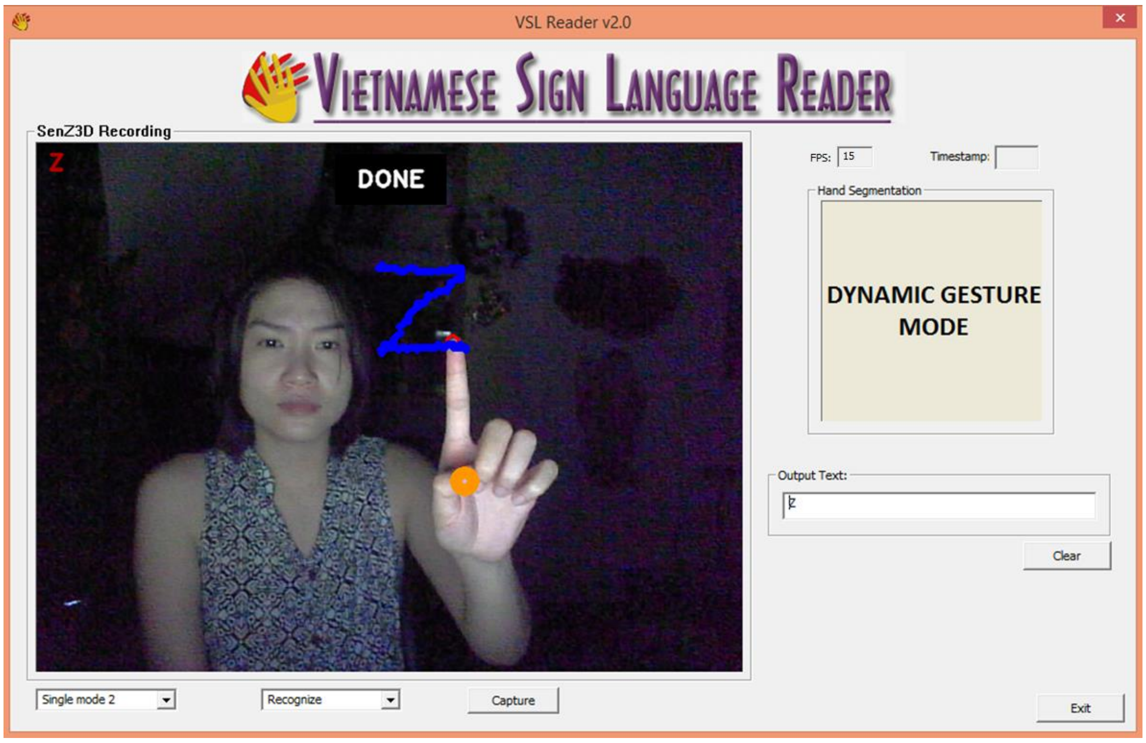


Figure 41: The Vietnamese Sign Language GUI in Dynamic Gesture mode

In order to evaluate the performance of the proposed method, each dynamic gesture is examined fifteen times on three different people. System working in Dynamic Gesture Mode achieves a very good accuracy rate with 97.14% as shown in Table 6. Most gestures get 100% accurate. Only the marker_1 sometimes gets confused with tone_4 because they are presented in a similar shape.

Table 6: System accuracy measurement for dynamic hand gesture recognition

	J	Z	M1	T1	T2	T3	T4	Accuracy
J	**15**	0	0	0	0	0	0	100.00%
Z	0	**15**	0	0	0	0	0	100.00%
M1	0	0	**12**	0	0	0	3	80.00%
T1	0	0	0	**15**	0	0	0	100.00%
T2	0	0	0	0	**15**	0	0	100.00%
T3	0	0	0	0	0	**15**	0	100.00%
T4	0	0	0	0	0	0	**15**	100.00%
System accuracy								**97.14%**

M1: marker_1 **T1**: tone_1 **T2**: tone_2 **T3**: tone_3 **T4**: tone_4

The obtained accuracy (97.14) is approximate with the original $1 recognizer algorithm (97%) which is optimized for touch screen devices. In other words, we have successfully transferred technique for working with 3D camera device. The accuracy could be improved with bigger database when we increase number of templates for each gesture, but it will cause system's computational burden.

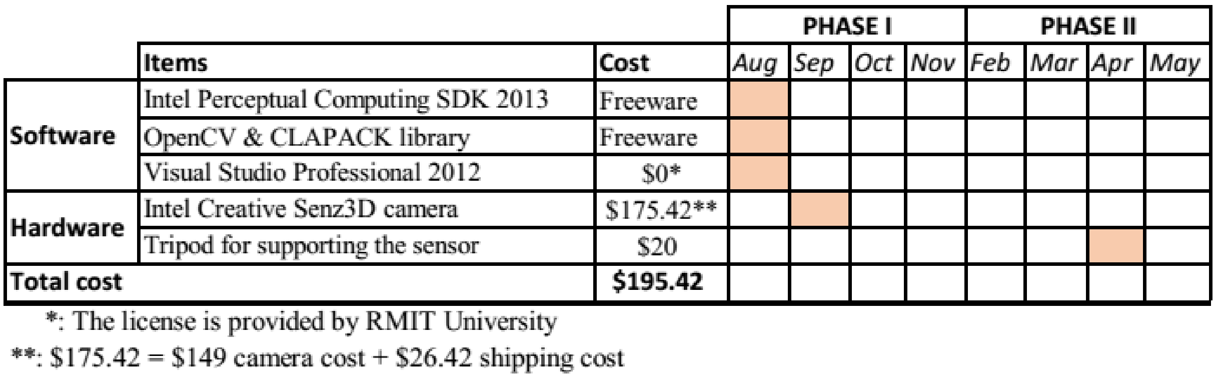
In term of operational speed, after optimizing, the program operates in real time with 15 fps. This speed is advanced over the real time system for dynamic hand gesture recognition with Kinect in [49], which runs at 10fps.

CHAPTER 7
PROJECT MANAGEMENT

7.1 Budget plan

Table 7 presents the items needed for this project as well as the time it should be available for a six-month-project. For the software items, Intel Perceptual Computing SDK, OpenCV and CLAPACK library are freeware. Additionally, Visual Studio Professional 2012 license is supported by RMIT hence it costs nothing for software items. The main budget is spent for the 3D sensor and the tripod with $195.42 in total.

Table 7: Budget plan



			PHASE I				PHASE II			
	Items	Cost	Aug	Sep	Oct	Nov	Feb	Mar	Apr	May
Software	Intel Perceptual Computing SDK 2013	Freeware	🟧							
	OpenCV & CLAPACK library	Freeware	🟧							
	Visual Studio Professional 2012	$0*	🟧							
Hardware	Intel Creative Senz3D camera	$175.42**		🟧						
	Tripod for supporting the sensor	$20							🟧	
Total cost		$195.42								

*: The license is provided by RMIT University

**: $175.42 = $149 camera cost + $26.42 shipping cost

7.2 Timeline

The project will be carried out for two semesters from August 2013 to May 2014. Five milestones should be delivered during this duration as shown in Figure 42 and Figure 43.

- *Planning*: sketch the plan for carrying out the project.
- *Research*: mainly focus on evaluating available methods and choosing the appropriate one for the project.
- *System design*: static and dynamic hand-gesture recognition will be implemented. This phase is the most important part of the whole project implementation and it also takes the longest time of the project timeline.
- *Integration & System testing level*: the two main parts: static and dynamic gesture recognition will be integrated to perform on one program. All possible test cases should be taken place to ensure the system could be applied in practice.
- *Project evaluation*: the overall system performance should be evaluated. Additionally, the report, presentation should be repaired for the final deliverable.

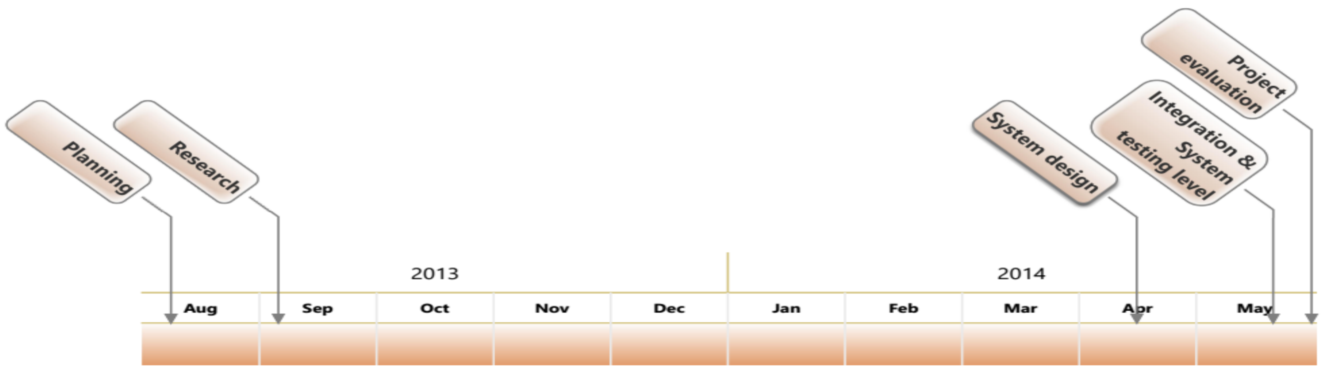
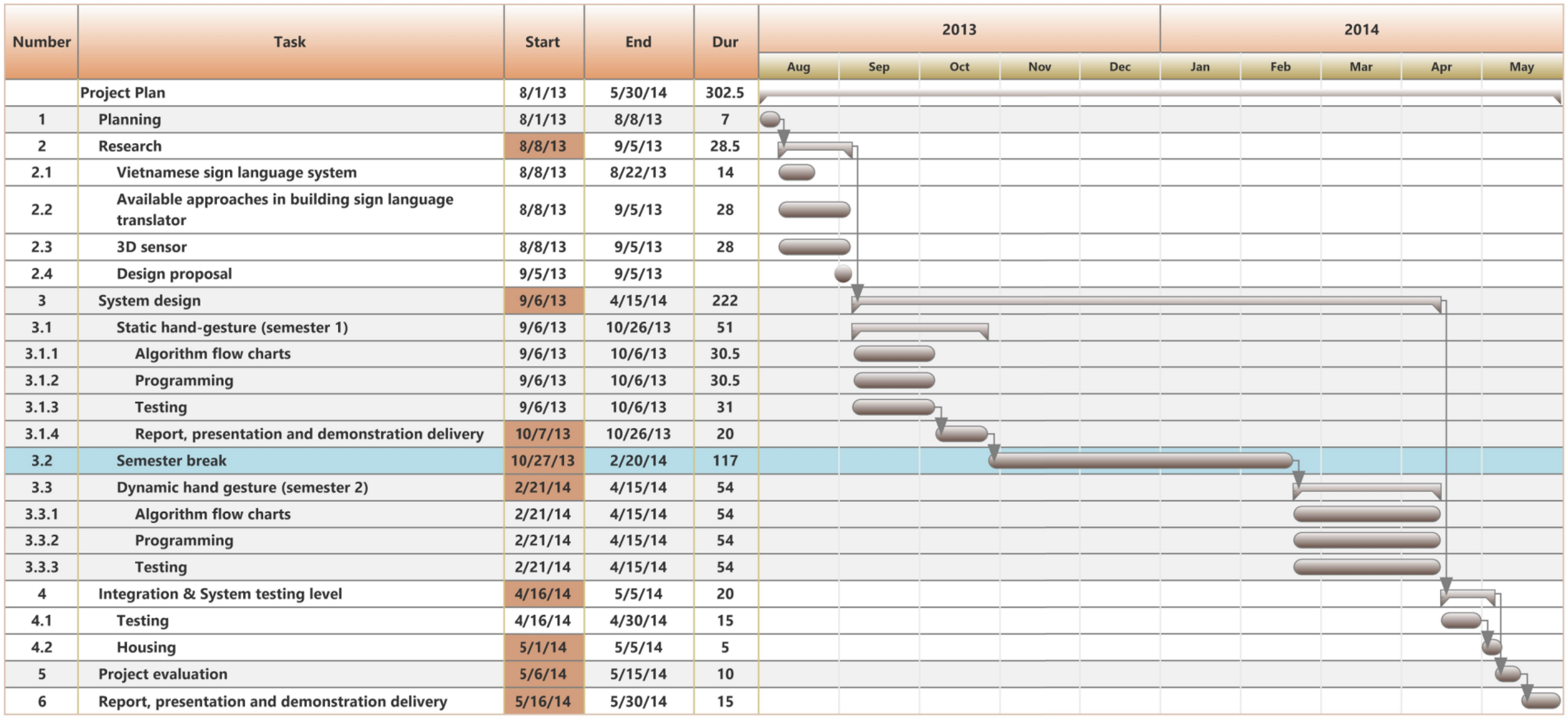


Figure 42: Project timeline



Number	Task	Start	End	Dur
	Project Plan	8/1/13	5/30/14	302.5
1	Planning	8/1/13	8/8/13	7
2	Research	8/8/13	9/5/13	28.5
2.1	Vietnamese sign language system	8/8/13	8/22/13	14
2.2	Available approaches in building sign language translator	8/8/13	9/5/13	28
2.3	3D sensor	8/8/13	9/5/13	28
2.4	Design proposal	9/5/13	9/5/13	
3	System design	9/6/13	4/15/14	222
3.1	Static hand-gesture (semester 1)	9/6/13	10/26/13	51
3.1.1	Algorithm flow charts	9/6/13	10/6/13	30.5
3.1.2	Programming	9/6/13	10/6/13	30.5
3.1.3	Testing	9/6/13	10/6/13	31
3.1.4	Report, presentation and demonstration delivery	10/7/13	10/26/13	20
3.2	Semester break	10/27/13	2/20/14	117
3.3	Dynamic hand gesture (semester 2)	2/21/14	4/15/14	54
3.3.1	Algorithm flow charts	2/21/14	4/15/14	54
3.3.2	Programming	2/21/14	4/15/14	54
3.3.3	Testing	2/21/14	4/15/14	54
4	Integration & System testing level	4/16/14	5/5/14	20
4.1	Testing	4/16/14	4/30/14	15
4.2	Housing	5/1/14	5/5/14	5
5	Project evaluation	5/6/14	5/15/14	10
6	Report, presentation and demonstration delivery	5/16/14	5/30/14	15

Figure 43: Project plan development

CHAPTER 8
CONCLUSION AND FUTURE WORKS

8.1 Conclusion

In conclusion, number of people experiencing difficulties in speaking and hearing occupies a large part of Vietnamese handicapped. Sign language is considered as a primary way in communication for the deaf/mute. With respect to Vietnamese sign language, it is comprised of static and dynamic hand gesture. In order to bridge the communication gap, this project aims to construct a system that can recognize both types of gesture Vietnamese sign language. There are two current approaches in building a sign language recognition system: gloved-based and vision-based. After a carefully research, the vision-based method implementing on 3D sensor is selected due to its lower cost and capability in tracking fingertip.

The implementation is mainly divided into two parts: one for static and one for dynamic gesture recognition. Both implementation flows have successfully recognized 27 based letters, 3 markers and 5 tones in VSL system. In term of performance, the system achieves a good result with more than 90% of accuracy and real-time running at the speed exceeding 10 frames per second.

8.2 Future works

The accuracy obtained in static gesture recognition is 93.14% that could be improved by increasing number of features extracting from each gesture. Furthermore, adding more training images could also contribute to the overall system accuracy. In term of operational speed, it could be improved by optimizing the implemented code.

The current system working on two separately modes: static and dynamic mode. Because most sign languages in practice are presented as a combination of these two gesture types, we aim to construct the combination working mode in the future works. Additionally, the facial emotional that is also considered sign language could be added to the program as a new feature.

In order to popularize the application, the program intend to extend dictionary not only for Vietnamese but also for other languages.

The Senz3D technology might be integrated on mobile devices such as laptop, tablet or even smartphone in the short future. Therefore, we also aim to develop the program for working on other platforms including Windows RT, iOS and Android.

REFERENCES

[1] C. J. Neidle, *The syntax of American Sign Language: Functional categories and hierarchical structure*: The MIT Press, 2000.

[2] S. R. o. V. N. G. S. Office, "Comprehensive Results on the Household Living Standards Survey 2006," H. S. P. House, Ed., ed, 2006.

[3] P. B. Braem, *Einführung in die Gebärdensprache und ihre Erforschung*: Signum-Verlag, 1995.

[4] H. C. University of Education, "Vietnamese Sign Language Dictionary For People With Hearing Impairment," ed, 2004.

[5] (12 August). *Phase I: Practical Dictionaries of Asia-Pacific Sign Languages (2003-2007)*. Available: http://www.cslds.org/apsl/research.php?id=1#vietnam

[6] T. N. T. Nguyen, "A 3D conversational agent for presenting digital information for deaf people," in *Agent Computing and Multi-Agent Systems*, ed: Springer, 2009, pp. 319-328.

[7] D. J. Sturman and D. Zeltzer, "A survey of glove-based input," *Computer Graphics and Applications, IEEE*, vol. 14, pp. 30-39, 1994.

[8] R. Watson, "A survey of gesture recognition techniques," 1993.

[9] M. W. Kadous, "Auslan sign recognition using computers and gloves," in *Deaf Studies Research Symposium*, 1998.

[10] *5DT Data Glove 14 Ultra*. Available: http://www.5dt.com/products/pdataglove14.html

[11] (22 August). *CyberGlove III*. Available: http://cyberglovesystems.com/products/cyberglove-III/overview

[12] S. Saengsri, V. Niennattrakul, and C. A. Ratanamahatana, "TFRS: Thai finger-spelling sign language recognition system," in *Digital Information and Communication Technology and it's Applications (DICTAP), 2012 Second International Conference on*, 2012, pp. 457-462.

[13] J. Weissmann and R. Salomon, "Gesture recognition for virtual reality applications using data gloves and neural networks," in *Neural Networks, 1999. IJCNN'99. International Joint Conference on*, 1999, pp. 2043-2046.

[14] S. Tamura and S. Kawasaki, "Recognition of sign language motion images," *Pattern Recognition*, vol. 21, pp. 343-353, 1988.

[15] C. Charayaphan and A. Marble, "Image processing system for interpreting motion in American Sign Language," *Journal of Biomedical Engineering*, vol. 14, pp. 419-425, 1992.

[16] T. Starner, J. Weaver, and A. Pentland, "Real-time american sign language recognition using desk and wearable computer based video," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 20, pp. 1371-1375, 1998.

[17] J. Rekha, J. Bhattacharya, and S. Majumder, "Shape, texture and local movement hand gesture features for indian sign language recognition," in *Trendz in Information Sciences and Computing (TISC), 2011 3rd International Conference on*, 2011, pp. 30-35.

[18] M. Lamari, M. S. Bhuiyan, and A. Iwata, "Hand alphabet recognition using morphological PCA and neural networks," in *Neural Networks, 1999. IJCNN'99. International Joint Conference on*, 1999, pp. 2839-2844.

[19] C. Vogler and D. Metaxas, "A framework for recognizing the simultaneous aspects of american sign language," *Computer Vision and Image Understanding*, vol. 81, pp. 358-384, 2001.

[20] Y.-H. Lee and C.-Y. Tsai, "Taiwan sign language (TSL) recognition based on 3D data and neural networks," *Expert systems with applications*, vol. 36, pp. 1123-1128, 2009.

[21] (2 August). *Xtion PRO LIVE (4001)*. Available: http://us.estore.asus.com/index.php?l=product_detail&p=3397

[22] (2 August). *Kinect for Xbox 360*. Available: http://www.microsoftstore.com/store/msusa/en_US/pdp/Kinect-for-Xbox-360/productID.253169000

[23] Z. Zhang, "Microsoft kinect sensor and its effect," *Multimedia, IEEE*, vol. 19, pp. 4-10, 2012.

[24] iPiSoftWiki. (2013, 12 August). *Depth Sensors Comparison*. Available: http://wiki.ipisoft.com/Depth_Sensors_Comparison
[25] (2013, 28 August). *Creative announces the Senz3D™ interactive gesture camera*. Available: http://www.creative.com/corporate/pressroom/?id=13361
[26] I. Corporation, "Intel® Perceptual Computing SDK Human Interface Guidelines," 2013.
[27] Microsoft. (28 August). *Kinect for Windows SDK*. Available: http://msdn.microsoft.com/en-us/library/hh855347.aspx
[28] Microsoft. (28 August). *Kinect for Windows Sensor Components and Specifications*. Available: http://msdn.microsoft.com/en-us/library/jj131033.aspx
[29] C. T. Ltd, "InteractIve Gesture camera for your PC," 2013.
[30] O. D. Team. (2013). *Open Source Computer Vision Library*. Available: http://opencv.org/about.html
[31] T. L. team. (2012). *CLAPACK (f2c'ed version of LAPACK)*. Available: http://www.netlib.org/clapack/
[32] Intel, "Intel®Perceptual Computing SDK Reference Manual," 2013.
[33] X. Tan and B. Triggs, "Enhanced local texture feature sets for face recognition under difficult lighting conditions," in *Analysis and Modeling of Faces and Gestures*, ed: Springer, 2007, pp. 168-182.
[34] D.-Y. Huang, W.-C. Hu, and S.-H. Chang, "Vision-based hand gesture recognition using PCA+ Gabor filters and SVM," in *Intelligent Information Hiding and Multimedia Signal Processing, 2009. IIH-MSP'09. Fifth International Conference on*, 2009, pp. 1-4.
[35] W. Zhang, S. Shan, W. Gao, X. Chen, and H. Zhang, "Local Gabor binary pattern histogram sequence (LGBPHS): A novel non-statistical model for face representation and recognition," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, 2005, pp. 786-791.
[36] K. Yan, Y. Chen, and D. Zhang, "Gabor Surface Feature for face recognition," in *Pattern Recognition (ACPR), 2011 First Asian Conference on*, 2011, pp. 288-292.
[37] C. Liu and H. Wechsler, "Gabor feature based classification using the enhanced fisher linear discriminant model for face recognition," *Image processing, IEEE Transactions on*, vol. 11, pp. 467-476, 2002.
[38] K. I. Diamantaras and S. Y. Kung, *Principal component neural networks*: Wiley New York, 1996.
[39] K. Fukunaga, *Introduction to statistical pattern recognition*: Access Online via Elsevier, 1990.
[40] R. A. Fisher, "The use of multiple measures in taxonomic problems," *Ann. Eugenics*, vol. 7, pp. 179–188, 1936.
[41] "UCI Machine Learning Repository," ed. University of California, School of Information and Computer Science: Bache, K.

Lichman, M., 2013.
[42] J. W. Eaton. (18 October). *GNU Octave*. Available: http://www.gnu.org/software/octave/about.html
[43] H. Anton, *Elementary linear algebra*: Wiley. com, 2010.
[44] J. Fridrich, M. Goljan, and R. Du, "Lossless embedding of data in digital objects," ed: Google Patents, 2006.
[45] T. M. Sezgin and R. Davis, "HMM-based efficient sketch recognition," in *Proceedings of the 10th international conference on Intelligent user interfaces*, 2005, pp. 281-283.
[46] C. C. Tappert, "Cursive script recognition by elastic matching," *IBM Journal of Research and development*, vol. 26, pp. 765-771, 1982.
[47] J. A. Pittman, "Recognizing handwritten text," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 1991, pp. 271-275.

[48] J. O. Wobbrock, A. D. Wilson, and Y. Li, "Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes," in *Proceedings of the 20th annual ACM symposium on User interface software and technology*, 2007, pp. 159-168.

[49] A. Kurakin, Z. Zhang, and Z. Liu, "A real time system for dynamic hand gesture recognition with a depth sensor," in *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European*, 2012, pp. 1975-1979.