國 立 交 通 大 學

電控工程研究所

碩 士 論 文

基於影像特徵點擷取結合深度資訊之即時手勢辨識系統設計

Real-Time Hand Gesture Recognition System Design Based on Image Feature Points Extraction and Depth Information

研 究 生：吳仕政

指導教授：陳永平　教授

中 華 民 國 一 百 零 二 年 六 月

基於影像特徵點擷取結合深度資訊之即時手勢辨識系統設計

Real-Time Hand Gesture Recognition System Design Based on Image Feature Points Extraction and Depth Information

研 究 生：吳仕政　　Student：Shi-Cheng Wu

指導教授：陳永平　　Advisor：Professor Yon-Ping Chen

國 立 交 通 大 學

電控工程研究所

碩 士 論 文

A Dissertation
Submitted to Institute of Electrical Control Engineering
College of Electrical and Computer Engineering
National Chaio Tung University
In Partial Fulfillment of the Requirements
For the Degree of Master
In
Electrical Control Engineering
June 2013
Hsinchu, Taiwan, Republic of China

中華民國一百零二年六月

基於影像特徵點擷取結合深度資訊之即時手勢辨識系統設計

學生：吳仕政　　指導教授：陳永平 教授

國立交通大學電控工程研究所

摘 要

近年來，手勢辨識可應用的領域相當廣泛，因此受到重視且被深入的研究與探討，例如人機溝通、遠距遙控等皆是。一般而言，手勢辨識系統先根據手勢模型找出其特徵，再利用這些特徵來做辨識。本篇論文提出以 Kinect 之彩色及深度影像為基礎的手勢特徵擷取，來設計即時手勢辨識系統。整個系統分成三個部分：前景偵測、特徵擷取及手勢識別。首先利用膚色偵測配合聯通物件法濾掉背景並找出可能的手勢範圍；之後藉由距離轉換並尋找距離轉換區域的最大值來萃取特徵點，進而利用特徵點找出手勢特徵，包括手的方向、掌心位置、指尖位置及指向。最後，設計以手勢特徵為依據之即時手勢辨識系統，從實驗結果可知，本論文所提之手勢辨識系統確實可以達成具有成效之即時辨識功能。

Real-Time Hand Gesture Recognition System Design Based on Image Feature Points Extraction and Depth Information

Student：Shi-Cheng Wu Advisor：Prof. Yon-Ping Chen

Institute of Electrical Control Engineering

National Chiao-Tung University

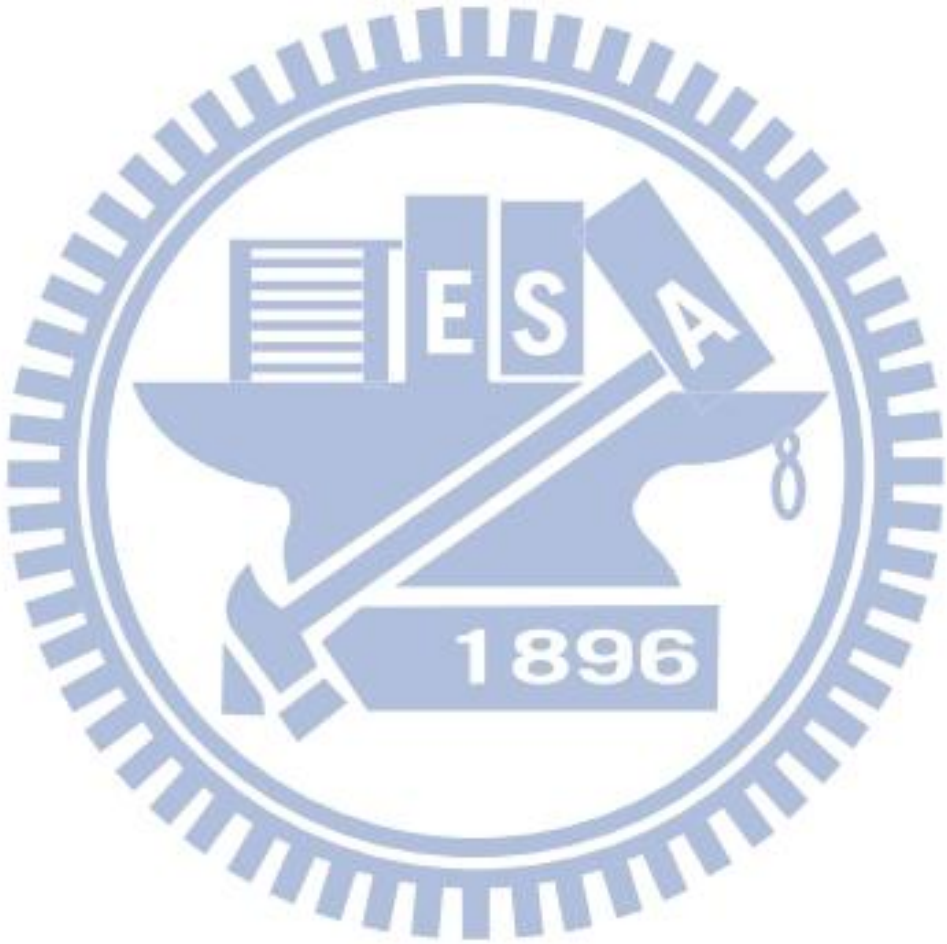
ABSTRACT

In recent years, hand gestures recognition(HGR) approaches have been widely applied to a diversity of areas, like human-computer interface(HCI) and remote control systems. The HGR systems usually rely on a hand model to extract useful hand gesture features. This thesis proposes a robust and fast feature extraction method for hand gesture based on the depth and RGB information from Kinect to implement a real-time HGR system. The system is divided into three parts, including region-of-interest (ROI) selection, feature extraction and hand gesture recognition. First, the skin color detection and connected component labeling(CCL) are applied to select the potential ROIs. Then, pixels with local maximum distance-transformation-value in the potential ROIs can be extracted as feature points. Further, these feature points could be used to find hand gesture features such as hand orientation, palm center and fingertip positions and directions. Finally, the extracted hand gesture features are send into the HGR system. From the experimental results, the proposed hand gesture recognition system can perform in real-time and possess good recognition rates.

Contents

List of Figures

Index of Tables



Chapter 1

Introduction

1.1 Preliminary

In recent years, the hand gesture recognition(HGR) becomes a popular subject since it provides a natural and intuitive communication for human-computer interaction(HCI). There are many applications using HGR such as video games and robot control. However, HGR is a complex issue because our hands consist of many connected joints and high order degree of freedoms(DOFs). To reach a real-time HCI via hand gestures, it has to meet some requirements such as computation time, accuracy and robustness against different background. Earlier HGR researches used data gloves or makers to make image processing easier and more accurate [15], but they are not convenient for users and thus recent investigators pay more attentions to recognizing bare hand gestures without the aid of any gloves or makers.

The HGR techniques can be classified into three categories: color-based algorithms, appearance-based algorithms, and 3-D hand model-based approaches. Color-based algorithms directly apply the image color to model the visual appearance of the hand by searching the skin colored regions in the image [1]. Although it has a good real-time performance, it's very sensitive to lighting condition and often results in false detection of other skin-like objects. As to the appearance-based algorithms, they compare these parameters of the selected features related to visual appearance or mathematical transformation of hands [4-11]. Most of the appearance-based methods are pixel based, so the computation cost is usually too high to implement in real time systems. For the 3-D hand model-based approaches, they depend on the 3-D

kinematic hand model with a large scale of DOFs and estimate the hand parameters by comparing the difference between input images and possible 2-D appearance projected by a 3-D hand model [3,16]. This model generally provides a rich information to recognize a wide class of hand gestures, but a huge image database is required to cover all the possible shapes under different views. Meanwhile, matching every input frame image with the whole image database will lead to a large amount of time consumption in computation. In this thesis, a color-based algorithms is used to select the image ROI.

In general, the system could be separated into three stages: foreground segmentation, feature extraction, and hand gestures recognition. Foreground segmentation is implemented to select the region of interest(ROI) and filter out the background region. Therefore, the remaining search region would be reduced such that the processing speed could be highly enhanced. Furthermore, the appropriate features would be extracted to distinguish different gestures efficiency and correctly. Finally, the selected features would be used as the input of hand gestures recognition system to get the final result. The related methods would be introduced in the following sections.

1.2 System Overview

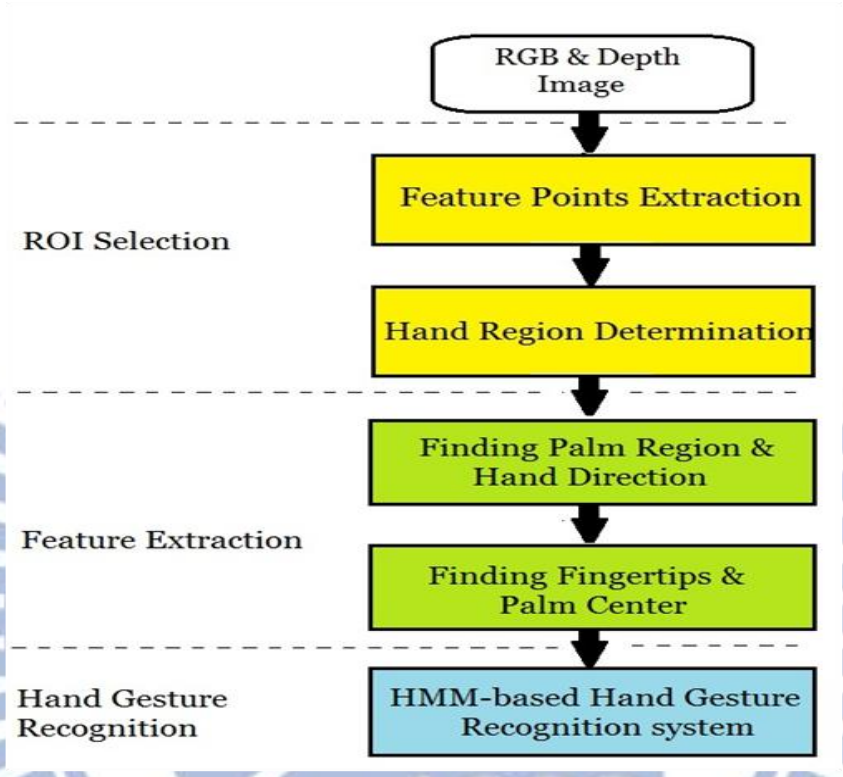
For the hardware architecture, our system uses Xbox Kinect as the image sensor, including RGB image and depth information, and Table 1.1shows the specification of Kinect. In the depth image, the pixel with lower intensity indicates that the distance between object and camera is smaller, and all the points are set to 0 if the sensor is not able to measure their depth. The image captured by Kinect would be delivered into Personal Computer(PC) and then be processed to implement hand gesture recognition. The specification is Intel® Core™ i5-3210M CPU @2.50GHz, 8GB memory, and Windows 7 operation system. The frame rate is about 30 frames per second and the frame is processed using C/C++ and Matlab.

Table 1.1 Specification of Kinect

	Effective Range
Depth sensor range	1m ~ 4m
Field of view	Horizontal field of view: 57 degrees Vertical field of view: 43 degrees
Physical tilt range	±27 degrees
Data stream	320×240 16-bit depth @ 30 frames/sec 640×480 32-bit color @ 30 frames/sec

For the software architecture, Fig- 1.1 is the flowchart of the proposed system. At first, the system receives the RGB and depth images from Kinect and then selects the region-of-interest (ROI). After ROI selection, the feature extraction is implemented. At the final stage, the overall features are delivered into the hand gesture recognition system to distinguish different gestures. The experimental environment is our laboratory and the Kinect camera is at about 100cm height.

Moreover, there are two limitations when implementing the system: first, the detection distance is between 0.5m to 2m because of the hardware limitation of Kinect. Second, the system only focus on one-user implementation.



1.1 Software architecture

Chapter 2 describes the related works of the system. Chapter 3 introduces the proposed system in detail. Chapter 4 shows the experimental results. Chapter 5 is the conclusions of the thesis and the future works.

Chapter 2

Related Works

2.1 Hand Gesture Recognition Method

In recent years, many hand gestures recognition approaches have been proposed. In general, the overall methods could be roughly divided into three categories: color-based algorithms, appearance-based algorithms, and 3-D hand model-based approaches. 3-D hand model-based approach [3,16] relies on the 3-D kinematic hand model with a large scale of DOF and calculates the hand parameters by comparing the difference between the input images and the possible 2-D appearance projected by the 3-D hand model. This approach is suitable for the realistic interactions in virtual environments since it provides a rich information to describe different type of gestures. However, the major drawback is that requires a huge image database to deal with the entire characteristic shapes under different views.

The skin color are the image features that are frequently used for hand detection[1]. However, color-based algorithms face the difficult task that human arm and face have the similar color with hands. This methods also very sensitive to the lighting conditions. So when the lighting does not satisfy some environment requirements, this algorithm usually perform not well.

For the appearance-based algorithms, shape descriptors are used to represent the hand shape. In [4] , Fourier descriptor and Zernike moments are used to recognize hand gesture, but the computational cost is too high since it's pixel based algorithm. [5] obtained the integrated hand contour and then computed the curvature of each point

on the contour, but the noise and the unstable illumination in the segmented background are the major problems for this method. The eigenspace is another technique, which provides an efficient representation using a set of basis vectors, but this method is not invariant to translation, scaling, and rotation. For the reason, a number of researchs on local invariant features, such as SIFT and Haar-like features, are proposed. In [6], SIFT features are used to achieve rotation invariant hand detection. The authors of [7] used Haar-like features, which describe the ratio between dark and bright regions rather than single pixel value, to reach a hand gestures recognition, but the computation cost makes it hard to implement on real-time systems.

Another appearance-based approaches focus on building a hand model containing the palm and finger structures. [8] determined the fingertip positions and the center of the palm using the first moment of the 2-D probability distribution that based on the contour of the hand segmented region. [9] detected the fingertips using the momentum of the skin color region and used the Kalman filter to reach a robust finger tracking. Since there are a number of pixels on the contour or edges, so the computation cost for both [8] and [9] are considerably high. Template matching can also be used to detect fingertips and palms [10] , but the distance from camera to hand cannot be changed and a good hand segmentation results are also required. [11] proposed a method that combines particle filtering with particles random diffusion. However, the cost of this method is still too high to implement.

2.2 Skin Color Segmentation

This is an important step to extract all skin pixels from the Kinect sensor. Most skin color segmentation methods are based on the statistical approach, which can be divided into two categories: parametric and non-parametric. The parametric statistical approach, such as Gaussian model, Gaussian-mixture model, etc., represents the probability density function(PDF) of skin-color distribution in a parametric form. The main advantages of this approach are lesser computation time and no training data to be saved. However, a suitable parametric order, especially for Gaussian-Mixture model case, is hard to determined and may not fit the real skin-color distribution. Non-parametric approach uses quantized histogram to represent the PDF, or uses training data to estimate the skin-color density function, such as the kernel method or support vector machine(SVM). Although this approach can be evaluated intuitively and adequate to different training data sets, a major drawback is that it requires to keep a larger amount training data and costs a relatively high computation time. Consequently, model selection is a trade-off between proper fitting and computation complexity.

The choice of color space is considered as the primary step in skin-color segmentation. The statistical approaches usually use a suitable color space to reduce the effect of varying luminance and decrease the overlap between skin and non-skin pixels. Usually, HSV and YCbCr are most popular color space for skin detection due to the robustness of varying illuminant and the minimum overlap between skin-color and background-color [12] .

The parameters of the GMM can be obtained from the training data through the iterative expectation-maximization (EM) technique [13] . After proper parameter

estimation, both conditional probability densities for skin and non-skin colors are obtained, denoted as $p(X \mid \text{skin})$ and $p(X \mid \text{nonskin})$, where $X = [\text{Cr Cb}]^T$. Given this class conditional probabilities of skin and non-skin models, a skin classifier can be built using Bayes classifier [14] . The classification boundary is determined where the likelihood ratio of $p(X \mid \text{skin})$ and $p(X \mid \text{nonskin})$ exceeds some threshold based on the ROC(receiver operating characteristics) curve. That is, for a given image pixel $\boldsymbol{x}_n = [\text{Cr}(n)\ \text{Cb}(n)]^T$, it is classified as skin when it satisfies:

$$\frac{p(skin \mid \boldsymbol{x}_n)}{p(nonskin \mid \boldsymbol{x}_n)} = \frac{p(\boldsymbol{x}_n \mid skin)\, p(skin)}{p(\boldsymbol{x}_n \mid nonskin)\, p(nonskin)} > K \tag{2.1}$$

where K is a constant and $p(skin) = 1 - p(nonskin)$. Rearranging (2.1), it becomes:

$$\frac{p(\boldsymbol{x}_n \mid skin)}{p(\boldsymbol{x}_n \mid nonskin)} > K \frac{p(nonskin)}{1 - p(nonskin)} \triangleq K' \tag{2.2}$$

The threshold K' is usually determined from the ROC curve, which shows the relationship between the true positives and false positives. The Bayes classifier has been widely used for skin segmentation since its simplicity and less computation time. We only need to compute the likelihood ratio in (2.2) and check whether it is larger than the threshold K'.

2.3 Classification From Linear Model to ANNs

The goal of classification problem is to assign an input data $\boldsymbol{x}_n \in X$, $n = 1,2,...,N$, in the database, to one of the K classes C_k, $k = 1,2,..,K$. Usually, these classes are disjoint such that $\boldsymbol{x}_n$ is classified into at most one (or none of them). To simplify the discussion in this section, here we only consider the model for two-class classification problems.

2.3.1 Linear Model

To determine which class an input $\boldsymbol{x}_n$ belongs to, the simplest decision function for two-class classification problems is usually given as the following linear model:

$$y = \boldsymbol{w}^T \boldsymbol{x}_n + w_0 = \sum_j w_j x_{nj} + w_0 \qquad (2.3)$$

where $\boldsymbol{w}$ is called the weighting vector and w_0 is the bias. Based on the linear model (1), first choose y=0 as the decision boundary, and then classify the inputs satisfying $y \geqq 0$ into the class 1, denoted as C_1, and the other inputs into C_2. An example for the above two-class classification is shown in Fig- 2.1 (a), where the decision boundary is included. However, this linear model does not work well when there exists an overlap between the two classes, such as the XOR problem depicted in Fig- 2.1 (b). To solve this problem, a generalized linear model is proposed.

2.3.2 Generalized Linear Model

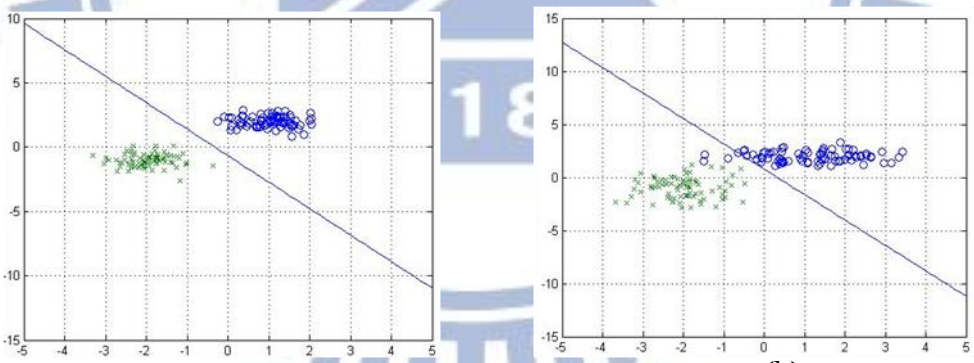
The idea of generalized linear model is to transform the input data $\boldsymbol{x}_n$ in the database X into another space, and then uses the transformed data as a substitute of $\boldsymbol{x}_n$ in (2.3), described as below:

$$y = \boldsymbol{w}^T \boldsymbol{\varphi}(\boldsymbol{x}_n) + w_0 \qquad (2.4)$$

where $\boldsymbol{\varphi}(\cdot)$ is a user defined basis function. In other words, the basis function transforms the database X to the linearly separable feature space. Moreover, an activation function $f(\cdot)$ can be further introduced to modify (2.4) as below:

$$y = f\left(\boldsymbol{w}^T \boldsymbol{\varphi}\left(\boldsymbol{x}_n\right) + w_0\right) = f\left(\sum_j w_j \varphi_j\left(\boldsymbol{x}_n\right) + w_0\right) \tag{2.5}$$

which y represents the predicted discrete class labels, or posterior probability lying in the range [0,1]. However, an adequate basis function $\boldsymbol{\varphi}(\cdot)$ is often difficult to determine. In order to apply the modified model (2.5) to a variety of problems, it is necessary to adapt the basis functions correspondingly. Recently, investigators have successfully adopted the feed-forward neural network to implement (2.5) due to the fact that the neural network uses fixed number of basis functions and allows them to be adjusted during training. The cost of this model is to face the nonconvex optimization problem during training stage which would stocked in the local minimum.



(a) (b)

Fig- 2.1 The examples for (a) linear separable set (b) linear inseparable set

2.3.3 Artificial Neural Network

ANNs have been successfully applied to some complicated problems, such as image analysis, speech recognition, adaptive control, etc. Usually, the ANNs will be adopted to implement human detection via intelligent learning algorithms. The basic

structure of ANNs is called neuron, whose input-output relationship is shown in Fig-2.2 and described as

$$z = h(\boldsymbol{w}^T \boldsymbol{x} + w_0) = h(\sum_{i=1}^{D} w_i x_i + w_0) \tag{2.6}$$

which has the same form as (1) with the output z and the input $\boldsymbol{x} = [x_1, x_2, ..., x_D]^T \in \Re^D$. Usually, the activation function $h(\cdot)$ is chosen to be logistic sigmoid function ,linear function, or tangent sigmoid function which are described as below：

(1) Logistic sigmoid function：

$$h(x) = \frac{1}{1+e^{-x}} \tag{2.7}$$

(2) Tangent sigmoid function：

$$h(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.8}$$

(3) Linear function：

$$h(x) = x \tag{2.9}$$

where (2.7) is range [0,1] and (2.8) is [-1,1]. In summary, a linear output is needed for regression problem while use logistic sigmoid output for classification problem.

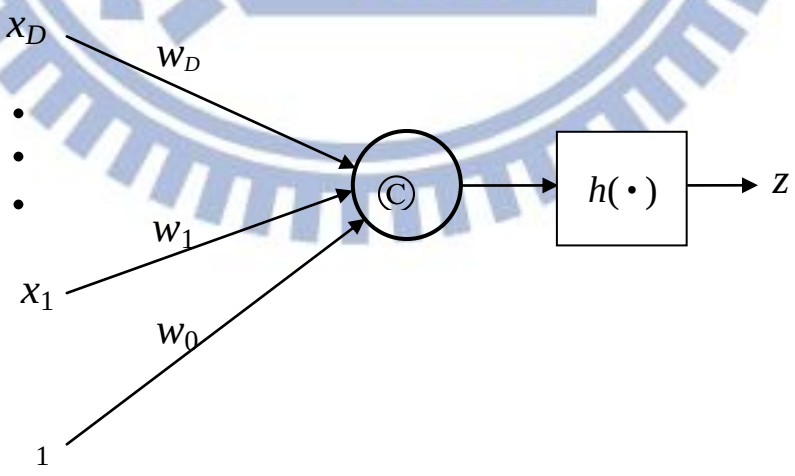


Fig- 2.2 Basic structure of a neuron

A general ANN is composed of one input layer, one output layer, and one hidden layer, as shown in Fig- 2.3. The input layer just receives input signals, so each node in this layer is taken as a buffer. For the other two layers, their nodes are realized by the structure of neuron as depicted in Fig- 2.2. Each node in the second layer, i.e., the hidden layer, has the same form as (2.6) given by

$$z_j = h(\boldsymbol{w}_j^{(1)T}\boldsymbol{x}_n + w_{j0}^{(1)}) = h(\sum_{i=1}^{D} w_{ji}^{(1)} x_{ni} + w_{j0}^{(1)}) \tag{2.10}$$

where j=1,...,D and the superscript (1) indicates that the weights are related to the first layer of ANNs. Similarly, the output of each neuron in third layer, called the output layer, is with the same form as previous:

$$y_k = f(\boldsymbol{w}_k^{(2)T}\boldsymbol{z} + w_{k0}^{(2)}) = f(\sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)}) \tag{2.11}$$

where k=1,...,K. Substitute (2.10) into (2.11), then each output y_k becomes:

$$y_k = f(\sum_{j=1}^{M} w_{kj}^{(2)} h(\sum_{i=1}^{D} w_{ji}^{(1)} x_{ni} + w_{j0}^{(1)}) + w_{k0}^{(2)}) \tag{2.12}$$

Comparing with (2.5), ANN use this structure to find an adequate basis function $\varphi(\cdot)$ adaptive to the training data. As expected, ANNs are indeed able to deal with more complicated problems compared to generalized linear model.

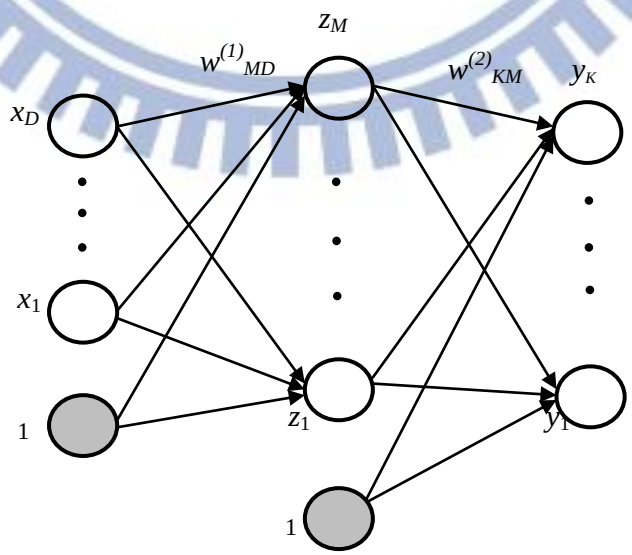


Fig- 2.3 ANN structure

2.4 Classification for Sequential Data

The learning algorithms mentioned before such as linear model and ANN are primarily focused on the independent and identically distributed(i.i.d.) observations. The i.i.d. assumption allows us to express the likelihood function of all observations as:

$$p(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N \mid \theta) = p(\boldsymbol{x}_1 \mid \theta)\, p(\boldsymbol{x}_2 \mid \theta) \cdots p(\boldsymbol{x}_N \mid \theta) \tag{2.13}$$

where $\boldsymbol{x}_n$ is the observation at time n, n=1,...,N, and θ is a user-defined parameter. However, this assumption would fail in many applications when the observations are sequential and depend on the previous ones, like speech data, rainfall measurements, etc.. Therefore, it is necessary to consider the correlation among all observations. But in real application, it would be difficult to find a general dependence of present observation to all previous observations since the complexity would grow as time increases. To solve this problem, Markov model is proposed under the assumption that the present data $\boldsymbol{x}_n$ only depends on the previous one $\boldsymbol{x}_{n\text{-}1}$.

2.4.1 Markov Model

Given N observations $\{\boldsymbol{x}_1, ..., \boldsymbol{x}_N\}$, we can always use the product rule to express the joint distribution for this sequence of observations as below:

$$\begin{aligned} p(\boldsymbol{x}_1, ..., \boldsymbol{x}_N) &= p(\boldsymbol{x}_1)\, p(\boldsymbol{x}_2 \mid \boldsymbol{x}_1)\, p(\boldsymbol{x}_3 \mid \boldsymbol{x}_1, \boldsymbol{x}_2) \cdots p(\boldsymbol{x}_N \mid \boldsymbol{x}_1, ..., \boldsymbol{x}_{N-1}) \\ &= p(\boldsymbol{x}_1) \prod_{n=2}^{N} p(\boldsymbol{x}_n \mid \boldsymbol{x}_{n-1}, ..., \boldsymbol{x}_1) \end{aligned} \tag{2.14}$$

where Fig- 2.4 shows the fully dependent sequence, that is, the observation $\boldsymbol{x}_n$ at time n is correlated to all the previous $n-1$ observations. Assume that each of the conditional distribution only depends on the previous one, then the n^{th} conditional

distribution in (2.14) can be simplified as:

$$p(\boldsymbol{x}_n \mid \boldsymbol{x}_1, ..., \boldsymbol{x}_{n-1}) = p(\boldsymbol{x}_n \mid \boldsymbol{x}_{n-1}), \quad n = 1, 2, ..., N \tag{2.15}$$

Therefore, the joint distribution of N observations is rewritten as

$$p(\boldsymbol{x}_1, ..., \boldsymbol{x}_N) = p(\boldsymbol{x}_1) \prod_{n=2}^{N} p(\boldsymbol{x}_n \mid \boldsymbol{x}_{n-1}) \tag{2.16}$$

which is known as the first-order Markov chain and shown in Fig- 2.5. In most practical applications, all the N–1 conditional distributions are assumed to be equal, that is, the relation between any adjacent observations is the same. The model with this property is then called the homogeneous first-order Markov model. We can use a higher order Markov chain to provide more information from the previous samples, but it would need a large amount of computation time as the order increased. Rather than increasing the number of order, we can provide a rich information of the Markov model by introducing additional latent variables, which forms the so-called hidden Markov model (HMM) and has been widely used to solve the complicated classification problems.

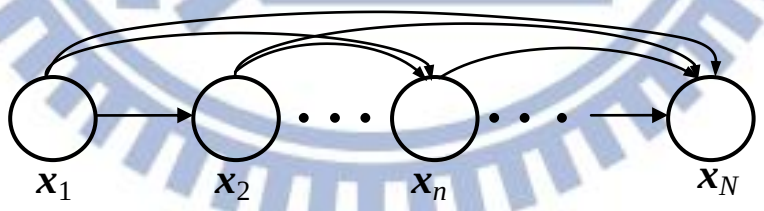


Fig- 2.4 Sequence of fully dependent observations

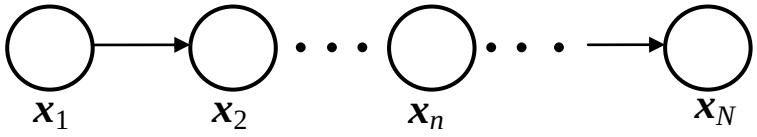


Fig- 2.5 Sequence of first-order Markov chain

2.4.2 Hidden Markov Model

The HMM depicted in Fig- 2.6 includes 1st-order Markov chain $Z=\{\boldsymbol{z}_1,...,\boldsymbol{z}_N\}$ where $\boldsymbol{z}_n$ is the latent variable, or called hidden state, corresponding to the observation data $\boldsymbol{x}_n$, $n=1,...,N$. It is obvious that $\boldsymbol{z}_n$ is only related to the previous latent variable $\boldsymbol{z}_{n-1}$ and the resulted joint distribution is given by:

$$p(\boldsymbol{x}_1,...,\boldsymbol{x}_N,\boldsymbol{z}_1,...,\boldsymbol{z}_N) = p(\boldsymbol{z}_1)\,p(\boldsymbol{x}_1|\boldsymbol{z}_1)\,p(\boldsymbol{z}_2|\boldsymbol{z}_1)\cdots p(\boldsymbol{z}_N|\boldsymbol{z}_{N-1})\,p(\boldsymbol{x}_N|\boldsymbol{z}_N)$$

$$= p(\boldsymbol{z}_1)\prod_{n=2}^{N} p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1})\prod_{n=1}^{N} p(x_n|z_n) \qquad (2.17)$$

where $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1})$, $p(\boldsymbol{x}_n|\boldsymbol{z}_n)$ are called the transition probability and the emission probability, respectively. Nowadays, the HMM has been widely used in many territories, such as speech recognition, natural language modeling, on-line handwriting recognition, and HGR[15][16].

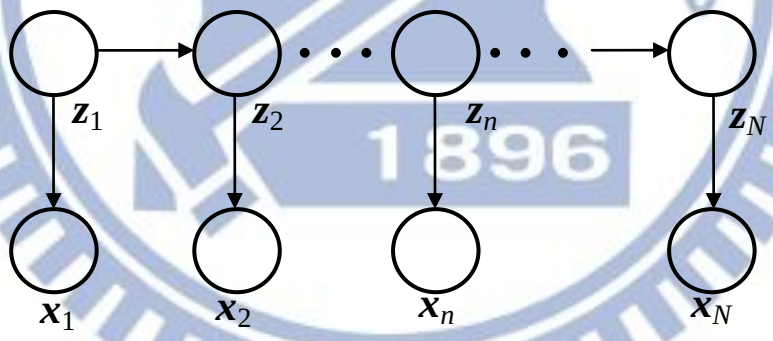


Fig- 2.6 Sequence of hidden Markov model

In real-time applications, there are two problems that must be solved for the use of HMM in (2.17) listed as below:

Problem1: Given N observations $X=\{\boldsymbol{x}_1,...,\boldsymbol{x}_N\}$ and all the model parameters $p(\boldsymbol{z}_1)$, $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1})$, and $p(\boldsymbol{x}_n|\boldsymbol{z}_n)$, for $n=1,...,N$, how to compute the posterior probability of each latent variable, $p(\boldsymbol{z}_n|X)$?

Problem2: Given N observations $X=\{\boldsymbol{x}_1,...,\boldsymbol{x}_N\}$, how to estimate the parameters

$p(\mathbf{z}_1)$, $p(\mathbf{z}_n|\mathbf{z}_{n-1})$, and $p(\mathbf{x}_n|\mathbf{z}_n)$ for n=1,...,N?

For the first problem, given the observations and all model parameters, (2.17) can be obtained and use the property described below:

$$p(Z \mid X) \propto p(Z, X) \tag{2.18}$$

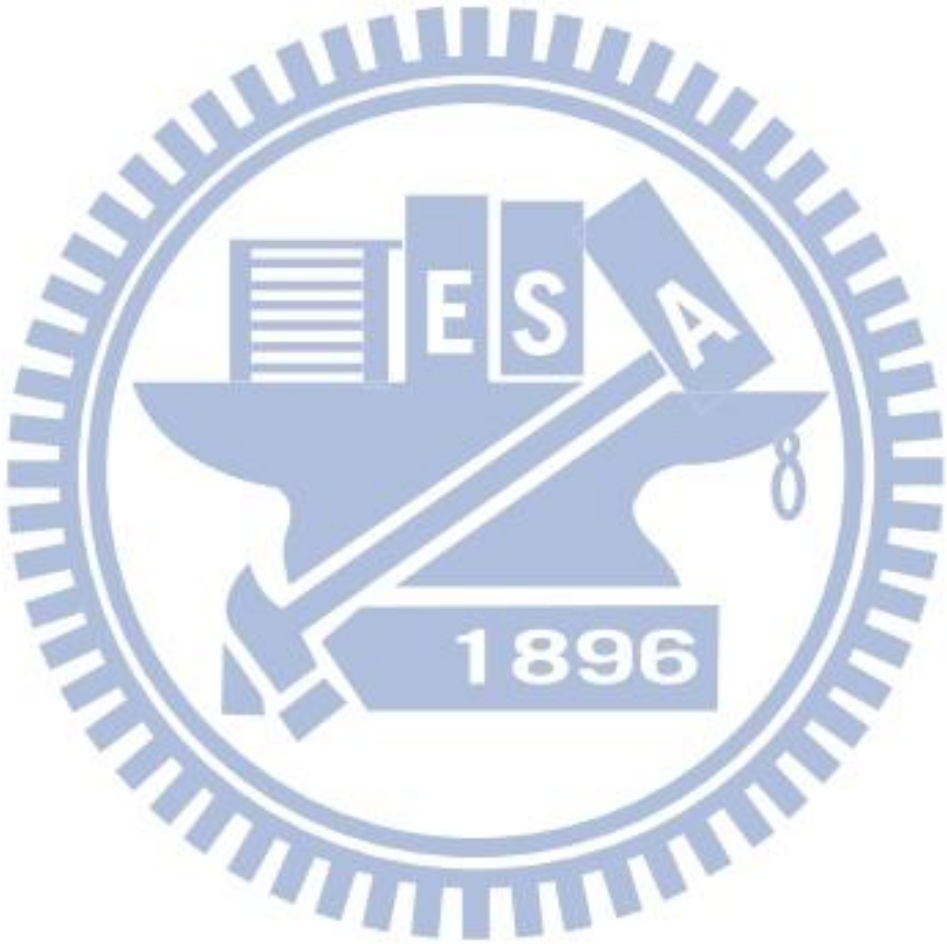
where X ={ $\mathbf{x}_1$,..., $\mathbf{x}_N$} and Z={ $\mathbf{z}_1$,..., $\mathbf{z}_N$}. Using the probability sum rule, the posterior can be intuitively calculated as:

$$p(\mathbf{z}_n \mid X) \propto p(\mathbf{z}_n, X) = \sum_{\mathbf{z}_1} \cdots \sum_{\mathbf{z}_{n-1}} \sum_{\mathbf{z}_{n+1}} \cdots \sum_{\mathbf{z}_N} p(\mathbf{z}_1, \ldots, \mathbf{z}_N, X) \tag{2.19}$$

for n=1,...,N. It's an intuitive way to find the posterior probability, but it would need a vary large-scale computation time. Assume each latent variable $\mathbf{z}_n$ is M-state discrete random variable, (2.19) would need N-1 times of M summation calculation and the computation order is $O(M^{N-1})$! Due to this reason, a more efficient approach, the forward-backward algorithm[17] is used to reduce the computation time and the detail derivation is described in Appendix. A. To further speed up the calculation, the Viterbi algorithm is proposed that finding the most probable sequence of latent variables for a given observations X ={ $\mathbf{x}_1$,..., $\mathbf{x}_N$}. These two methods are based on the well-known model parameters, but in real applications, these parameters are unknown and should be proper estimated given observations.

Problem 2 is a more difficult than previous since it need to determine a method to adjust the model parameters such that the likelihood function of the observations given model parameters, $p(X|\theta)$, where θ is model parameter, is maximum. However, given any observation sequence as training data, there's no close-form solution for estimating model parameters. Instead, an iterative procedure known as Baum-Welch method[18] or expectation-maximization (EM) algorithm is used to maximize the

likelihood function and obtain optimal parameters. Different initial values for the parameters would cause different result, so an suitable initial guess for these values is an important step when using EM algorithm.



Chapter 3

Hand Gesture Recognition System

3.1 Hand Region Detection

It's important to detect the exact hand region in real-time applications. A number of research has been proposed to extract the hand region precisely, such as the Adaboost learning algorithm [16] and SIFT features [17], but the cost of the computation time make it hard to implement on the real-time detection system. Since the extraction of skin color regions is fast, this method is usually used in hand detection even though it is very sensitive to lighting conditions. Based on skin region, the system could implement ROI selection with connected component labeling (CCL) to increase detection rate. To further distinguish the hand and face regions, the scheme proposed in [19] is adopted.

3.1.1 Skin Color Extraction

With the use of Kinect, a suitable distance range from the user to the screen can be selected and the depth information with lower value implies a smaller distance. Besides, all the points are offset to 0 if the sensor is not able to measure their depth. In our proposed remote hand-gesture control system, it is required that the hands move in the range around 0.5m to 2.0m. There are many skin color extraction methods which have been described in Chapter 2.2. To speed up the processing, we use a look-up-table method in YCbCr color space and the result is shown in Fig- 3.1. There are still a lot of noise pixels in skin color, so a method to remove them is required. As observed, these noise pixels usually have a small region comparing with the hand and

face regions, and thus can be removed by the connected component labeling (CCL) method.

Connected Component Labeling (CCL) [20] is a technique to identify different components and often used to detect connected regions in binary images. This thesis applies a 4-pixel connected component to label interesting regions. Furthermore, in addition to recognizing the connected regions, CCL also compute their areas. If the total number of a connected region is less than a threshold, it will be treated as a noise and then removed. As a result, only large enough connected objects are retained. To further improve the selected CCL objects, the dilation operator [20] is employed to fill the holes of connected components as shown in Fig- 3.2. The next step is to distinguish the hand region based on the feature points searched and extracted in the selected CCL objects.

Fig- 3.1 Skin Color Region.

Fig- 3.2 The final CCL image

3.1.2 Feature Points Extraction

There are many distance transformation (DT) algorithms to transform a binary image into a distance images, which are different in the way distances are computed

[21,22]. In this thesis, two-passed scanning distance transform is used in a binary image for its simplicity and efficiency in calculation, Fig- 3.3(a) given an example. To extract the feature points of distance transform image, a useful characteristics has been introduced in [19] based on two important qualities. First, the skeleton of an object can be extracted by distance transformation, as shown in Fig- 3.3(b), where the skeleton pixels are usually possessed of the local maximum distance transformation value. Second, the distance transformation value of skeleton pixels on the finger is usually not high comparing to the palm region. Based on these information, the system could implement feature points extraction in two steps, which are introduced as following paragraph.

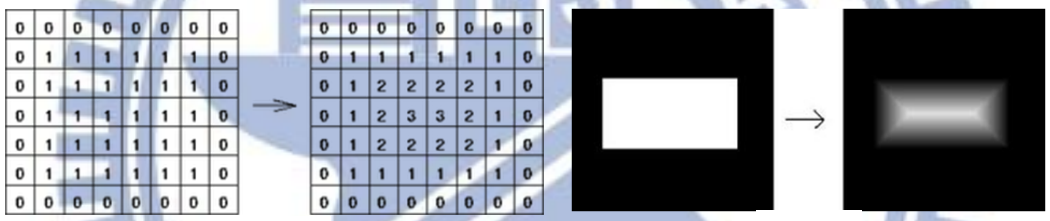


Fig- 3.3 Examples of distance transform (a) Shown its distance value
(b) Shown its skeleton

These distance transformation based feature points can be extracted with the following two steps. First, extract the local maximum pixels based on the distance transformation value. On the distance transformation image, a local maximum pixel would satisfy the following condition:

$$\prod_{i,j} G(x+i, y+j) = 1 \tag{3.1}$$

where $i,j \in \{1,0,-1\}$, and the function G is defined as:

$$G(x+i, y+j) = \begin{cases} 1 & \text{if } D(x,y) \geq D(x+i, y+j) \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

where $D(x,y)$ is the distance transformation value at point (x,y). This condition implies that $D(x,y)$ of the point at (x,y) is greater than or equal to those of its 8 neighborhood pixels, and thus a set of local maximum pixels can be extracted accordingly from the distance transformation image.

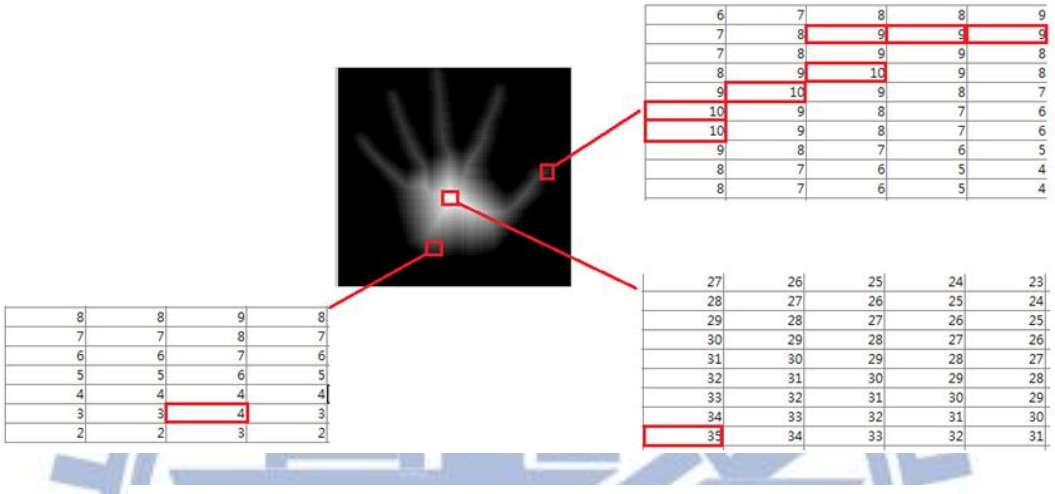


Fig- 3.4 The local maximum distance-based feature pixels in different regions

Second, determine the feature points from the set of local maximum points. In Fig- 3.4, the red-labeled pixels represent the extracted local maximum points. It can be found that those in the middle regions of the palm and arm have bigger $D(x,y)$ from 20 to 50, so are those in the face. While along the contour, the pixels usually possess much lower $D(x,y)$ from 2 to 5. For the local maximum pixels around the finger region, they have the middle distance transformation value satisfying the following condition:

$$10 \geq DV(x, y) \geq 3 \qquad (3.3)$$

Significantly, (3.3) can be used as the condition to extracted the feature points which potentially contain the desired finger region. Fig- 3.5 also shows the extracted feature points, in blue. As shown in Fig- 3.5, we have found out the set of the feature points

still occurs in the wrist and face regions, so the depth information is used to detect the hand region.

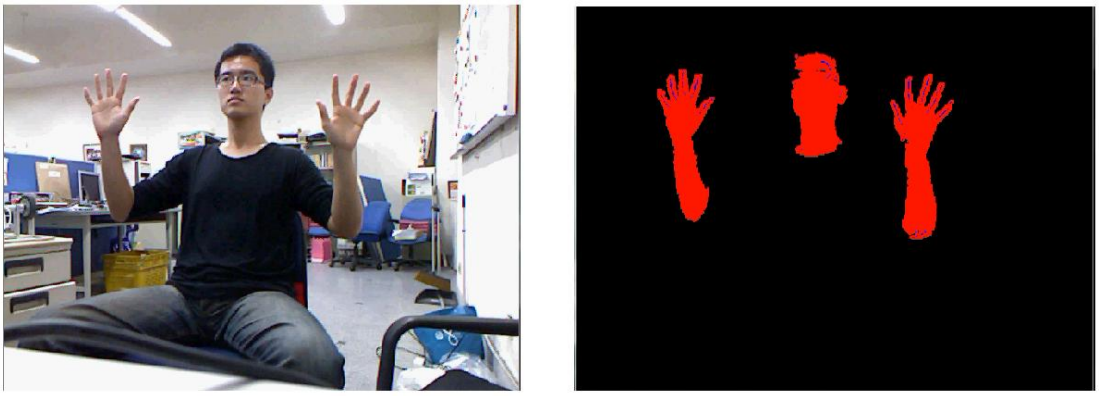


Fig- 3.5 Feature points extraction

3.1.3 Hand Classifier

After several experiment, we have found two important facts: First, under the same depth value, the total number of feature points on the face or other noise region is usually smaller than that on the open hand. Therefore, with the same number of contour, the hand region would have more feature points than that on face. Second, the total number of feature points is effected by the depth value. Due to these two observations, the relationship between the depth and FPs ratio(feature point number/total contour number) is needed and shown in Fig- 3.6, where the 1873 red crosses indicate the region only contain hand and the 1252 green crosses would contain the part of the forearm. The x-axis is the total feature points of the connected component divided by the total contour number while the y-axis is the mean depth of the object. To enhance the processing speed, a Gaussian likelihood model for hand detection is used for this thesis. This model makes the distribution of the hand as a 2-d Gaussian, and the model parameter can be found using the collecting hand training

data. This method provides an intuitive way to determine the similarity between input CCL object and the hand training data.

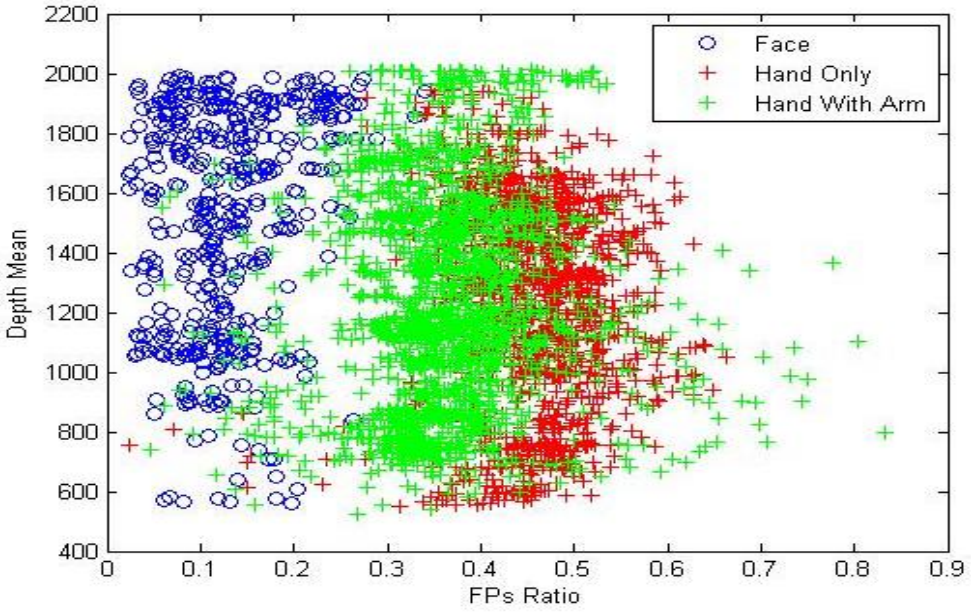


Fig- 3.6 FPs ratio V.S. Depth

To determine connected component object is hand or not, the probability distribution of the hand is defined as follows:

$$p(\boldsymbol{x}|C) \triangleq \left(\frac{1}{2\pi|\Sigma|}\right)^{\frac{1}{2}} exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)\right) \quad (3.4)$$

where $\boldsymbol{x} = [x_1\ x_2]^T$,and x_1 is the FPs ratio and x_2 is the mean depth of the connected component object and C indicates the hand class. The two parametersΣ andμ is covariance matrix and mean vector. Assume the training data are i.i.d., and the likelihood function can be found as below:

$$p(X|C) \triangleq \prod_n p(\boldsymbol{x}_n|C) \quad (3.5)$$

where X is the training data set. Use the maximum likelihood estimation, these two parameters can be obtained and shown as below:

$$\boldsymbol{\mu} = \frac{1}{N}\sum_j \boldsymbol{x}_j \tag{3.6}$$

$$\Sigma = \frac{1}{N}\sum_j \left(\boldsymbol{x}_j - \boldsymbol{\mu}\right)\left(\boldsymbol{x}_j - \boldsymbol{\mu}\right)^T \tag{3.7}$$

where N denotes the total number of training data. The CCL object is classified as hand when (3.4) is granter than a threshold θ. The choice of value θ is a trade-off between the true and false positives, and the ROC curve via different θ is shown in Fig- 3.7. The result shows that the true positive rate can reach 95% while the false probability is less than 1%. To further avoid the false alarm situation, a CCL object is detected as hand when satisfies the following condition:

$$\prod_{m=1}^{M} HF_m = 1 \tag{3.8}$$

where HF_m indicates the hand detection flag at time sequence m and the function is defined as:

$$HF_m = \begin{cases} 1 & \text{if } p\left(\boldsymbol{x}_m \mid C\right) > \theta \\ 0 & \text{otherwise} \end{cases} \tag{3.9}$$

The bigger M would cause a lower false alarm detection but slower down the processing speed. To reach a real-time HCI, M is chosen to be 8 in this thesis that meets both accuracy and processing speed requirements. The detection results shows in Fig- 3.8, where the green rectangle is the selected ROI region. Under a indoor lighting environment, there are not much noises appeared on the detected image and performs very fast. But some errors occur when there are overlaps between the hand and the other skin color object, like face and skin color suits shown as Fig- 3.9. To overcome this problem, a refining process is proposed using the depth distribution information.

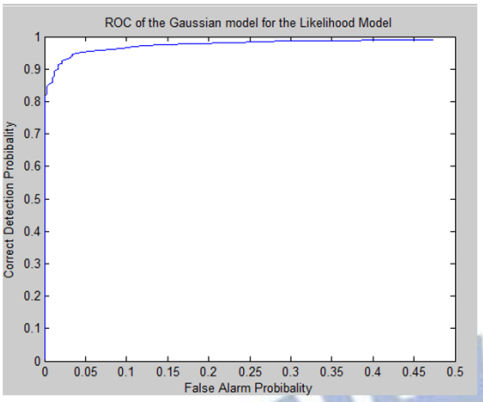


Fig- 3.7 ROC curve under different threshold

Fig- 3.8 The hand detection result.

(a) (b)

Fig- 3.9 The miss detected results where the hand and (a) face (b) skin color suits are overlapping.

3.1.4 Depth Cutting

Some miss detection situation happens when there is a overlapping problem since the CCL algorithm would take the whole connected skin region as a same component. In general, two different skin color object exist in two different depth plane, so a hand detection processing under different depth level can be used to

separate these objects. Based on the depth information of skin color image, the system could implement histogram in three steps, which are introduced as following:

Step 1: After a skin-color extraction on whole image, first use CCL algorithm to filter out the skin color noise which has small area of connected points. This step clear out the small skin color background of the whole depth level.

Step 2: The system computes the depth histogram of the skin color image where the intensity levels is on the range [0, 255], where Fig- 3.10 shows the example. The depth distribution of Fig- 3.10 (b) is shown as Fig- 3.10 (c), which can be divided into two clusters: one is the right hand part that close to the camera; the other is face and the left hand regions. Let the depth histogram of the skin color image be:

$$\boldsymbol{h} = \begin{bmatrix} h_1 & \cdots & h_{255} \end{bmatrix} \tag{3.10}$$

where h_i is histogram number related to intensity i, i=1~255. The k-th depth region cutting would satisfy the following condition:

$$\prod_{j=L_k}^{U_k} h_j = 1 \tag{3.11}$$

where L_k and U_k indicate the lower and upper bound of k-th depth cutting region, respectively. With the aid of the cutting region, we can process skin color detection and CCL operation corresponding to different cutting region such that the overlapping skin color regions can be separated.

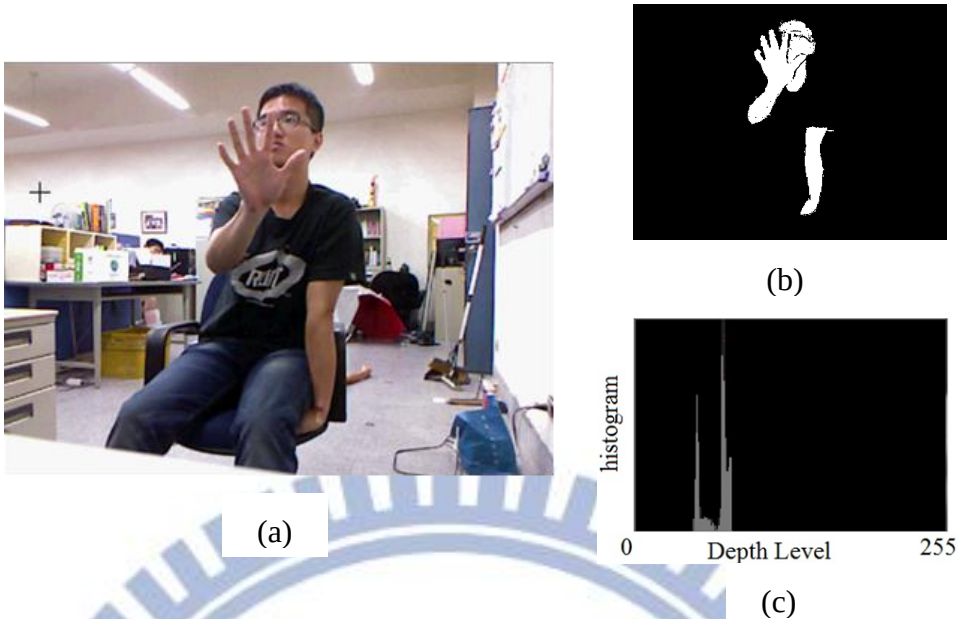


Fig- 3.10 (a) The original image. (b) The skin color region after CCL thresholding (c) Depth histogram of image.

Step 3: Assume there are *K* depth searching regions, the selected depth search level is determined by the following algorithm:

$$
\begin{aligned}
&for\ k = 1 \sim K \\
&\quad if\ \ U_k - L_k < 2 \\
&\quad\quad continue; \\
&\quad else\ if\ \ L_k + 7 < U_k \\
&\quad\quad Search\ Level = [L_k, L_k + 7]; \\
&\quad else \\
&\quad\quad Search\ Level = [L_k, U_k]; \\
&end
\end{aligned}
$$

This algorithm stats that we will skip searching when the depth continuous distribution region is too small, and the maximum depth search interval is 7 since the hand is usually the closest object to the camera. The system then implements the feature points extraction and hand classifier respect to different depth cutting regions. Using the refining step can solve this overlapping problem and the Fig- 3.11 shows the result.

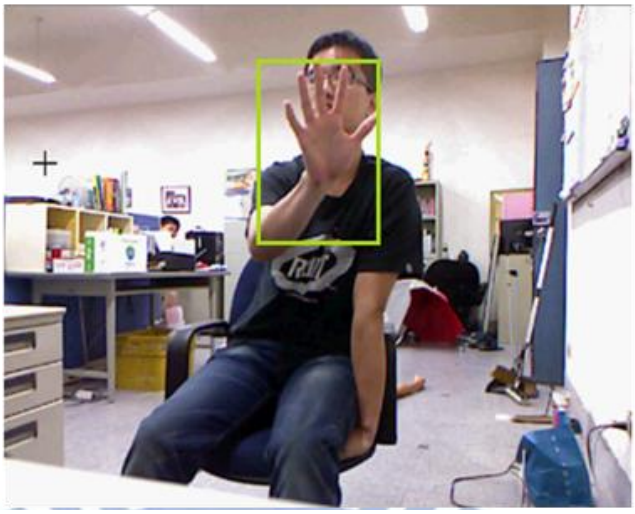


Fig- 3.11 The correct detection result even when hand and face are overlapping.

Using the depth cutting refining can overcome the overlapping problem, and once it has been detected, the searching region is determined around the depth of the detected hand. However, the detected hand region still include the forearm part, which is not helpful for the HGR. So the next step is the forearm cropped procedure and find useful hand features to further increase the processing speed.

3.2 Hand Feature Extraction

After hand region detection, the system has to extract useful features to increase the detection rate and decrease the computational cost. Based on the distance-based feature points extracted from the previous section, a hand feature with the center of the palm, the hand direction and the five fingertip positions can be extracted fast and precisely. These features are very important parameters for both gesture recognition and hand tracking.

3.2.1 Hand Direction and Hand Region

The direction of the hand is an important cue for recognition. With the usage of the feature point and the binary hand region, this feature can be obtained in these two steps. First, the CCL center can be obtained by calculating the central moment on a binary segmented hand image $I(x,y)$, where the $I(x,y) \in \{0,1\}$. Then the center point $(x_{CCLMean}, y_{CCLMean})$ can be determined using:

$$x_{mean} = \frac{\sum_{x} x \sum_{x,y} I(x,y)}{\sum_{x,y} I(x,y)}, y_{mean} = \frac{\sum_{x} y \sum_{x,y} I(x,y)}{\sum_{x,y} I(x,y)} \quad (3.12)$$

Second, using the same equation (3.12), calculate the feature points mean, denote as (x_{FPMean}, y_{FPMean}). The direction is the line the connects the hand center and feature points mean, and the degree of the hand direction is determined by using the following function:

$$HD° = arctan2\left(\frac{y_{CCLMean} - y_{FPMean}}{x_{CCLMean} - x_{FPMean}}\right) \quad (3.13)$$

where *arctan2* function returns the positive angle for counter-clockwise angles (upper half-plane, $y > 0$), and negative for clockwise angles (lower half-plane, $y < 0$). Fig-

3.12 shows the result, where the blue, green, and black pixels are feature points, feature mean point, and CCL center, respectively. These figures show that even through the CCL object contains the forearm part, the hand direction can be extracted precisely. The next step is to exclude the forearm region from the CCL object.

(a) (b)

Fig- 3.12 The hand direction (a) without forearm (b) with forearm.

Obviously, if the object is farer from the camera, the object would have smaller size in the image. Therefore, the detected hand size would be influenced by different distances. The relation between the size of the hand in the image and the distance from object to camera as shown in Fig- 3.13, where x-axis is the width of the hand and y-axis is the corresponding depth value. Due to this characteristic, the hand size can be modeled as a linear function of the depth value shown as below:

$$W = -a \times Depth + b \tag{3.14}$$

where a,b are positive parameters. The hand region can be obtained in two steps. First, determine two points (x_d,y_d) and (x_p,y_p) on the hand direction line and satisfy the following conditions: the distance from (x_{FPMean}, y_{FPMean}) to point (x_d,y_d) is 0.45 times the hand width W, and the point is on the same side of the hand direction, while point (x_p,y_p) is $0.55W$ and on the opposite side of hand direction. Consequently , (x_d,y_d) and (x_p,y_p) can be calculated as below:

$$\begin{cases} x_d = x_{FPMean} + 0.45W \times cos\left(HD^\circ\right) \\ y_d = y_{FPMean} + 0.45W \times sin\left(HD^\circ\right) \end{cases} \tag{3.15}$$

$$\begin{cases} x_p = x_{FPMean} - 0.55W \times cos\left(HD^\circ\right) \\ y_p = y_{FPMean} - 0.55W \times sin\left(HD^\circ\right) \end{cases} \tag{3.16}$$

Therefore, the hand direction line can be determined by the two points. Second, determine a line passing through (x_p,y_p) with length W and is perpendicular to the hand direction. This line is near the wrist which can be used to separate the fingers and palm from the wrist and arm as Fig- 3.14. These two lines can determine a rectangle region that excludes the forearm region shown in Fig- 3.15, where the green rectangle is the CCL region and blue one is selected hand region. Therefore, the palm center, (x_{center},y_{center}), can be determined by calculating the mean point within the blue rectangle area.

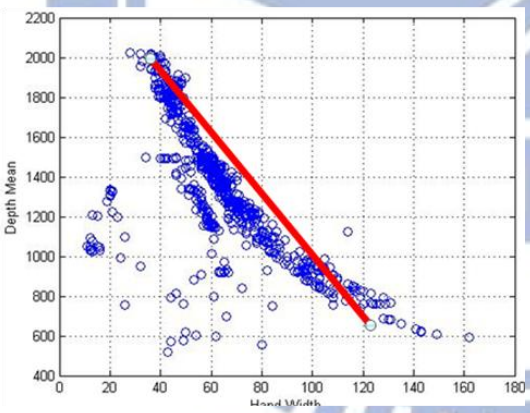


Fig- 3.13 The hand size under different depth value

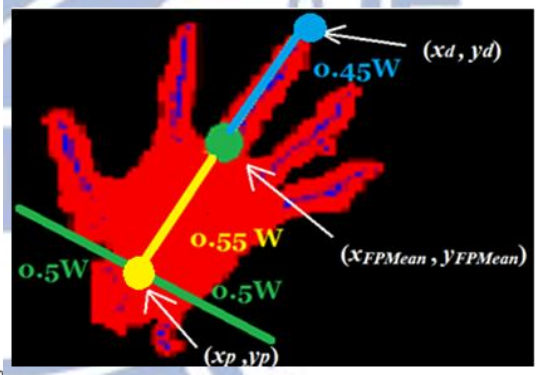
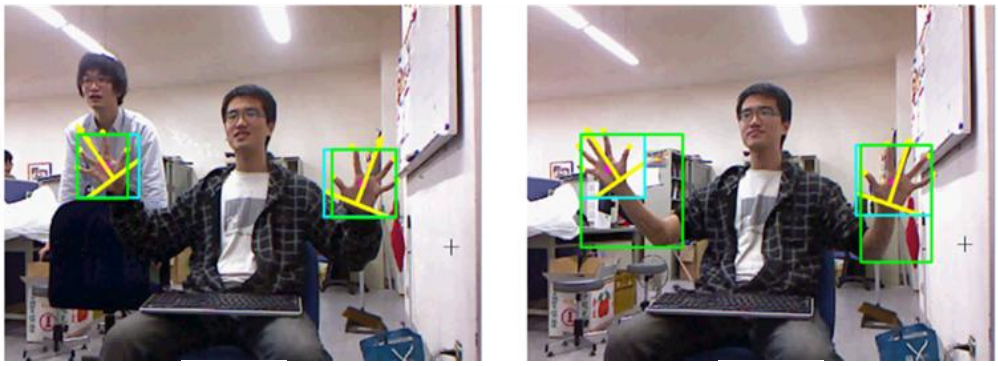


Fig- 3.14 The process separating the forearm part.

The process above usually takes a little time since the total number of distance-based feature points is usually small. The next step is to search the fingertips location.



(a) (b)

Fig- 3.15 The result finding hand region (a) without forearm part (b) with forearm.

3.2.2 Fingertip Positions

As shown in Section 3.1.2, there are many feature points in the fingertips region, and these points can be used to determine the fingertips position quickly and accurately. Fig- 3.16 (a) shows the detected hand region, where red pixels are the skin color and blue pixels are feature points, and the fingertips position can be found by the following steps.

Step1: Extract the feature points in Fig- 3.16 (a), shown as Fig- 3.16 (b), and then implement dilation operation to connect the discontinuous feature points, and the result shown as Fig- 3.16 (c).

Step2: Use CCL to label connected region and compute each CCL region. Only larger enough CCL area is considered as fingers, and the area is related to the distance from camera to the hand. Fig- 3.17 shows the average connected region of finger corresponding to different depth value. The other CCL region which is less than one deviation value would be taken as noise, and the Fig- 3.16 (c) shows the result.

Step3: Calculate the mean and standard deviation of each CCL object, and two candidate fingertip positions, $CT_{i,1}$ and $CT_{i,2}$, i=1~5, can be calculated, shown as Fig- 3.16(d). Calculate the distance between the candidate fingertip and wrist line, which is obtained by Chapter 3.2.1. The point which is far from the wrist line is the selected fingertip position, and the result shown in Fig- 3.16(e). And the fingertip angle is the connection from $CT_{i,1}$ and $CT_{i,2}$.

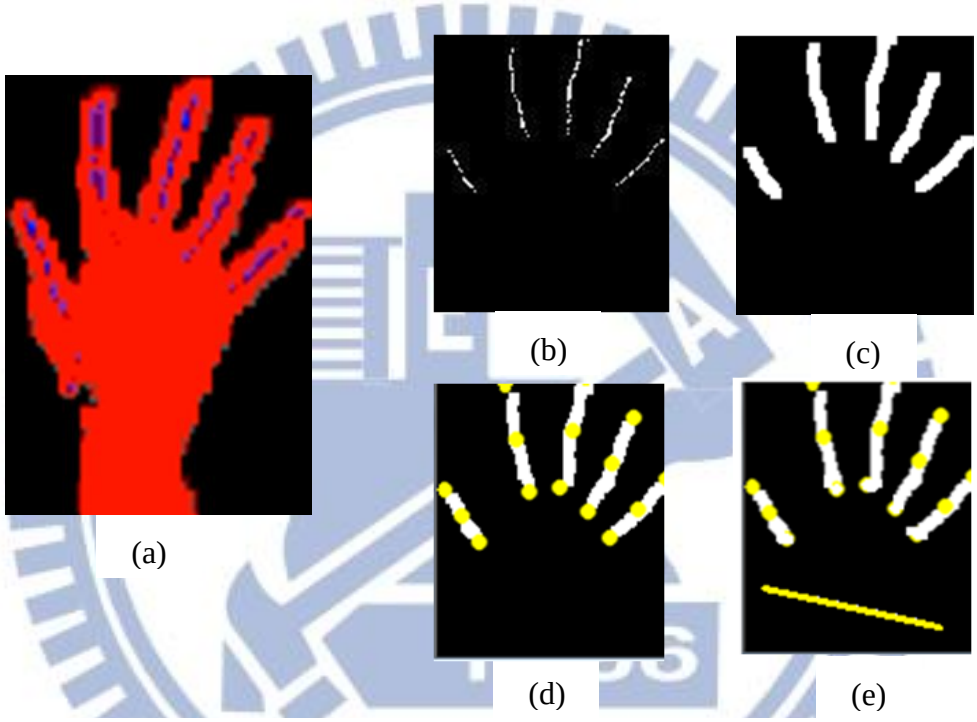


Fig- 3.16 The procedure of finding fingertips position (a) detected hand region (b) the feature points (c) implement dilation (d) mean and standard deviation corresponding to each CCL object (e) the selected fingertips.

With the proposed method, the hand feature extraction process will determined the all outstretched hand direction, fingertip positions and the palm center. Once these features are obtained, the recognition system can grasp the posture of the hand more easily, and the hand tracking process can be also performed more easily and precisely.

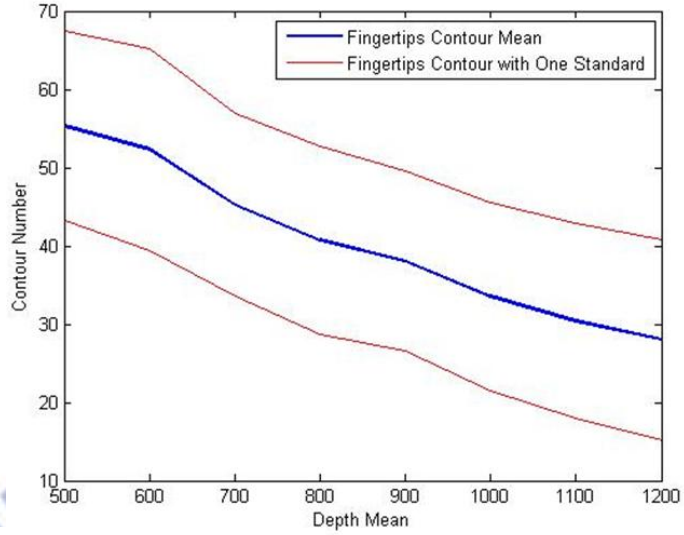


Fig- 3.17 The procedure of finding fingertips position (a) detected

In detail, the set of the features includes:

1. The position of palm center, (x_{center},y_{center}).
2. The displacement of palm center.
3. The hand direction, HD°.
4. Total detected fingertips number and its positions.(from 1 to 5).
5. Direction of each finger.
6. The depth mean of the hand.
7. The change of the mean depth.
8. The mean distance from (x_{center},y_{center}) to each fingertip.

From the above hand features, a hand recognition model can be built. Moreover, these features can be exploited not only by HGR system but also by hand tracking system.

3.3 Hand Gesture Recognition

After hand region detection, the system has to judge recognize different gestures based on the extracted features. To test the ability of variation tolerance, both sequential and non-sequential training method. This thesis adopts discriminative classification model, neuron network and hidden Markov model to implement the HGR systems. The features which are extracted in chapter 3.2 are sent into these HGR systems and compare their recognition rate and processing speed. The resolution in the experiments are 640×480 at 20 frames-per second. These approaches would be introduced below and their performances would be compared in Chapter 4. This experiment test 6 hand gestures to be recognized, including moving left, right, upward, downward, open and click.

3.3.1 Discriminative Model

The discriminative model directly assume the posterior probability of class C_k as a logistic sigmoid acting on a linear function of the input vectors:

$$\begin{cases} p\left(C_j \mid \boldsymbol{x}\right) \triangleq \dfrac{\exp\left(\boldsymbol{w}_j^T \boldsymbol{x}\right)}{1+\sum_{k=1}^{K-1}\exp\left(\boldsymbol{w}_k^T \boldsymbol{x}\right)}, j = 1 \sim K-1 \\ p\left(C_K \mid \boldsymbol{x}\right) \triangleq \dfrac{1}{1+\sum_{k=1}^{K-1}\exp\left(\boldsymbol{w}_k^T \boldsymbol{x}\right)} \end{cases} \tag{3.17}$$

where $\boldsymbol{x}$ is input vector and $\boldsymbol{w}$ is the weighting vector. Given the data set $\{\boldsymbol{x}_n, \boldsymbol{t}_n\}$, n=1~N, where $\boldsymbol{t}_n$=[$t_{n,1}$ $t_{n,2}$... $t_{n,K-1}$]^T and $t_{n,k}$=1 when $\boldsymbol{x}_n$ belongs to class k while the others are 0, the likelihood function can be written as below:

$$p\left(T \mid \boldsymbol{w}_1, ..., \boldsymbol{w}_{K-1}\right) = \prod_{n=1}^{N}\prod_{k=1}^{K-1} y_{n,k}^{t_{n,k}} \tag{3.18}$$

where $y_{n,k}$=p(C_k| $\boldsymbol{x}$). As usual, define the error function by taking the negative

logarithm of (3.18), which gives the cross-entropy function in the following form:

$$E(\boldsymbol{w}_1,...,\boldsymbol{w}_{K-1}) = -\sum_{n=1}^{N}\sum_{k=1}^{K-1} t_{n,k} \ln y_{n,k} \tag{3.19}$$

The error function can be minimized by the Newton-Raphson iterative optimization method, which uses a local quadratic approximation to the log likelihood function. The Newton-Raphson update for minimizing error function takes the following form:

$$\boldsymbol{W}^{new} = \boldsymbol{W}^{old} - H^{-1}\nabla E(\boldsymbol{W}) \tag{3.20}$$

where $\boldsymbol{W}=[\boldsymbol{w}_1\ \boldsymbol{w}_2...\ \boldsymbol{w}_{K-1}]^T$ is the whole weighting vector and H is the Hessian matrix whose elements contain the second derivatives of $E(\boldsymbol{W})$ with respect to each $\boldsymbol{w}_j$. The each component of the gradient is given as below:

$$\nabla E(\boldsymbol{W}) = \left[\nabla_{\boldsymbol{w}_1} E(\boldsymbol{W}) \nabla_{\boldsymbol{w}_2} E(\boldsymbol{W}) \quad \cdots \quad \nabla_{\boldsymbol{w}_{K-1}} E(\boldsymbol{W})\right]^T \tag{3.21}$$

where

$$\nabla_{\boldsymbol{w}_j} E(\boldsymbol{W}) = \sum_{n=1}^{N}\left(y_{n,j} - t_{n,j}\right)\boldsymbol{x}_n \tag{3.22}$$

The each component of Hessian matrix is computed using the following:

$$-H = \begin{bmatrix} \nabla_{\boldsymbol{w}_1}\nabla_{\boldsymbol{w}_1} E(\boldsymbol{W}) & \cdots & \nabla_{\boldsymbol{w}_1}\nabla_{\boldsymbol{w}_{K-1}} E(\boldsymbol{W}) \\ \vdots & \ddots & \vdots \\ \nabla_{\boldsymbol{w}_{K-1}}\nabla_{\boldsymbol{w}_1} E(\boldsymbol{W}) & \cdots & \nabla_{\boldsymbol{w}_{K-1}}\nabla_{\boldsymbol{w}_{K-1}} E(\boldsymbol{W}) \end{bmatrix} \tag{3.23}$$

where

$$\nabla_{\boldsymbol{w}_k}\nabla_{\boldsymbol{w}_j} E(\boldsymbol{W}) = -\sum_{n=1}^{N} y_{n,k}\left(\boldsymbol{I}_{k,j} - y_{n,j}\right)\boldsymbol{x}_n \boldsymbol{x}_n^T \tag{3.24}$$

The Hessian matrix for the multiclass classification problem is positive definite and the error function can converge to a unique minimum. In this thesis, K is chosen to be 6 to recognize the defined gestures.

3.3.2 Neural-Network-Based Recognition

The structure of neural network is shown in Fig- 3.18, which contains one input

layer with O neurons, one hidden layer with P neurons, and one output layer with Q neuron. After feature extraction, these features are sent into the neural network as inputs. The O neurons of the input layer are represented by $\boldsymbol{x}$. The i-th input neuron is connected to the j-th neuron, j=1,2,…,P, of the hidden layer with weighting $\boldsymbol{w}_j^{(1)}$. Besides, the j-th neuron of the hidden layer is also with an extra bias $w_{j0}^{(1)}$. Finally, the j-th neuron of the hidden layer is connected to k-th output neuron with weighting $\boldsymbol{w}_k^{(2)}$, k=1,2,…,Q, and a bias $w_{k0}^{(1)}$ is added to the output neuron.

Let the activation function of the hidden layer be the hyperbolic tangent-sigmoid transfer function and the output vector of hidden layer is expressed as

$$\boldsymbol{z} = [z_1 \quad z_2 \quad \cdots \quad z_P]^T \tag{3.25}$$

where

$$z_j = \frac{\exp\left(-\boldsymbol{w}_j^{(1)T}\boldsymbol{x}\right) - \exp\left(-\boldsymbol{w}_j^{(1)T}\boldsymbol{x}\right)}{\exp\left(-\boldsymbol{w}_j^{(1)T}\boldsymbol{x}\right) + \exp\left(-\boldsymbol{w}_j^{(1)T}\boldsymbol{x}\right)}, \quad j = 1, 2, ..., P \tag{3.26}$$

Let the activation function of the output layer be the softmax function and the output is expressed as

$$\boldsymbol{y} = [y_1 \quad y_2 \quad \cdots \quad y_Q]^T \tag{3.27}$$

where

$$\begin{cases} y_k = \dfrac{\exp\left(\boldsymbol{w}_k^{(2)T}\boldsymbol{z}\right)}{1 + \sum_{k=1}^{Q-1} \exp\left(\boldsymbol{w}_k^{(2)T}\boldsymbol{z}\right)}, & k = 1, 2, ..., Q-1 \\ y_Q = 1 - \sum_{k=1}^{Q-1} y_k \end{cases} \tag{3.28}$$

The above operations are shown in Fig- 3.18. The weights of neural network would be adjusted through the process of back-propagation learning algorithm. After learning, the HGR system can be recognized according to the output value of neural network.

The final recognition result is made using the following decision strategy:

$$HG = \arg\max_{q} y_q, q = 1, 2, ..., Q \tag{3.29}$$

In this experiment, the number of neuron in each layer are 8, 10, 6 corresponding to input, hidden, and output layer.

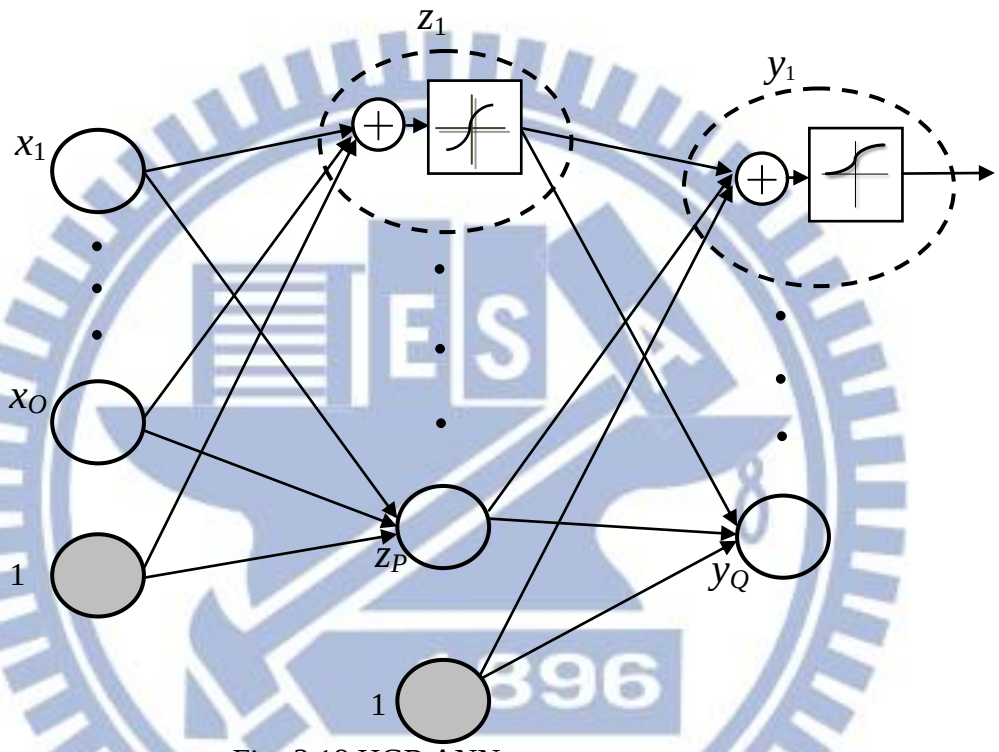


Fig- 3.18 HGR ANN structure

3.3.3 HMM-Based Recognition

As a hand gesture is basically an action composed of a sequence of hand postures that are connected by continuous motions, thus describing the hand gestures in terms of a sequential input is suitable in HGR systems. HMM have been applied to HGR tasks with success. The classification of the input sequence proceeds by computing the sequence's similarity to each of the gesture class models. The joint distribution is given by:

$$p(\boldsymbol{x}_1,...,\boldsymbol{x}_N,\boldsymbol{z}_1,...,\boldsymbol{z}_N) = p(\boldsymbol{z}_1)\prod_{n=2}^{N} p(\boldsymbol{z}_n \mid \boldsymbol{z}_{n-1})\prod_{n=1}^{N} p(x_n \mid z_n) \tag{3.30}$$

where $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1})$, $p(\boldsymbol{x}_n|\boldsymbol{z}_n)$ are called transition and emission probability, respectively. $\boldsymbol{z}_n$ is the latent variable, or called hidden state, corresponding to the input data $\boldsymbol{x}_n$. An iterative expectation-maximization (EM) algorithm procedure is used to maximize the likelihood function and obtain optimal parameters. To speed up the calculation, the Viterbi algorithm is used that finding the most probable sequence of latent variables for a given observations $X = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_N\}$. The posterior probability of time t is proportional to the jointly probability:

$$p(z_1,...,z_t \mid x_1,...,x_t) \propto p(z_1,...,z_t,x_1,...,x_t) \tag{3.31}$$

Let $z_1,...,z_t \equiv z_{1:t}$ and $x_1,...,x_t \equiv x_{1:t}$, and the message passed in the max-sum algorithm are given by:

$$\begin{aligned}\mu(z_t) &\triangleq \max_{z_1,...,z_{t-1}} p(z_{1:t},x_{1:t}) \\ &= \max_{z_{t-1}}[p(x_t \mid z_t)p(z_t \mid z_{t-1}) \max_{z_1,...,z_{t-2}} p(z_{1:t-1},x_{1:t-1})] \\ &= \max_{z_{t-1}}[p(x_t \mid z_t)p(z_t \mid z_{t-1})\mu(z_{t-1})]\end{aligned} \tag{3.32}$$

The Viterbi algorithm consider explicitly all of the exponentially many paths, evaluate the probability for each and then select the path having the highest probability. In this experiment, the latent variable z_n has 6 states and N is chosen to be 10.

This experiment test 6 hand gestures to be recognized, including moving left, right, upward, downward, open and click, and Fig- 3-19 shows the defined gestures. The hand gestures are defined in natural way rather than in particular ordered hand pose. The palm center difference between the current and previous frames can be used to recognize the these gestures. The above three algorithms are used to recognize these six hand gestures.

Fig- 3.19 The defined hand gestures.

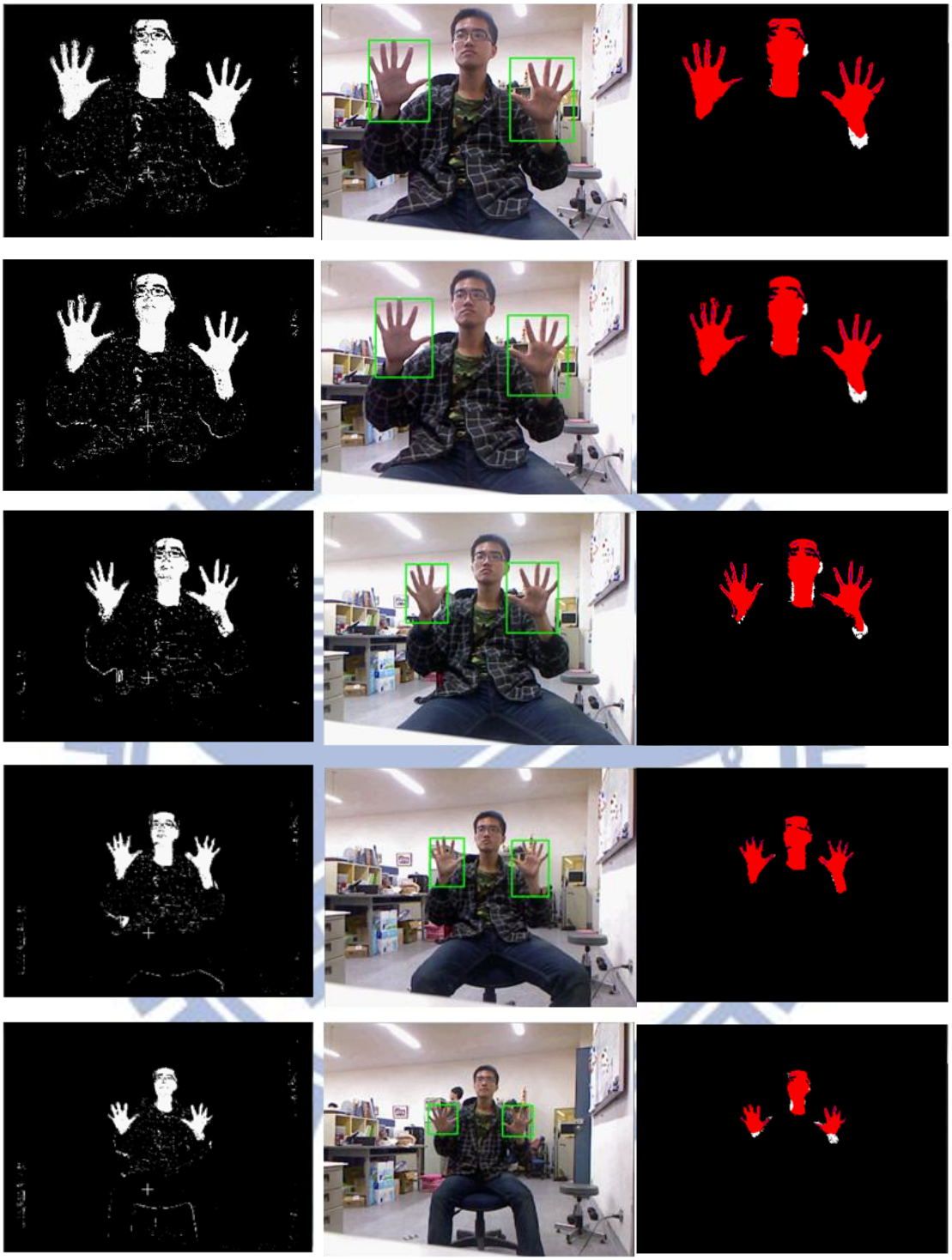
Chapter 4

Experimental Results

In the previous chapters, the three main steps of the proposed HGR system are introduced. In this chapter, the experiment results of each step will be shown in detail and the results of the proposed algorithm will be obtained by MATLAB R2010b and OpenCV 2.2.

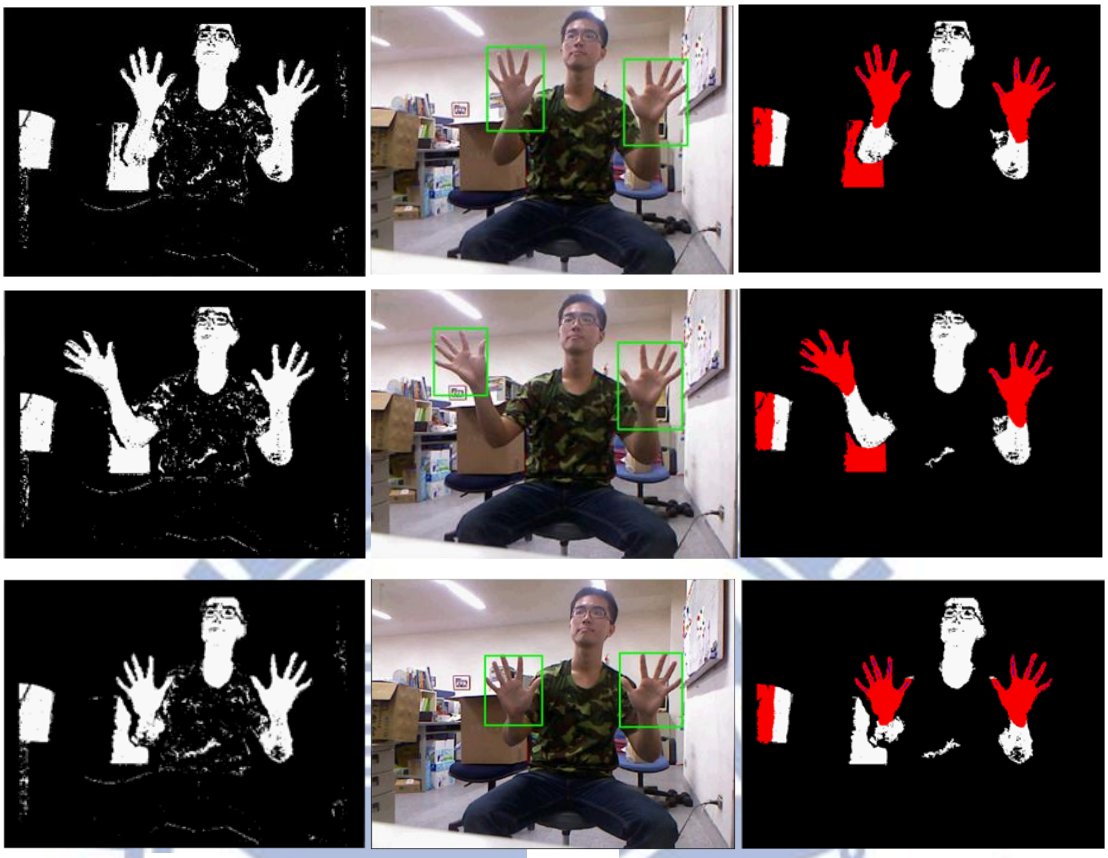
4.1 Hand Region Detection

To examine the reliability of hand region detection, the system is tested in many different situations, including different distances, the skin color background, overlapping by other skin object and more than one human, and the results are shown from Fig- 4.1 to Fig- 4.4, respectively. The left column contains the skin color images, the middle one shows the ROI images, and the right one represents the skin color regions with large enough area, where the white pixel indicates skin color and red represents the depth searching region. Note that the green rectangles in the middle are the selected ROIs. Fig- 4.1 shows even though human keeps moving away from the camera, the system would not fail to extract hand region. The distance between the human and camera is from 0.5m to 2m. In Fig-4.2, there are some skin color objects in the images, and the system also could detect the hand regions.



(b) (c)

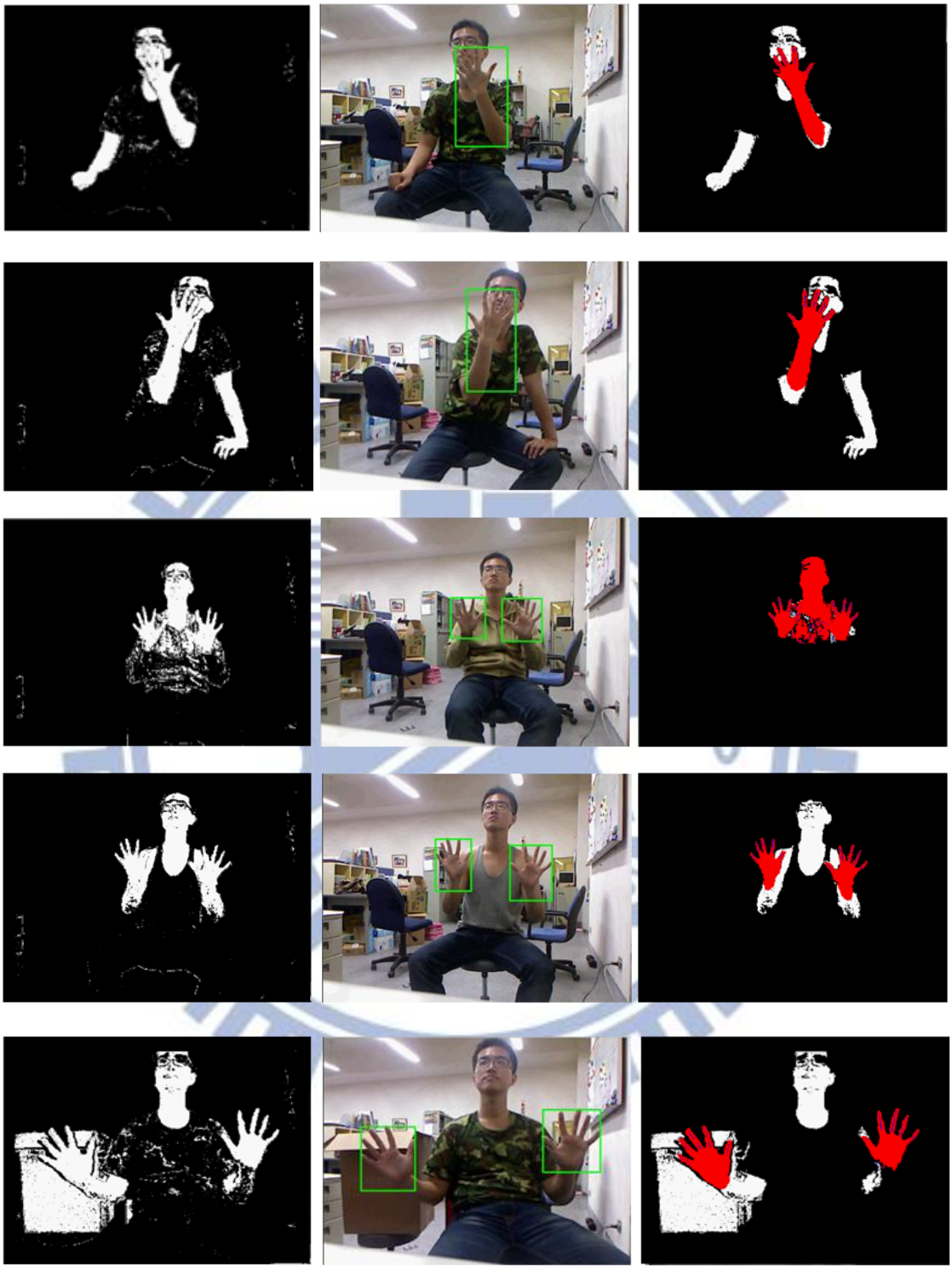
Fig- 4.1 Results of hand region detection in different distances. (a) The original skin color region (b) The ROI images (c) skin color regions with large enough area. Note that the green rectangles in (b) are the selected ROIs.



(a) (b) (c)

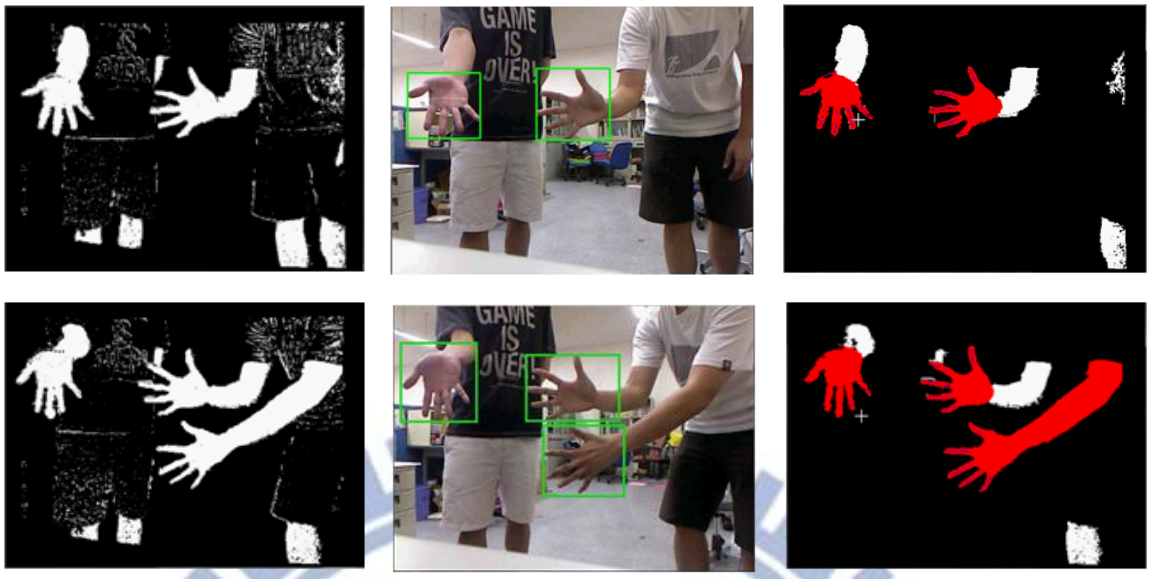
Fig- 4.2 Results of complex skin color background.

The situations in Fig- 4.3 and Fig- 4.4 are more complex. In Fig- 4.3, there are overlapping between the hand and the other skin-color object, like face and shirt. The results show that system still could extract hand regions and even when suffering from serious overlapping problem. In Fig-4.4, there are more than one human standing in front of the camera, and the system could filter out these regions to reduce the number of ROI and still success to extract the hand regions. The programming tools are Visual Studio 2010 and OpenNi for controlling the Kinect, and the resolution of the video is 640×480 pixels. The average processing speed is 0.2ms which is fast enough to be used in a real-time HCI system.



(a) (b) (c)

Fig- 4.3 Results of hand region detection in the condition of overlapping between the hand and other skin color objects.



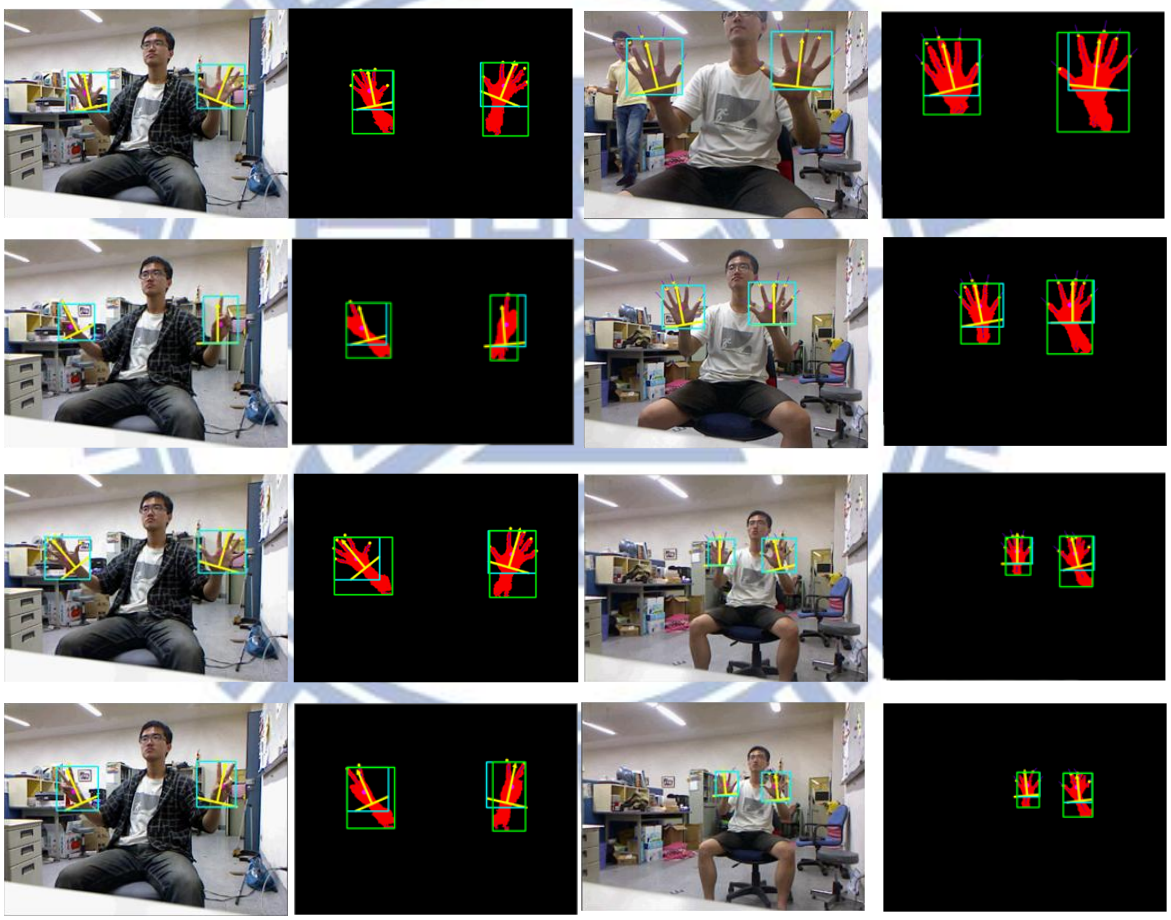
(a) (b) (c)

Fig- 4.4 Results of hand region detection in the condition of multiple users.

4.2 Feature Extraction

In this section, the experimental results are presented in three parts. The first part focuses on the forearm cropping. The second part focuses on the performance of different hand orientation. The third part shows the results of finding fingertip positions.

Experiment 1:



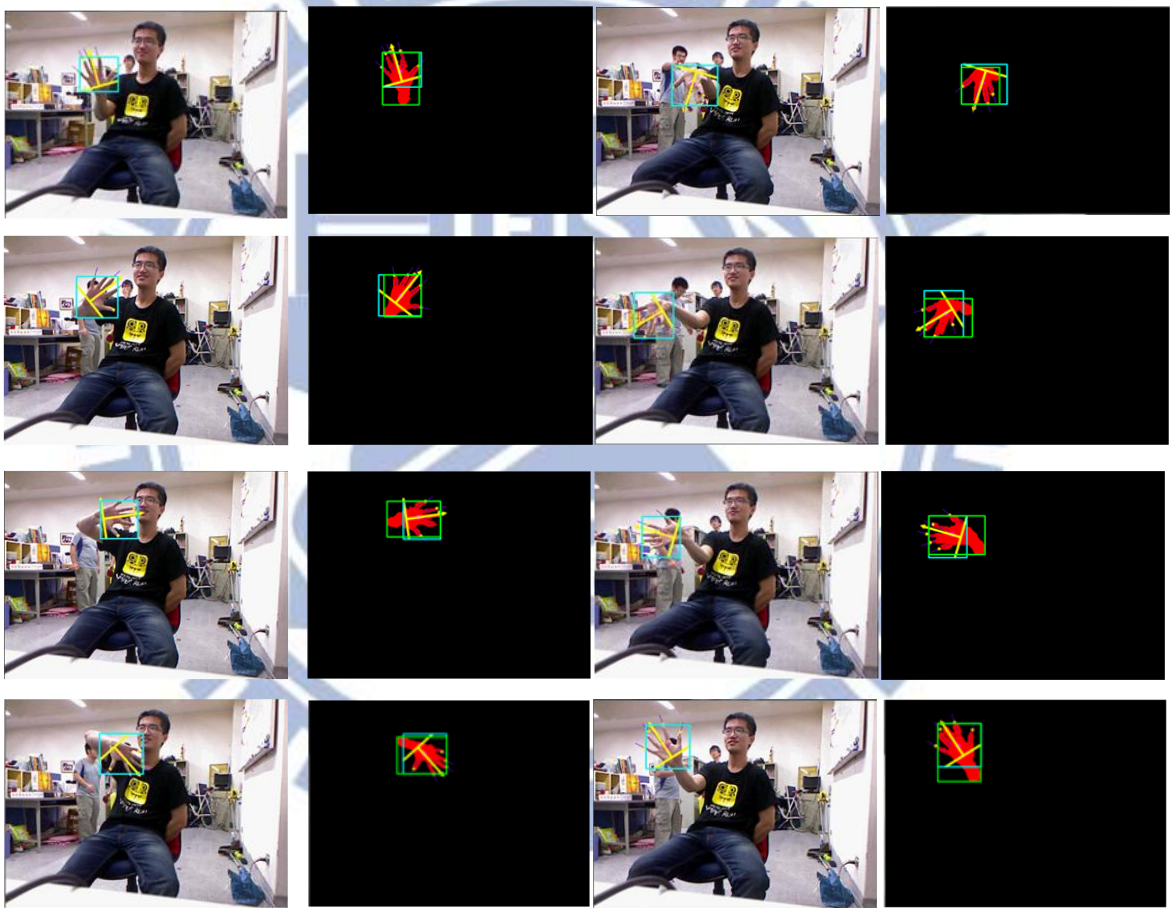
(a) (b)

Fig- 4.5 Results of forearm cropped in condition of (a) different postures (b) different distances.

To evaluate the ability of the proposed method to crop the forearm region, and Fig- 4.5 shows the results, where the green rectangles are boundary of detected CCL

objects and blue ones are hand regions. The result of Fig- 4.5 (a) shows that even though the hands are not open, the forearm regions are still be cropped. In order to examine the function of normalization, the experiment test different distances between the human and camera including 0.5m, 1.0m, 1.5m and 2m, and Fig- 4.5 (b) shows the result,. Obviously, the influence of distance is highly reduced.

Experiment 2:

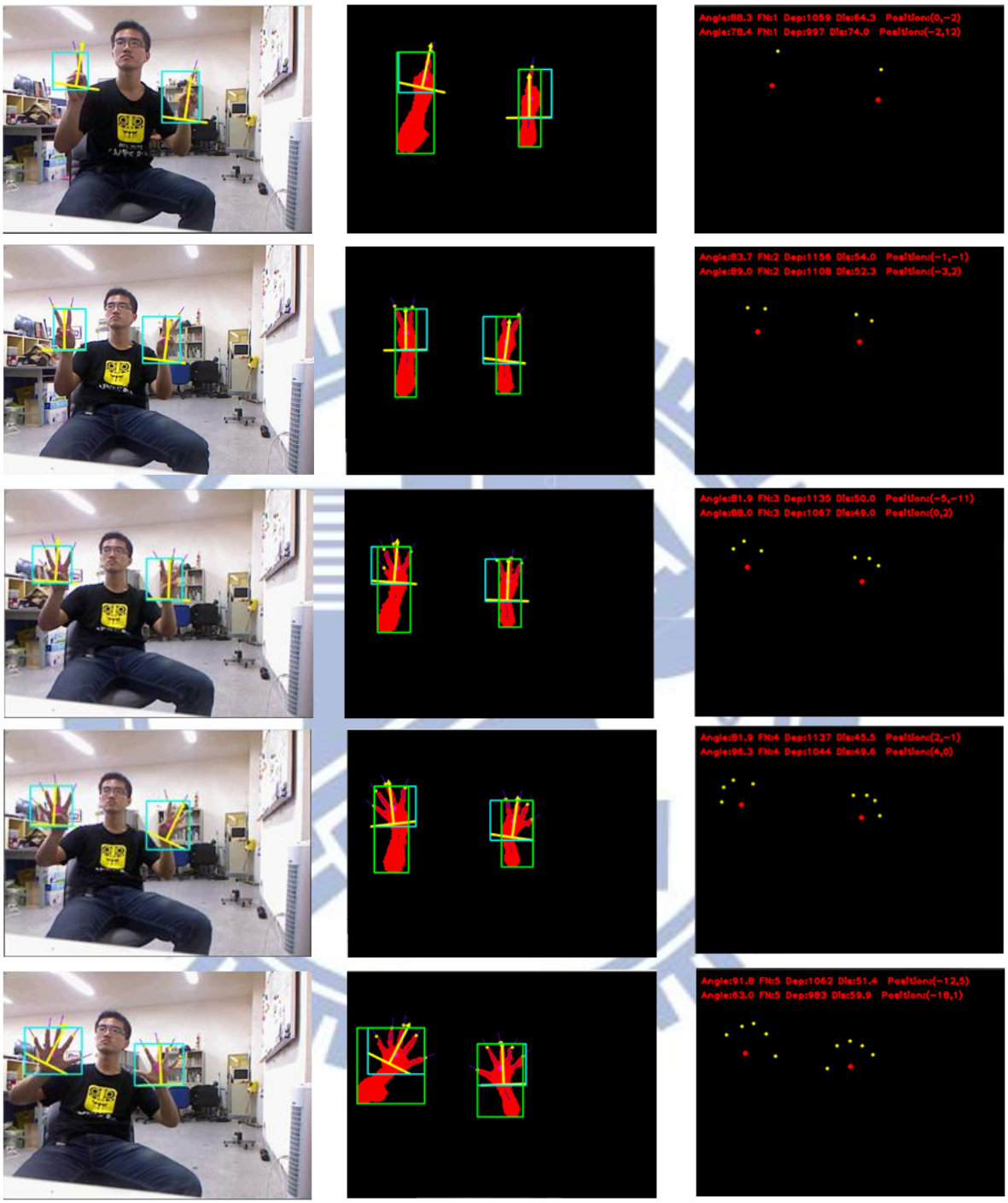


(a) (b)

Fig- 4.6 Results of different hand direction.

This experiment shows the hand direction can be detected with fast processing speed and accurate rate. These information can be applied in a sequential HGR process.

Experiment 3:



(a) (b) (c)

Fig- 4.7 Results of finding fingertip positions.

The fingertip positions are shown in Fig- 4.7 (c), where the red and yellow

points denote palm center and fingertips, respectively. This experiment shows the fingertip positions can be extracted fast and precisely.

4.3 Hand Gesture Recognition

In this section, the hand recognition system would be tested in algorithms to examine the performance and reliability. This thesis adopts discriminative classification model, neuron network and hidden Markov model to implement the HGR systems. This experiment test 6 hand gestures to be recognized, including moving left, right, upward, downward, open and click. The performances of these methods in different algorithm are shown in Table 4.1, which include 4483 training dataset.

Table 4.1 Recognition Results of Different Classifier

	Correct rate(%)	Process Time(s)
Discriminative	85.6	0.09
NN-based Approach	89.83	0.24
HMM-based Approach	91.06	0.31

This experiment shows that the hand feature extraction method can be used in different recognition method and posses good performance, and the result shows in Fig- 4.9. Furthermore, a hand gesture composed of a sequence of hand postures that are connected by continuous motions can be recognized using the extracted features.

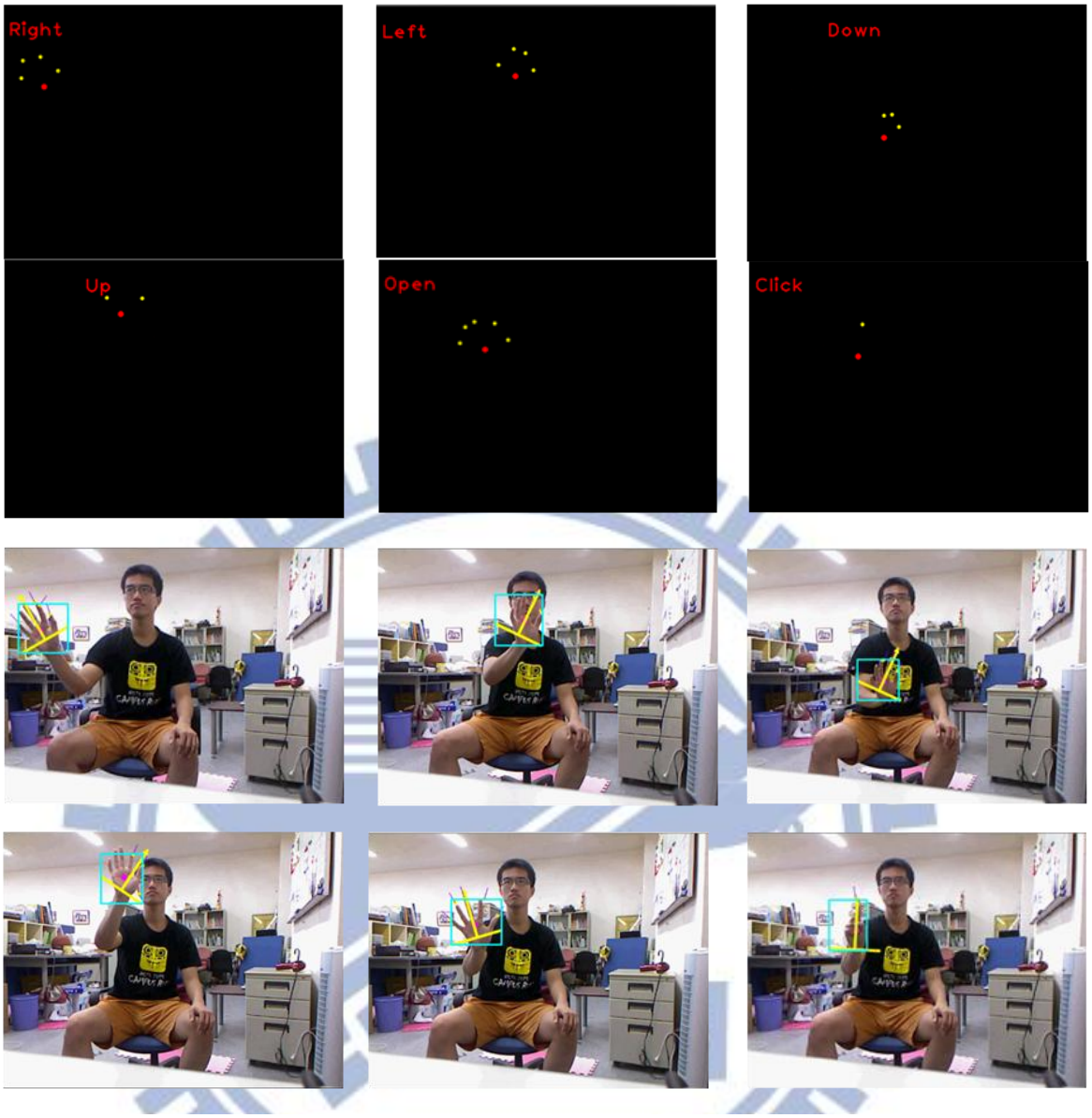


Fig- 4.8 Results of HGR systems.

It is obvious that the accuracy rate of HMM-based HGR system is higher than that of the other two HGR systems. However, the computational cost of HMM-based is lower than other two HGR systems and the average executing time of Set-I is lower than 0.1sec. This experiment shows that the proposed hand feature extraction method is useful and meaningful for the HGR system, no matter what the classifier is. Fig- 4.9 shows the decision boundary of the four moving direction using NN-based algorithm.

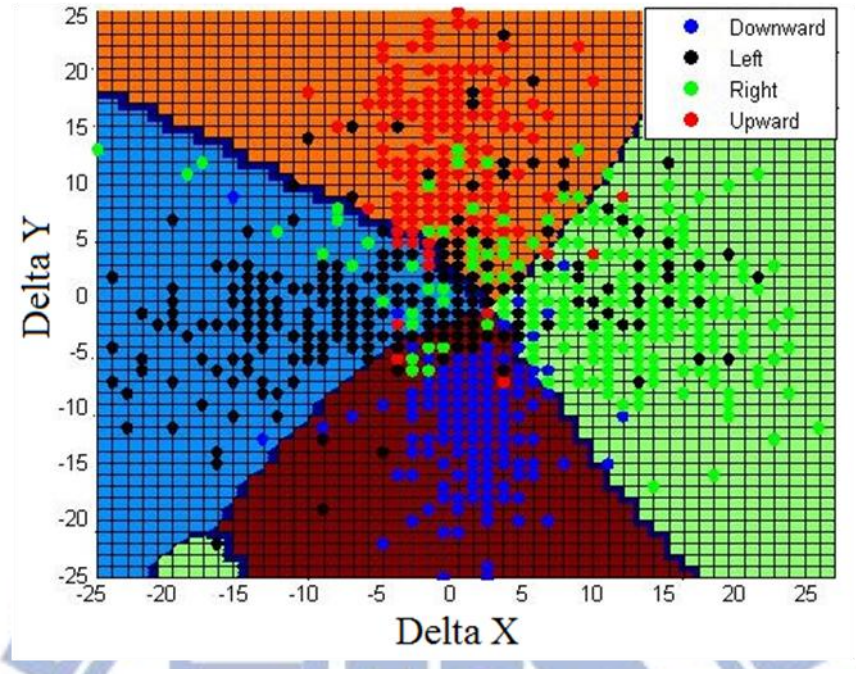


Fig- 4.9 Decision boundary of four moving direction

The next experiment test a real application on HCI, which use the HGR system to control the direction and position of the mouse, and Fig- 4.10 shows this implementation. For more defined gestures, the mouse control can process more complicated movement.

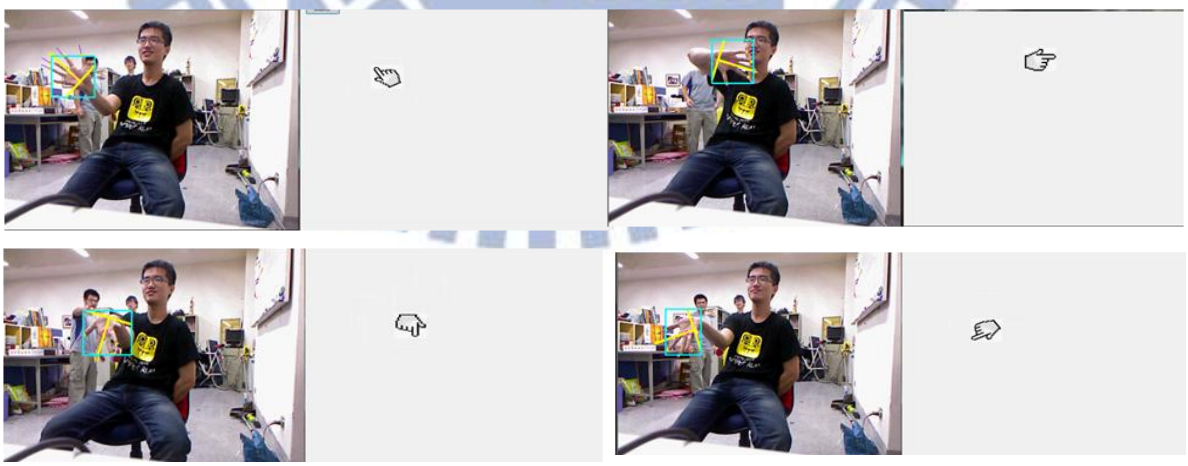


Fig- 4.10 Application of controlling mouse.

Chapter 5

Conclusions and Future Works

This thesis proposes a fast and robust hand feature extraction method based on depth and RGB information generated by Kinect that can be used in real-time HGR or hand tracking. The system is divided into three parts, including ROI selection, feature extraction and HGR. First, the skin color detection and connected component labeling (CCL) are applied to select the potential ROIs. Through distance transformation, the system could extract feature points that can be used to find hand features, which includes direction, fingertip positions, and palm center. Finally, these features are sent into several HGR systems. From the experimental results, there are some conclusions listed as below:

- The ROI selection could detect hand region with accuracy rate higher than 90% and the average executing time about 0.2sec/frame. Besides, with the help of depth image, the system could also detect hands correctly even suffering from the overlapping between the target hand and other skin color object.

- The feature points extracted by this thesis can be used to find useful hand features for HGR. With the depth information, the forearm could be cropped and these feature points could generate the fingertip positions and hand direction. These hand features are useful and meaningful for the usage of human-computer interaction(HCI).

- The extracted features are sent into the HGR system, which is implemented by different algorithms. The experimental results show that the proposed hand feature extraction method is useful and meaningful for HGR system and can work in

real-time and possess high recognition rate.

The proposed feature extraction method has been demonstrated to be successful in HGR system, which is an important function in the field of HCI. With this HGR system, the HCI could be more accurate and natural. There are two primary future works to further investigate which are presented as below:

- The ROI selection could not only detect outstretched hands but hands with curved fingers. Furthermore, with the depth distribution information of a CCL object, the 3-D direction can be estimated to build up a hand model.
- Two or more users can appear in the screen to allow a more complicated hand gestures. Therefore, the ROI selection and HGR system should have the ability to discriminate these hand gestures when faces occlusion problem. Furthermore, the more robust HGR system should be designed to handle the hand gestures with ambiguous movement.

Appendix. A:Forward-Backward Algorithm

Given observation sequence $X=\{\boldsymbol{x}_1,...,\boldsymbol{x}_N\}$ and the joint probability of HMM as below:

$$p(X,Z)=p(\boldsymbol{z}_1)\prod_{n=2}^{N}p(\boldsymbol{z}_n\mid\boldsymbol{z}_{n-1})\prod_{n=1}^{N}p(x_n\mid z_n) \tag{A.1}$$

where all the model parameters $p(\boldsymbol{z}_1)$, $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1})$, and $p(\boldsymbol{x}_n|\boldsymbol{z}_n)$ are known and $Z=\{\boldsymbol{z}_1,...,\boldsymbol{z}_N\}$. Assume $\boldsymbol{z}_n$ is M-state discrete random variable, where $n=1,...,N$. The posterior probability of latent variable can be decoupled as:

$$\begin{aligned}p(\boldsymbol{z}_n\mid X)&=\frac{p(\boldsymbol{x}_1,...,\boldsymbol{x}_n,\boldsymbol{z}_n)\,p(\boldsymbol{x}_{n+1},...,\boldsymbol{x}_N\mid\boldsymbol{x}_1,...,\boldsymbol{x}_n,\boldsymbol{z}_n)}{p(X)}\\&=\frac{p(\boldsymbol{x}_1,...,\boldsymbol{x}_n,\boldsymbol{z}_n)\,p(\boldsymbol{x}_{n+1},...,\boldsymbol{x}_N\mid\boldsymbol{z}_n)}{p(X)}\triangleq\frac{\alpha(z_n)\beta(z_n)}{p(X)}\end{aligned} \tag{A.2}$$

since $\{\boldsymbol{x}_{n+1},...,\boldsymbol{x}_N\}$ is conditional independence with $\{\boldsymbol{x}_1,...,\boldsymbol{x}_n\}$ given $\boldsymbol{z}_n$ and where we have defined:

$$\begin{aligned}\alpha(z_n)&=p(\boldsymbol{x}_1,...,\boldsymbol{x}_n,\boldsymbol{z}_n)\\\beta(z_n)&=p(\boldsymbol{x}_{n+1},...,\boldsymbol{x}_N\mid\boldsymbol{z}_n)\end{aligned}$$

where $\alpha(\boldsymbol{z}_n)$ is called forward message that accumulated from the previous n observations and $\beta(\boldsymbol{z}_n)$ is backward message of all future data from time $n+1$ to N. The $\alpha(\boldsymbol{z}_n)$ can be expressed in terms of $\alpha(\boldsymbol{z}_{n-1})$ as follows:

$$\begin{aligned}\alpha(z_n)&=p(\boldsymbol{x}_1,...,\boldsymbol{x}_n,\boldsymbol{z}_n)\\&=\sum_{\boldsymbol{z}_{n-1}=1}^{M}p(\boldsymbol{x}_1,...,\boldsymbol{x}_n,\boldsymbol{z}_n,\boldsymbol{z}_{n-1})\\&=\sum_{\boldsymbol{z}_{n-1}=1}^{M}p(\boldsymbol{x}_n\mid\boldsymbol{z}_n,\boldsymbol{x}_1,...,\boldsymbol{x}_{n-1},\boldsymbol{z}_{n-1})p(\boldsymbol{z}_n\mid\boldsymbol{z}_{n-1},\boldsymbol{x}_1,...,\boldsymbol{x}_{n-1})\,p(\boldsymbol{z}_{n-1},\boldsymbol{x}_1,...,\boldsymbol{x}_{n-1})\end{aligned}$$

since $\boldsymbol{x}_n$ is conditional independence with $\{\boldsymbol{x}_1,...,\boldsymbol{x}_{n-1},\boldsymbol{z}_{n-1}\}$ given $\boldsymbol{z}_n$ and $\boldsymbol{z}_n$ is conditional independence with $\{\boldsymbol{x}_1,...,\boldsymbol{x}_{n-1}\}$ given $\boldsymbol{z}_{n-1}$, $\alpha(\boldsymbol{z}_n)$ can be simplified as:

$$\alpha(z_n) = \sum_{z_{n-1}=1}^{M} p(\boldsymbol{x}_n \mid \boldsymbol{z}_n) p(\boldsymbol{z}_n \mid \boldsymbol{z}_{n-1}) p(\boldsymbol{z}_{n-1}, \boldsymbol{x}_1, ..., \boldsymbol{x}_{n-1})$$

$$= p(\boldsymbol{x}_n \mid \boldsymbol{z}_n) \sum_{z_{n-1}=1}^{M} p(\boldsymbol{z}_n \mid \boldsymbol{z}_{n-1}) \alpha(\boldsymbol{z}_{n-1}) \quad \text{(A.3)}$$

where $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1})$ and $p(\boldsymbol{x}_n|\boldsymbol{z}_n)$ are known parameters. To start this recursion, the initial forward message is given by

$$\alpha(z_1) \triangleq p(\boldsymbol{x}_1, \boldsymbol{z}_1) = p(\boldsymbol{z}_1) p(\boldsymbol{x}_1 \mid \boldsymbol{z}_1) \quad \text{(A.4)}$$

Therefore, the overall cost of evaluating these quantities for n=1,...,N is of $O(NM^2)$.Similarly, the backward message $\beta(\boldsymbol{z}_n)$ can find a recursion relation as:

$$\beta(z_n) = p(\boldsymbol{x}_{n+1}, ..., \boldsymbol{x}_N \mid \boldsymbol{z}_n)$$

$$= \sum_{z_{n+1}=1}^{M} p(\boldsymbol{x}_{n+1}, ..., \boldsymbol{x}_N, \boldsymbol{z}_{n+1} \mid \boldsymbol{z}_n)$$

$$= \sum_{z_{n+1}=1}^{M} p(\boldsymbol{x}_{n+2}, ..., \boldsymbol{x}_N \mid \boldsymbol{z}_{n+1}, \boldsymbol{z}_n, \boldsymbol{x}_{n+1},) p(\boldsymbol{x}_{n+1} \mid \boldsymbol{z}_{n+1}, \boldsymbol{z}_n) p(\boldsymbol{z}_{n+1} \mid \boldsymbol{z}_n)$$

because $\{ \boldsymbol{x}_{n+2}, ..., \boldsymbol{x}_N \}$is conditional independence with $\{ \boldsymbol{x}_{n+1}, \boldsymbol{z}_n\}$ given $\boldsymbol{z}_{n+1}$, this would become:

$$\beta(z_n) = \sum_{z_{n+1}=1}^{M} p(\boldsymbol{x}_{n+2}, ..., \boldsymbol{x}_N \mid \boldsymbol{z}_{n+1}) p(\boldsymbol{x}_{n+1} \mid \boldsymbol{z}_{n+1}) p(\boldsymbol{z}_{n+1} \mid \boldsymbol{z}_n)$$

$$= \sum_{z_{n+1}=1}^{M} \beta(\boldsymbol{z}_{n+1}) p(\boldsymbol{x}_{n+1} \mid \boldsymbol{z}_{n+1}) p(\boldsymbol{z}_{n+1} \mid \boldsymbol{z}_n) \quad \text{(A.5)}$$

also, a backward message $\beta(\boldsymbol{z}_n)$ can be evaluated in terms of $\beta(\boldsymbol{z}_{n+1})$, and for n=N-1, the backward message is:

$$\beta(\boldsymbol{z}_{N-1}) \triangleq p(\boldsymbol{x}_N \mid \boldsymbol{z}_{N-1}) = \sum_{z_N=1}^{M} p(\boldsymbol{x}_N, \boldsymbol{z}_N \mid \boldsymbol{z}_{N-1})$$

from (A.5), this term is:

$$\beta(\boldsymbol{z}_{N-1}) = \sum_{z_N=1}^{M} \beta(\boldsymbol{z}_N) p(\boldsymbol{x}_N \mid \boldsymbol{z}_N) p(\boldsymbol{z}_N \mid \boldsymbol{z}_{N-1})$$

$$= \sum_{z_N=1}^{M} \beta(\boldsymbol{z}_N) p(\boldsymbol{x}_N, \boldsymbol{z}_N \mid \boldsymbol{z}_{N-1}) = \sum_{z_N=1}^{M} \beta(\boldsymbol{z}_N) \beta(\boldsymbol{z}_{N-1})$$

so

$$\beta(\boldsymbol{z}_N) = 1 \quad \text{(A.6)}$$

Reference

[1] S. Sclaroff and V. Athitsos, "Skin color-based video segmentation under time-varying illumination, " *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol.26, pp.862-877, July 2004.

[2] R. C. González and R. E. Woods, *Digital Image Processing, 3rd Edition*: Pearson/Prentice Hall, 2008.

[3] M. de La Gorce, D. J. Fleet and N. Paragios, "Model-Based 3D Hand Pose Estimation from Monocular Video," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol.33, pp.1793-1805, September 2011.

[4] C.W. Ng and S. Ranganath, "Gesture recognition via pose classification," in *Pattern Recognition, 2000 IEEE 15th International Conference on*, vol. 3, pp. 699–704, 2000.

[5] R. H. Liang and M. Ouhyoung, "A real-time continuous gesture recognition system for sign language," in *Automatic Face and Gesture Recognition,1998 Third IEEE International Conference on*, pp. 558–565, 1998.

[6] A. Morales, "Towards contactless palmprint authentication," in *IET Computer Vision*, vol. 5, pp. 407–416, 2011.

[7] D.Nicolas and E. M. Petriu, "Hand gesture recognition using haar-like features and a stochastic context-free grammar," *Instrumentation and Measurement, IEEE Transactions on*, vol.57, pp.1562-1571, 2008.

[8] L. Gupta and M. Suwei, "Gesture-based interaction and communication: automated classification of hand gesture contours," *Systems, IEEE Transactions on Man, and Cybernetics, Part C: Applications and Reviews*,

vol:31, pp.114-120, 2001.

[9] S. M. Dominguez, T. Keaton and A. H .Sayed, " A robust finger tracking method for multimodal wearable computer interfacing," *Multimedia, IEEE Transactions on*, vol.8, pp.956-972, October 2006.

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

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

[12] J. C. Terrillon and S. Akamatsu, "Comparative performance of different chrominance spaces for color segmentation and detection of human faces in complex scene images," *Proc. Vision Interface*, pp. 180–187, 1999.

[13] M. H. Yang, N. Ahuja, "Gaussian mixture model for human skin color and its application in image and video databases", *Proceedings of SPIE: Conference on Storage and Retrieval for Image and Video Databases*, vol. 3656, pp. 458 – 466,1999.

[14] S. M. Kay, *Fundamentals of Statistical Signal Processing Volume II Detection Theory*: Pearson/Prentice Hall, 1996.

[15] H. Youngmo,"A Low-Cost Visual Motion Data Glove as an Input Device to Interpret Human Hand Gestures," *Consumer Electronics, IEEE Transactions on*, vol.56, pp.501-509, May 2010.

[16] M. A. Al-Mouhamed, O. Toker, and A. Al-Harthy, "A 3-D vision-based man-machine interface for hand-controlled telerobot," *Industrial Electronics, IEEE Transactions on*, vol.52, pp.306-319, 2005.

[17] P. Viola and M. Jones, "Robust real-time object detection," *Computer Vision,*

International Journal on, vol. 2, no. 57, pp. 137–154, 2004.

[18] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *Computer Vision, International Journal on*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[19] L. Dung and M. Mizukawa, " Fast hand feature extraction based on connected component labeling, distance transform and hough transform," *Journal of Robotics and Mechatronics*, Vol.21 No.6, 2009

[20] R. Laganière, *OpenCV 2 computer vision application programming cookbook*: Packt Publ. Limited, 2011.

[21] G. Borgefors, "Distance transformations in digital images," *Computer vision, graphics, and image processing*, vol. 34, pp. 344-371, 1986.

[22] A. Meijster, J. B. T. M. Roerdink, and W. H. Hesselink, "A general algorithm for computing distance transforms in linear time," *Mathematical Morphology and its applications to image and signal processing*, pp. 331-340, 2002.

[23] 鍾振方，"基於凹槽匹配法之即時手勢辨識系統設計，" 國立交通大學電機學院電控工程研究所碩士論文，民國 101 年。