國 立 交 通 大 學

電機與控制工程學系

碩 士 論 文

智慧型手勢辨識系統設計

Intelligent Hand Gesture Recognition System Design

研 究 生：洪新光

指導教授：陳永平　教授

中 華 民 國 九 十 八 年 六 月

智慧型手勢辨識系統設計

Intelligent Hand Gesture Recognition System Design

研　究　生：洪新光　　　　Student：Iman Hung

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

國　立　交　通　大　學

電機與控制工程學系

碩　士　論　文

A Thesis
Submitted to Department of Electrical and Control Engineering
College of Electrical Engineering and Computer Science
National Chiao Tung University
In Partial Fulfillment of the Requirements
For the degree of Master
In
Electrical and Control Engineering
June 2009

Hsinchu, Taiwan, Republic of China

中華民國九十八年六月

智慧型運手勢辨識系統設計

學生：洪 新 光　　　　　　　　　指導教授：陳 永 平 博士

國立交通大學電機與控制工程學系

摘要

本論文主要目的是設計智慧型手勢辨識系統，此系統是根據人腦所認知手姿態狀態來識別不同的手勢，其中有九種手勢可以被此系統來描述，包含“向左”、“向右”、“左轉”、“右轉”、“向上”、“向下”、“熱機”、“追縱”和 “訓練”。手姿態的認知以及手勢的識別可以透過類神經網路的學習來處理，首先可利用觸發式類神經網路來達成手姿態的認知，再借由手勢分類器來完成手勢的識別。其中手勢分類器可分為前饋式類神經網路和遞迴式類神經網路兩種類型，雖然兩者都可以達到很好的手勢識別效能，但是仍以遞迴式的類神經網路為佳。

Intelligent Hand Gesture Recognition System Design

Student: Iman Hung Advisor: Dr. Yon-Ping Chen

Department of Electrical and Control Engineering
National Chiao Tung University

ABSTRACT

The main purpose of this thesis is to design the intelligent hand gesture recognition system, which can recognize different hand gestures according to cognitive posture states of human brain. There are nine hand gestures which can be described by this system, including "Turn right", "Turn left", "Upward", "Downward", "Right around", "Left around", "Warming", "Following" and "Learning". The cognition of hand posture states and recognition of hand gestures can be learned by neural network. A hand gesture analyzer, composed of a repeated state retriever and a gesture classifier, is applied to recognize the hand gestures. The hand gesture is closely related to the change of hand posture states; therefore, a repeated state retriever is used to turn hand posture state sequence into triggered state sequence, which can be further classified by the gesture classifier. The gesture classifier can be implemented by two types of neural network, feed-forward and recurrent. It can be shown that both types of gesture classifier can well recognize the hand gestures. However, since the feed-forward classifier is often interfered by undefined hand posture state sequence, the recurrent classifier has a better result in had gesture recognition.

Acknowledgement

在研究所的兩年中，首先我要感謝指導教授 陳永平教授的諄諄教導，讓我在表達能力、思考邏輯以及英文寫作上都有大幅的進步，也使我得以順利完成此篇論文。也要感謝桓展、世宏兩位學長在我遇到問題時幫助、陪伴我解決問題與提供寶貴的建議。最後，謝謝口試委員 楊谷洋教授與 張浚林教授提供寶貴的意見，讓整篇論文更加完整。

另外，感謝可變結構控制實驗室的承育、楊庭、瑋哲學弟、傳源學弟的幫助，讓我順利走過研究所的兩年歲月。最後特別感謝父母對於我的支持、鼓勵以及協助。謝謝你們！

謹以此篇論文獻給所有照顧我、關心我的親戚朋友們。

洪新光 2009.7.15

Contents

List of Figures

List of Tables

Chapter I

Introduction

1.1 Motivation

The field of human-robot interaction, how to communicate with a robot instinctively and directly is a major challenge. As using hand gestures is a natural way for interaction between people, hand detection and hand gesture recognition could be essential to human-robot interaction. It is worthwhile to research the area of human-robot interaction. However, there are still lots of limitation in vision cognitions of robot make the robot unintelligent. The human is intelligent is due to their learning and reasoning ability. The so-called learning ability is to learn and memorize something and reasoning ability is to reasoning something with experience. Therefore, a gifted artificial intelligent neuron is birth to emulated brain neuron. Basically, the artificial neural network based on the human neural network contains many neurons connected with synaptic weights. Thus, it is intelligent by learning, recalling, and reasoning from test data like a human brain.

The robot uses the camera as the eye which is completely different from human in vision cognition. In order solve above problems, we develop a human-like system which is according to human vision of cognition.

1.2 Related Works

Human Computer Interaction (HCI) is a very important research area. Researchers try to create convenient, intuitive and useful interface to improve the communication between human and computer. Keyboard and mouse are the most popular interface for personal computer. However, the current definition of”computer” is not only the personal computer but also means intelligent machine system, intelligent space, etc. For these”new type computer”, the standard keyboard and mouse are not satisfying the requirement. Therefore, vision based hand gesture interface is an important research topic. But recognizing gestures are a task which needs pattern recognition, machine learning, even motion modeling techniques. In the past, researchers have spent effort on hand gesture recognition. In this section, we will review hand recognition, and machine learning techniques.

1.2.1 Hand recognition

The human hand is a complex articulated object consisting of many connected parts and joints. Considering the global hand pose and each finger joint, the human hand motion has roughly 27 degrees of freedom (DOFs) [1]. To use human hands as a natural HCI, glove-based devices, such as the CyberGlove, have been used to capture human hand motions. However, the gloves and their attached wires are still quite cumbersome and awkward for users to wear, and moreover, the cost of the glove is often too expensive for regular users. With the latest advances in the fields of computer vision, image processing, and pattern recognition, real-time vision-based hand gesture classification is becoming more and more feasible for human–computer interaction in Virtual environment. In this thesis to make the image processing easier for the vision-based hand gesture recognition, color-based algorithm is implemented to meet the real-time performance, accuracy and robustness requirements.

1.2.2 Machine Learning

In recent years, many researchers have been devoted to developing artificial intelligence system, consisted of following theory, High-level Vision [2], representation and reasoning [3], spatial and temporal reasoning [3-7] and neural network[7, 8]. In general, the spatial temporal reasoning adopts the concept of state machine to describe a hand gesture represented by a specified state sequence; however, it is not intelligent to design a state machine manually. If a new state is added, the state machine must be redesign and all the transition probabilities should be changed accordingly which makes the spatial temporal reasoning not extendable. In order to solve above problems, we proposed a system combined neural network and spatial temporal reasoning together. However, it is difficult only plain neural network to learning the motion movement related time; therefore, a recurrent neural network is introduced for learning state machine automatically [9-18].

1.3 Organization

The thesis contains five chapters. The introduction is described in this chapter. Chapter 2 describes the basic neural network theory. The Intelligent Hand Gesture Recognition System is implemented by neural network explained in detail in Chapter 3 and the simulation results are demonstrated in Chapter 4. Finally, the conclusion is given in Chapter 5.

Chapter 2

Intelligent algorithms survey

Learning has long been and will continue to be a key issue in intelligent algorithms and systems design. By emulating the behavior of human learning the high levels such as symbolic processing and low levels such as neuronal processing has long been a dominant interest among researchers worldwide.

2.1 Feed-forward Neural Network

An artificial neural network (ANN) demonstrates the ability to learn, recall, and generalize from training patterns or data. Artificial neural networks which are modeled after the physical architecture of the human brain is proposed to simulate the learning function for intelligent machine. Therefore, ANN is highly interconnected by a large of processing elements, which are also called artificial neuron or neuron simply, and its connective behavior is like human brain.

A feedforward neural network is an ANN where connections among the elements do not form a directed cycle, as shown in Fig-1. The feed forward neural network was the first and arguably simplest type of ANN. In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network.

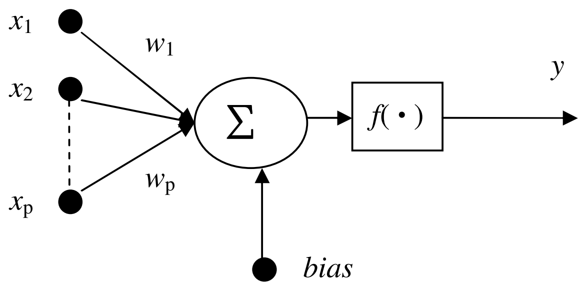


Figure 2.1 ANN

2.2 Recurrent neural network

A recurrent neural network (RNN) is a class of neural network where connections among elements form at least a directed cycle. This creates an internal state of the network which allows it to exhibit dynamic temporal behavior.

Recurrent neural networks are devised differently from feedforward neural networks, both when analyzing their behavior and training them. Recurrent neural networks can also behave chaotically and dynamical system theory is often used to model and analyze them. Unlike feedforward neural network, the RNN is a kind of temporal learning structure and can learn time sequence process task. Recently, investigators have paid more attention to the RNN and developed lots of RNN-related architecture and learning algorithms. The most famous recurrent networks are first-order and second-order RNNs. The first-order RNN is also known as Elman network. Both will be applied in this thesis.

2.2.1 Elman network

An Elman network is known to have memories in the structure and able to represent time in an implicit way. The basic structure of Elman network is shown in Figure 2.2 and formed by four layers including one input layer, one hidden layer, one context layer and one output layer. Clearly, the current input of the context layer receives the previous output $\boldsymbol{x}(k-1)$ from the hidden layer. The output $\boldsymbol{x}(k)$ of the hidden layer depends on the input layer and the context layer, and is described by the following activation function:

$$x_j(k) = f\left(h_j(k)\right) \quad , \qquad j=1, 2, \ldots, r \tag{2-1}$$

where

$$h_j(k) = \sum_{p=1}^{m} v_{jp} u_p(k) + \sum_{q=1}^{r} z_{jq} x_q(k\text{-}1) + q_j \tag{2-2}$$

Note that v_{jp} and z_{jq} are the weights connecting node p of input layer and node q of context layer to node j of hidden layer, and θ_j represents the threshold. After $x(k)$ are obtained, the output layer can be calculated as:

$$y_p(k) = g\left(s_p(k)\right), \qquad p=1, 2, \ldots, n \tag{2-3}$$

where

$$s_p(k) = \sum_{q=1}^{r} w_{pq}\, x_q(k) + \phi_p \tag{2-4}$$

with the weight w_{pq} connecting node p in the hidden layer to node q in the output layer and the threshold ϕ_p.

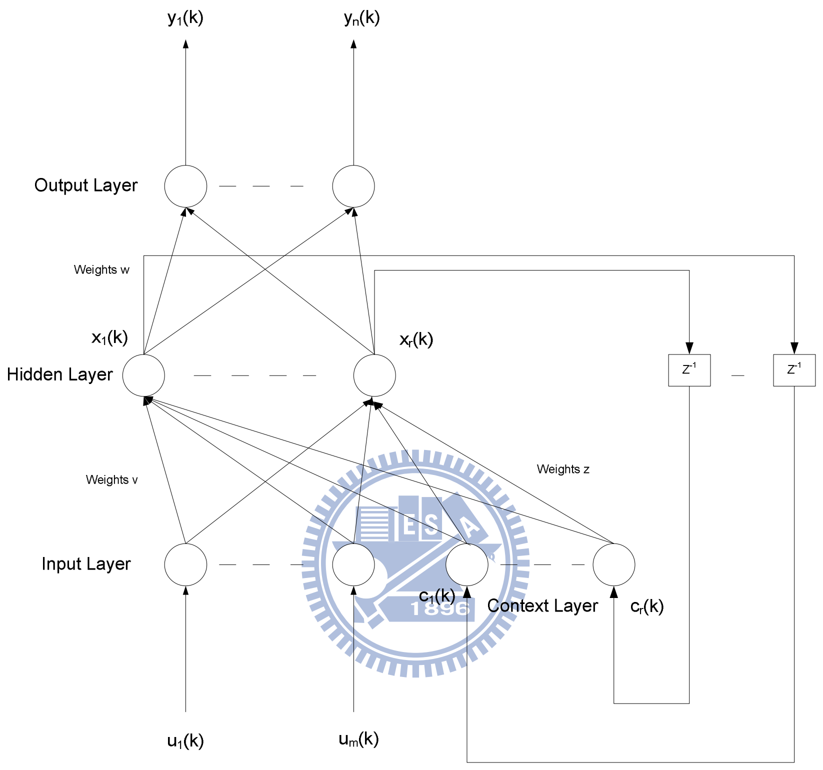


Figure 2.2 Elman network

In general, the summed squared error (SSE) is chosen as the cost function, expressed as

$$\boldsymbol{E}=\sum_{k=1}^{l}e(k)=\frac{1}{2}\sum_{k=1}^{l}\sum_{p=1}^{n}\left(d_p(k)-y_p(k)\right)^2, \quad k=1, 2,\ldots, l \quad (2\text{-}5)$$

where $d_p(k)$ is the desired output, l is the total number of available training samples and n is the total number of output nodes.

To train an Elman network, the gradient of the output error with respect to the weights is calculated, and the weights are incrementally adjusted to reduce the output error by the so-called Back Propagation Through Time (BPTT). According to gradient descent method, each weight change in the network should be proportional to the negative gradient of the cost function (1-5), and defined as

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} = -\eta \frac{\partial \left(\sum_{k=1}^{l} e(k) \right)}{\partial w_{ij}} = -\eta \sum_{k=1}^{l} \frac{\partial e(k)}{\partial w_{ij}} \tag{2-6}$$

where η is the learning rate. Based on the chain rule, it can be further rearranged as

$$\begin{aligned} \Delta w_{ij} &= -\eta \sum_{k=1}^{l} \frac{\partial e(k)}{\partial y_p(k)} \cdot \frac{\partial y_p(k)}{\partial s_p(k)} \cdot \frac{\partial s_p(k)}{\partial w_{ij}} \\ &= \eta \sum_{k=1}^{l} \delta_p(k) \cdot \frac{\partial s_p(k)}{\partial w_{ij}} \\ &= \eta \sum_{k=1}^{l} \delta_p(k) \cdot x_j(k) \end{aligned} \tag{2-7}$$

Note that the truth of $\frac{\partial s_p(k)}{\partial w_{ij}} = x_j(k)$ is directly derived from (2-4). Besides, from (1-5) and (1-3), the function $\delta_p(k)$ can be also express as

$$\begin{aligned} \delta_p(k) &= \frac{\partial e(k)}{\partial y_p(k)} \cdot \frac{\partial y_p(k)}{\partial s_p(k)} \\ &= \left(d_p(k) - y_p(k) \right) \cdot g'\left(s_p(k) \right) \end{aligned} \tag{2-7}$$

With the same process, the weight changes for *z* and *v* are found as

$$\Delta v_{jp} = \eta \sum_{k=1}^{l} \delta_j(k) x_p(k) \tag{2-8}$$

$$\Delta z_{jq} = \eta \sum_{k=1}^{l} \delta_j(k) h_q(k-1) \tag{2-9}$$

After the weight changes are obtained, the weights will be updated correspondingly as below

$$w_{ij} = w_{ij} + \Delta w_{ij} \tag{2-10}$$

$$v_{jp} = v_{jp} + \Delta v_{jp} \tag{2-11}$$

$$z_{jq} = z_{jq} + \Delta z_{jq} \tag{2-12}$$

These weights will be changed for every learning step until all the training sets are used. As for the learning cycle, it will be terminated when the total error satisfies the requirement.

2.2.2 Second-order recurrent neural network

Second-order recurrent neural network can directly identify different sequences which are grammatical or ungrammatical. The architecture of second-order RNN proposed by Giles is shown in Fig.2.3 which consists of two layers connected by weights. The first layer has two groups of neurons: three state neurons $\{S_0(k),\ S_1(k),\ S_2(k)\}$ and two input neurons $\{I_0(k),\ I_1(k)\}$. Define the current-state vector as $\boldsymbol{S}(k)=\left[S_0(k)\ S_1(k)\ S_2(k)\right]^T$ and the input vector as $\boldsymbol{I}(k)=\left[I_0(k)\ I_1(k)\right]^T$. The output layer includes three output neurons $\{S_0(k+1),\ S_1(k+1),\ S_2(k+1)\}$ and then define the output vector as $\boldsymbol{S}(k+1)=\left[S_0(k+1)\ S_1(k+1)\ S_2(k+1)\right]^T$.with one special neuron $S_0(k+1)$ as the indicator in the output layer to indicate the classification decision of the network, i.e., whether the input string is grammatical or ungrammatical. An index T_p for the p^{th} training string is needed for the supervised learning process. The forward propagation (with second-order connection weights) can be defined using the following formula

$$S(k+1)=\sum_p\sum_q w_{pq}\,S_p(k)\,I_q(k) \tag{2-13}$$

with the weight w_{pq} connecting node p of the recurrent output and node q of the input layer to the hidden layer. The objective function E to be minimized is defined as below

$$E=\frac{1}{2P}\sum_{p=1}^{P}\boldsymbol{e}_p^2=\frac{1}{2P}\sum_{p=1}^{P}\left(\boldsymbol{T}_p-S_0(k+1)\right)^2 \tag{2-14}$$

where P denotes the total number of strings in the training set.

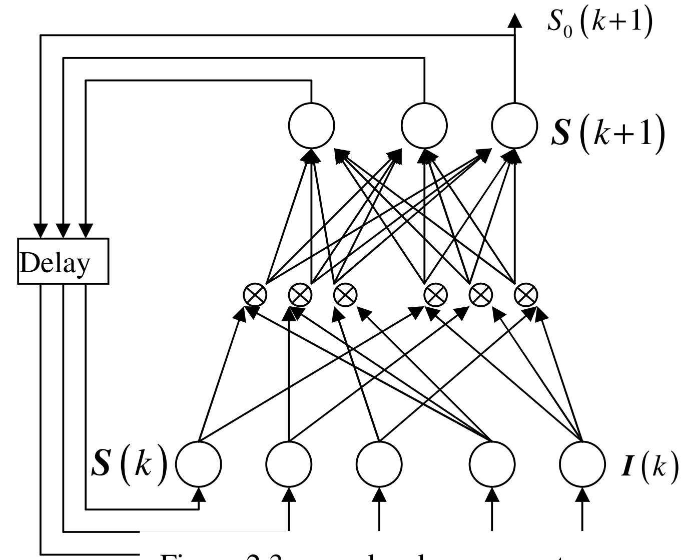


Figure 2.3 second-order recurrent

A convenient way to view a second-order recurrent network that deals with binary sequence is to decompose the network structure into two separate component networks, net0 and net1, controlled by an “enabling” or “gating” as shown in Fig.2.4. The network consists of two first-order recurrent networks with shared state nodes. The state node values are copies back to both net0 and net1 after each time step, and the input sequence acts as a switching control to enable or disable one of two nets. For example, when the current input is 1, net1 is enabled while net0 is disabled. The state node values are then determined by the state node values from the previous time step weighted by the weights in the net1.

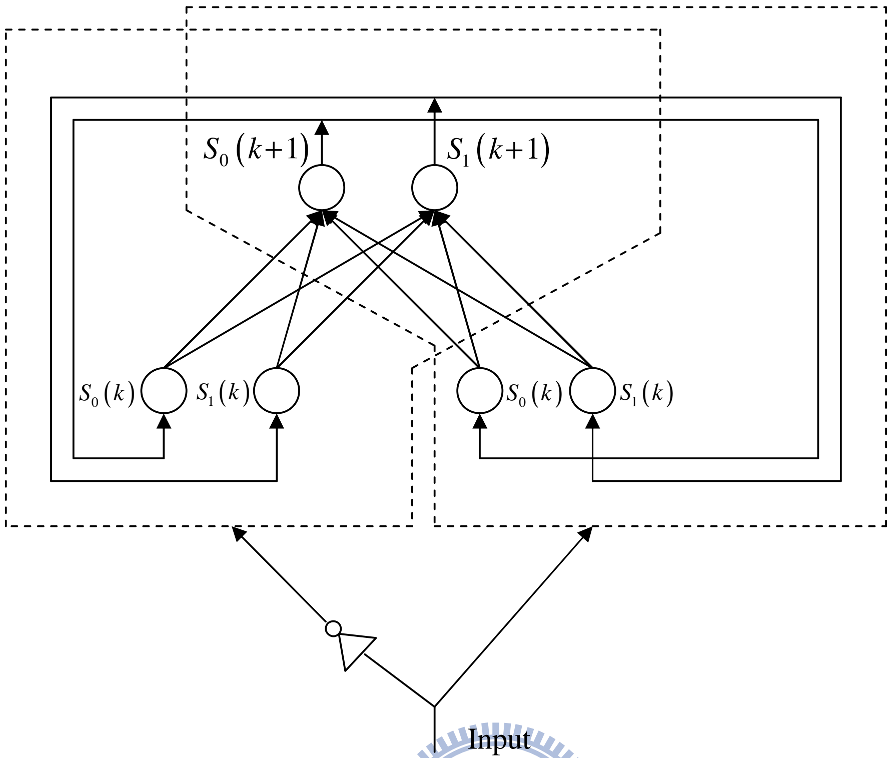


Figure 2.4 second-order recurrent neural network

Chapter III

System structure

In general, the so-called spatial temporal learning is adopted to describe hand gesture under the assumption that each hand gesture represented by a state sequence in fixed length. However, a practical hand gesture usually occur in variable time duration, which causes a state sequence in fixed length fail to deal with such situation. To improve this problem, a novel technology called spatial temporal learning is proposed for the hand gesture recognition system represented by state sequences not in fixed length.

In the literature of hand gesture recognition, there are two important definitions listed as below:

1. Hand posture is a static hand poses and hand location without any movement involved.
2. Hand gesture refers to a sequence of hand postures that are connected by continuous motions over a short time span with the intent to convey information or interact with computer.

Intelligent hand gesture recognition system with spatial temporal learning will be introduced as following. The system structure of the proposed hand gesture recognition system shown in Figure 3.1 includes image data retriever, posture data generator, posture state encoder, and gesture analyzer. The main goal of the proposed hand gesture recognition system is used to recognize hand gestures of the relative sequence of hand posture captured by a CCD camera.

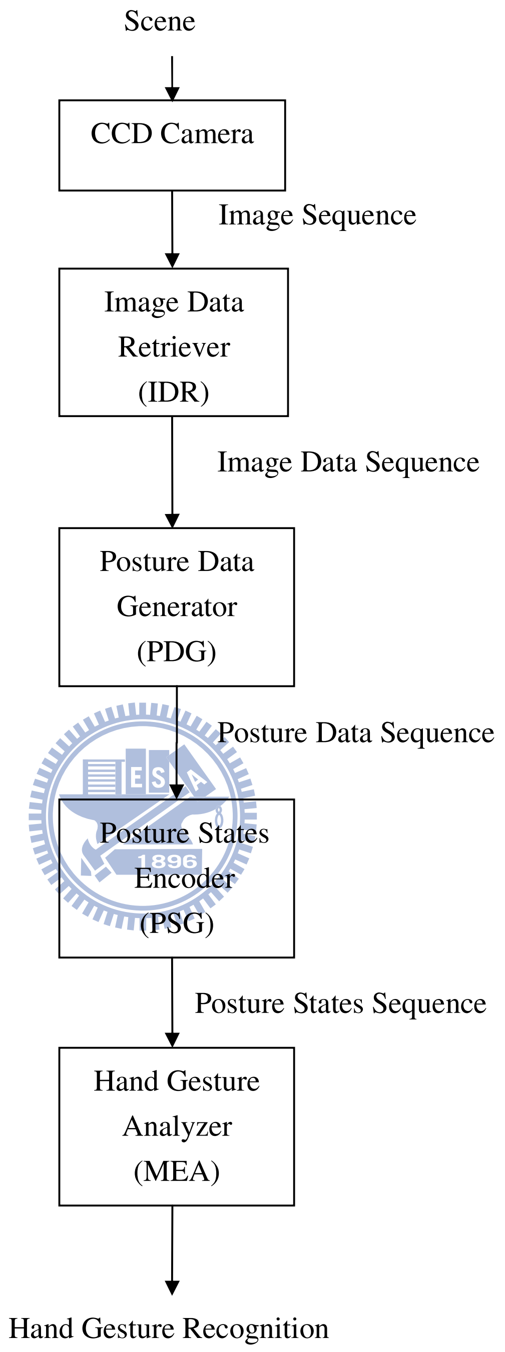


Figure 3.1 Intelligent Hand Gesture Recognition Systems

3.1.1 Image Data Retriever

The skin color and the hand shape are image features that are frequently used for hand posture detection. Nevertheless, color-based algorithms often face the difficulty in distinguishing objects which have similar color with the hand. To solve this problem, black background and white gloves is required to make the hand detection more steady and accurate.

After the sequence of images is captured by the CCD camera, the image data retriever shown in Figure 3.2 retrieves useful information of each image as the image data by image color processing and data extraction. The image data captured at time *k* includes center hand position, back of hand indicator, fingers average position and thumb position. For the convenience of data computation, let the image data be denoted by a vector form as

$$\boldsymbol{i}_k = \begin{bmatrix} Xhnd & Yhnd & Bhnd & Xfng & Yfng & Xthmb & Ythmb \end{bmatrix}^T \qquad (3\text{-}1)$$

where *X*,*Y* and *B* represent the horizontal position, vertical position and back of the hand respectively, while *hnd*, *fng* and *thmb* denote the hand, finger and thumb respectively. More detail definitions of the components of $\boldsymbol{i}_k$ are given in Table 3.1.

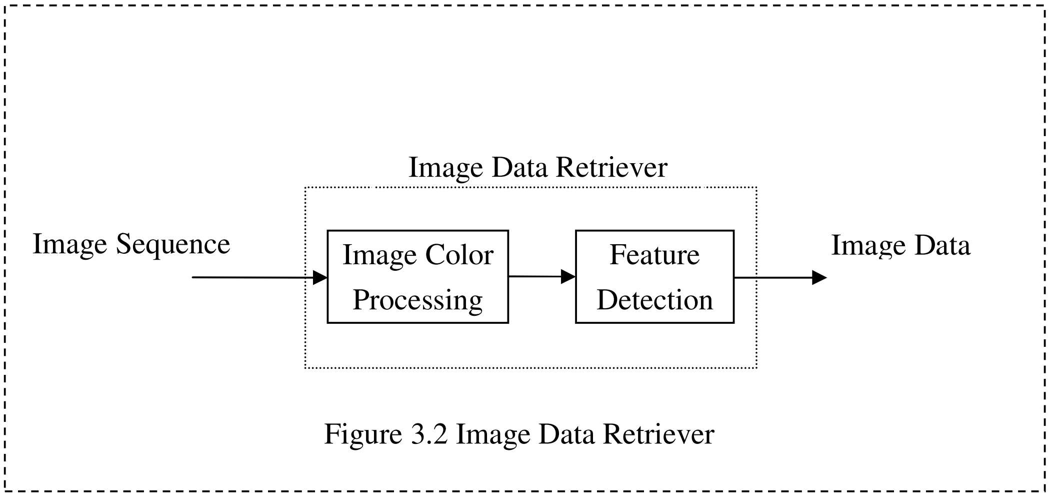


Figure 3.2 Image Data Retriever

Table 3.1 image data definition

Feature	Image Data Components	Definition
Hand	*Xhnd*	The average horizontal position of hand.
	Yhnd	The average vertical position of hand.
	Bhnd	The indicator of back hand.
Fingers	*Xfng*	The average horizontal position of fingers.
	Yfng	The average Vertical position of fingers.
Thumb	*Xthmb*	The average horizontal position of thumb.
	Ythmb	The average vertical position of fingers.

3.1.2 Posture Data Generator

With an image data $\boldsymbol{i}_k$ at time *k,* the posture data generator depicted in Figure 3.3 produces a posture data $\boldsymbol{p}_k$, which contains four posture situations at time *k*, namely *Fpos*, *Hpos*, *Tpos* and *Hcon*. Table 3.2 shows their definitions. Moreover, to suitably assign values for these four posture situations, some condition variables should be defined first as below:

$$fi = max\left(\boldsymbol{i}_k(2),\boldsymbol{i}_k(5)\right) = max\left(Yhnd, Yfng\right) \tag{3-2}$$

$$ti = max\left(\boldsymbol{i}_k(1),\boldsymbol{i}_k(6)\right) = max\left(Xhnd, Xthmb\right) \tag{3-3}$$

$$\begin{aligned} hi &= \sqrt{\left(\boldsymbol{i}_k(1)-\boldsymbol{i}_k(4)\right)^2+\left(\boldsymbol{i}_k(2)-\boldsymbol{i}_k(5)\right)^2} \\ &= \sqrt{\left(Xhnd-Xfng\right)^2+\left(Yhnd-Yfng\right)^2} \end{aligned} \tag{3-4}$$

where *fi* is the maxima value of *Yhnd* and *Yfng*, *ti* is the maxima value of *Xhnd* and *Xthmb* and *hi* is the distance between hand and finger.

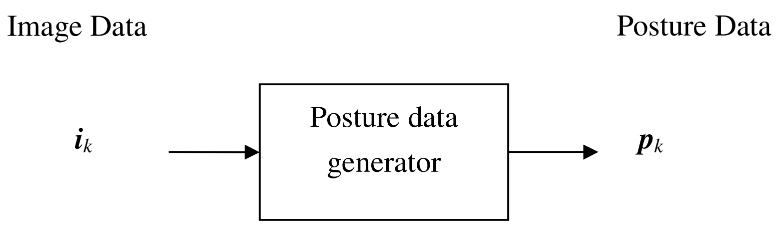


Figure 3.3 Posture data generator

The posture data trained by feed-forward neural network at time *k* includes four components, *Fpos*, *Hpos*, *Tpos* and *Hcon*, which are described in Table 3.2 and expressed as

$$\boldsymbol{p}_k = \left[Hpos \quad Fpos \quad Tpos \quad Hcon\right]^T \qquad (3\text{-}5)$$

Table 3.3 shows the values assigned to these four components in corresponding conditions. Note that the parameter *thc* in the condition of *Hcon* is a threshold and will be discussed later.

Table 3.2 Posture data description

Posture data Components	Description
Hpos	A value to describe the hand surface position in the front side or back side position.
Fpos	A value to describe the relation between hand and fingers whether the fingers is up or below the hand position.
Tpos	A value to describe the position between hand and thumb whether the thumb is right side or left side of the hand position.
Hcon	A value to describe the condition that hand is open or close.

Table 3.3 Posture data definition

Posture data	Output Value	Output Symbol	Condition
Hpos	1	'Front'	$Bhnd = 1$
	-1	'Back'	$Bhnd = 0$
Fpos	1	'Upside'	$fi = 1\ (Yhnd < Yfng)$
	-1	'Downside'	$fi = -1\ (Yhnd > Yfng)$
Tpos	1	'Right'	$ti = 1\ (Xhnd < Xthmb)$
	-1	'Left'	$ti = -1\ (Xhnd > Xthmb)$
Hcon	1	'Open'	$hi > thc$
	-1	'Close'	$hi < thc$

3.1.3 Posture State Encoder

To represent hand postures, Posture State Encoder, called PSE in brief, is implemented by Neural Network which transforms the posture data $\boldsymbol{p}_k$ into the posture state $\boldsymbol{s}_k$ as shown in Fig 3.4. Each posture state represents one hand posture given in Table-3.4. Note that although there are four posture data generated from Posture Data Generator, only three of them, *Hcon*, *Hpos* and *Fpos*, are required to form the posture states, with *Tpos* being treated as don't care. This thesis will focus on nine gestures, named *upward*, *downward*, *turn left*, *turn right*, *left around*, *right around*, *follow*, *learning* and *warming*. Each gesture is determined by a sequence of postures matching to posture states correspondingly shown in Table 3.5. In other words, each gesture is recognized by a sequence of posture states, which are partitioned into two parts including one starting command formed singly by the state S9 and one gesture command formed by two or three posture states chosen from S1 to S8.

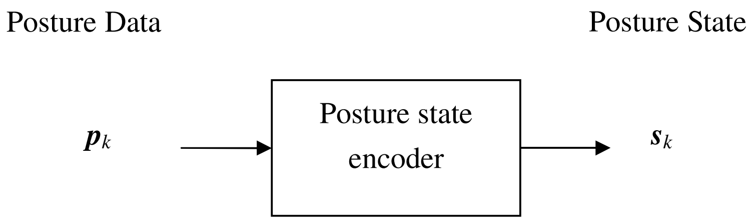
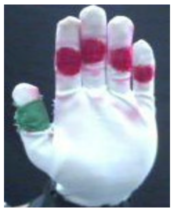
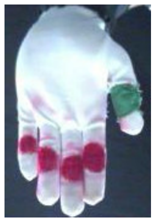
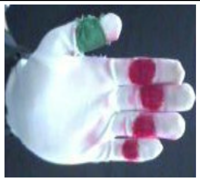
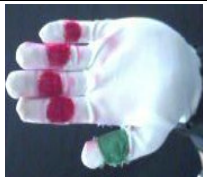
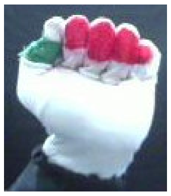
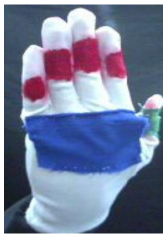
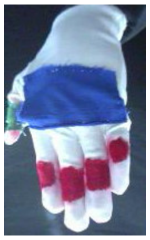
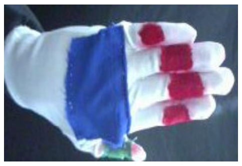
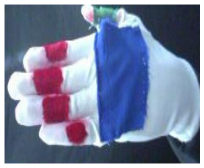


Figure 3.4 Posture State Encoder

Table 3.4 Posture State Encoder

Hcon	*Hpos*	*Fpos*	*Tpos*	S1	S2	S3	S4	S5	S6	S7	S8	S9
0	0	00	X	1	0	0	0	0	0	0	0	0
0	0	01	X	0	1	0	0	0	0	0	0	0
0	0	10	X	0	0	1	0	0	0	0	0	0
0	0	11	X	0	0	0	1	0	0	0	0	0
0	1	00	X	0	0	0	0	1	0	0	0	0
0	1	01	X	0	0	0	0	0	1	0	0	0
0	1	10	X	0	0	0	0	0	0	1	0	0
0	1	11	X	0	0	0	0	0	0	0	1	0
1	0	00	X	0	0	0	0	0	0	0	0	1

Table 3.5 Postures of Posture States

S1	S2	S3	S4	S9
				
S5	S6	S7	S8	
				

3.1.4 Hand Gesture Analyzer

For a hand in motion, the posture states are continuously captured by the CCD camera in sequence. There are two kinds of posture states in the sequence, named as single posture state and repeated posture state. When a posture state s_k at time k is different to the former $s_{k\text{-}1}$ and the latter s_{k+1}, it is called the single posture state, otherwise it is called the repeated posture state. Because the single posture state always happens from the change of posture states, it is treated as undesirable noise during the recognition process. The trigger net is designed to delete the single posture state away from the sequence and the resulted output is called the triggered state sequence. Then, a gesture classifier is set up following the trigger net to form the Hand Gesture Analyzer or HGA in short, as shown in Fig 3.5. The gesture classifier determines the gesture concerning the triggered posture states in terms of the following gesture vector, expressed as

$$\mathbf{g}_k = \begin{bmatrix} uw_k & dw_k & tr_k & tl_k & ra_k & la_k & w_k & f_k & l_k \end{bmatrix}^T \tag{3-6}$$

Clearly, the nine gestures are represented by the nine elements,

$$\mathbf{g}_k(i), \quad \text{for } i = 1,2,\ldots 9 \tag{3-7}$$

described in Table 3.6, respectively.

The detail learning structure of the Hand Gesture Analyzer will be described on the following section.

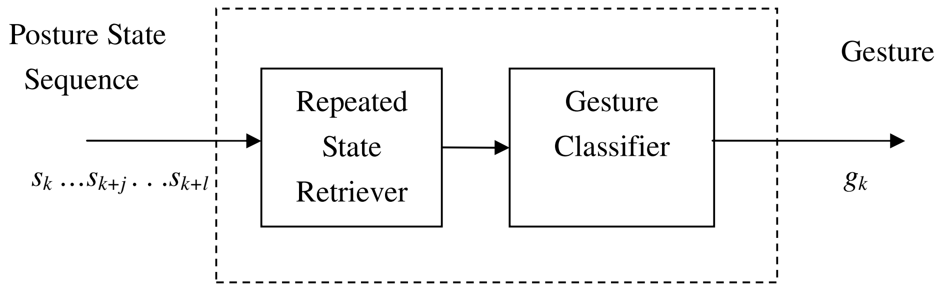


Figure 3.5 Hand Gesture Analyzer

Table 3.6 Gesture description

g_k	Gesture	Description
$g_k(1)$	Upward	The gesture to command the eyes robot to turn up the head to upward 90^o then turns back to the initial position.
$g_k(2)$	Downward	The gesture to command the eyes robot to turn down the head to downward 90^o then turns back to the initial position.
$g_k(3)$	Turn right	The gesture to command the eyes robot to turn the head to the right side 90^o then turns back to the initial position.
$g_k(4)$	Turn Left	The Gesture to command the eyes robot to turn the head to the left side 90^o then turns back to the initial position.
$g_k(5)$	Right around	The gesture to command the eyes robot to rotate the head clockwise direction 360^o then back to the initial position.

$\boldsymbol{g}_k(6)$	Left around	The gesture to command the eyes robot to rotate the head counter clockwise direction 360° then back to the initial position.
$\boldsymbol{g}_k(7)$	Warming	The gesture to command the eyes robot to check all the axes condition to make sure all the axes can normally work.
$\boldsymbol{g}_k(8)$	Following	The gesture to command the eyes robot to tracking the hand for about 30 seconds.
$\boldsymbol{g}_k(9)$	Learning	The gesture to command the eyes robot to learning the hand position relative trajectory.

3.2 Machine learning

To describe a hand gesture by a posture state sequence, it is often based on a fixed-length sequence of posture states. However, a hand gesture usually happens during an uncertain time; for example one hand posture can change to another hand posture quickly or slowly and thus it is difficult to represent a hand gesture by a fixed-length state sequence. To solve this problem, the hand gesture analyzer (HGA) contains a trigger net and a gesture classifier in Fig 3.5, which is developed to learn a posture state sequence unfixed length. As for the gesture classifier, two types of learning structure, called the feed-forward classifier and the recurrent classifier, will be introduced in the followings.

3.2.1 Repeated State Retriever

To analyze a hand gesture represented by a hand posture state sequence not in fixed length, the key component to be used is the Repeated State Retriever shown in Fig.3.6, which is composed of the Single State Eliminator, the Repeated State Processor, and the Trigger Net.

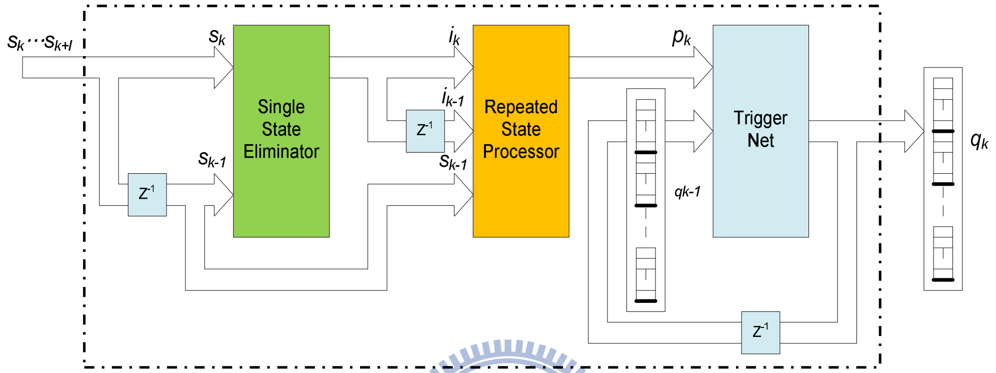


Figure 3.6 Repeated State Retriever

The Single State Eliminator and the Repeated State Processor are implemented by the Feedforward Neural Network, while the Trigger Net is implemented by the Recurrent Neural Network. The detailed structure of Repeated State Retriever is depicted in Fig.3.6. The first stage is the Single State Eliminator and described as

$$i_k = \begin{cases} 1 & \text{for } s_k = s_{k-1} \\ -1 & \text{for } s_k \neq s_{k-1} \end{cases} \tag{3-8}$$

for k=1,2,…,L, where the output i_k is an index to represent whether the posture state s_k at time k is the same as the former posture state s_{k-1} or not. An example is given in Table 3.8, when the time k=3, 4, 8, 10, 11, 12 and 15 are shows the cases when the current posture states are different with the former posture state, at these time the indexes are −1 and otherwise the indexes are 1.

The Repeated State Processor, which is the second stage of the Repeated State Retriever, is described as

$$p_k = \begin{cases} s_{k-1} & \text{for } i_k = -1 \text{ and } i_{k-1} = 1 \\ 0 & \text{otherwise} \end{cases} \tag{3-9}$$

for k=1,2,…,L, where the output p_k is equal to the former posture state s_{k-1} for the case that the index is changed from i_{k-1}=1 to i_k=−1 and set to be 0 for the other cases. Table 3.7 shows all the cases related to (3-9). The given example at Table 3.9 shows at the time k=2, 7, 9 and 14 the index is changed from i_{k-1}=1 to i_k=−1 and the output p_k is equal to the former posture state s_{k-1} otherwise the output p_k is equal to 0.

Table 3.7 Index condition

i_{k-1}	i_k	Condition
1	1	$s_{k-2} = s_{k-1}$ and $s_{k-1} = s_k$
1	-1	$s_{k-2} = s_{k-1}$ and $s_{k-1} \neq s_k$
-1	1	$s_{k-2} \neq s_{k-1}$ and $s_{k-1} = s_k$
-1	-1	$s_{k-2} \neq s_{k-1}$ and $s_{k-1} \neq s_k$

The third stage of the Repeated State Retriever is the Trigger Net, which is described by

$$q_k(i) = q_{k-1}(i)\ ,\ i = 1,2,\dots,N, \qquad \text{for } p_k = 0 \tag{3-10}$$

and

$$\begin{cases} q_k(1) = p_k \\ q_k(i+1) = q_{k-1}(i),\ i = 1,2,\dots,N-1 \end{cases} \qquad \text{for } p_k \neq 0 \tag{3-11}$$

Clearly, the Repeated State Retriever rejects all the state except the final state of each repeated posture state sequence. The example of trigger net is given at Table 3.10, at the time k=2, 7, 9 and 14 the trigger net is just triggered non zero processed posture state p_k and rejects the zero values of processed posture state p_k. When the trigger net is reject the processed posture state p_k, $q_k(i)$ is equal to the former output $q_{k-1}(i)$, otherwise $q_k(1)$ is equal to p_k.

Here is the example how the Repeated State Retriever work with the input posture state is [5 5 5 7 1 1 1 1 3 3 9 1 2 2 2 3] with the output length N is set as 3 firstly, the Single State Eliminator will be convert the input posture state to the posture state index as shown at Table 3.8.

Table 3.8 Example of Single State Eliminator

k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s_k	5	5	7	1	1	1	1	3	3	9	1	2	2	2	3
s_{k-1}	5	5	5	7	1	1	1	1	3	3	9	1	2	2	2
i_k	1	1	-1	-1	1	1	1	-1	1	-1	-1	-1	1	1	-1

when the current posture state is the same as the former posture state, the posture state index is set as 1(blue rectangle), otherwise will set as -1(red rectangle). Then for the next step, the Repeated State Processor will be observing the transition of the posture state index. As shown at the table 3.7 there is just one condition is considered, that is when the current posture state index is -1 and the former posture state index is 1 as shown at (red rectangle) Table 3.9.

Table 3.9 Example of Repeated State Processor

k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s_{k-1}	5	5	7	1	1	1	1	3	3	9	1	2	2	2	3
i_{k-1}	1	1	-1	-1	1	1	1	-1	1	-1	-1	-1	1	1	-1
i_k	1	-1	-1	1	1	1	-1	1	-1	-1	-1	1	1	-1	-1
p_k	0	5	0	0	0	0	1	0	3	0	0	0	0	2	0

after repeated state sequence is processed, the Trigger Net will trigger the processed state sequence whenever the processed state sequence is not zero as shown at (red rectangle) Table 3.10..

Table 3.10 Example of Trigger Net

k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
p_k	0	5	0	0	0	0	1	0	3	0	0	0	0	2	0
$q_{k-1}(1)$	0	0	5	5	5	5	5	1	1	3	3	3	3	3	2
$q_{k-1}(2)$	0	0	0	0	0	0	0	5	5	1	1	1	1	1	3
$q_{k-1}(3)$	0	0	0	0	0	0	0	0	0	5	5	5	5	5	1
$q_k(1)$	0	5	5	5	5	5	1	1	3	3	3	3	3	2	2
$q_k(2)$	0	0	0	0	0	0	5	5	1	1	1	1	1	3	3
$q_k(3)$	0	0	0	0	0	0	0	0	5	5	5	5	5	1	1

Next, this Repeated State Retriever will be applied to the feed-forward gesture classifier and the recurrent gesture classifier, with different N. There are two kinds of clock mode, synchronous and asynchronous, used in this gesture recognition system. If Repeated State Retriever is applied for feed-forward classifier, then the gesture recognition system adopts synchronous clock mode for recurrent classifier, otherwise, adopts asynchronous clock mode. In synchronous clock mode, the output of feed-forward classifier will be processed according to each coming clock k. In asynchronous clock mode, however, the output of recurrent classifier will be processed according to the output of trigger net which is shifted down. Gesture classifier will be discussed at different clock modes in next section.

3.2.2 Gesture Classifier

Different hand gestures can be recognized by gesture classifier according to different triggered state sequences. The two types of classifier, feed-forward and recurrent, are developed according to different number of outputs of the trigger net. In fact, the length of triggered state sequence can be extended to N, where N is the maximum length of triggered state sequence among all the hand gestures. In this thesis, the number of outputs of the trigger net is set N=3 for feed-forward classifier and N=1 for recurrent classifier.

3.2.2.1. Feed-forward classifier

For the gesture classifier, there are three triggered state sequences at time k, which are denoted by $q_k(1)$, $q_k(2)$ and $q_k(3)$ and sent in parallel to the input layer of the Feed-forward classifier to analyze the nine gestures described in Section 3.4 and represented as a vector $\boldsymbol{g}_k$ shown in Figure 3.7. A fixed length of triggered state sequences for each hand gestures are shown in Table 3.11. These triggered state sequences can be identified by the gesture classifier.

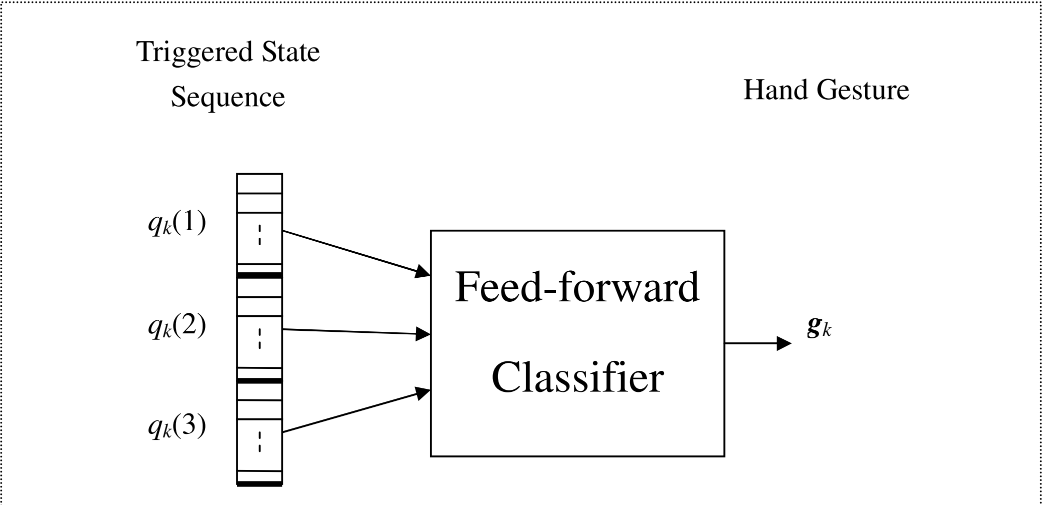


Figure 3. 7. feed-forward classifier

Table 3.11 Hand gesture and triggered state sequence

Hand Gesture	Triggered State Sequence	Output g_k
Upward	$S_9 \rightarrow S_2 \rightarrow S_5$	[1 0 0 0 0 0 0 0 0]
Downward	$S_9 \rightarrow S_1 \rightarrow S_6$	[0 1 0 0 0 0 0 0 0]
Turn right	$S_9 \rightarrow S_4 \rightarrow S_7$	[0 0 1 0 0 0 0 0 0]
Turn Left	$S_9 \rightarrow S_7 \rightarrow S_4$	[0 0 0 1 0 0 0 0 0]
Right around	$S_9 \rightarrow S_1 \rightarrow S_5$	[0 0 0 0 1 0 0 0 0]
Left around	$S_9 \rightarrow S_5 \rightarrow S_1$	[0 0 0 0 0 1 0 0 0]
Warming	$S_9 \rightarrow S_1 \rightarrow S_7$	[0 0 0 0 0 0 1 0 0]
Following	$S_9 \rightarrow S_1 \rightarrow S_4$	[0 0 0 0 0 0 0 1 0]
Learning	$S_9 \rightarrow S_1 \rightarrow S_3$	[0 0 0 0 0 0 0 0 1]

3.2.2.2 Recurrent classifier

A concept of "Transient Pattern" will be presented here for recurrent classifier, the outputs of recurrent classifier will give a transient pattern for learning hand gesture according to state transient condition given in following Table 3.12 and a simple example of transient pattern is given at Table 3.13.

Table 3.12 Transient Pattern

$q_{k-2}\rightarrow q_{k-1}\rightarrow q_k$	$\boldsymbol{g}_k$									Transient
	$g_k(1)$	$g_k(2)$	$g_k(3)$	$g_k(4)$	$g_k(5)$	$g_k(6)$	$g_k(7)$	$g_k(8)$	$g_k(9)$	pattern
$S_x\rightarrow S_9\rightarrow S_1$	0	0.6	0	0	0.6	0	0.6	0.6	0.6	tp_1
$S_9\rightarrow S_1\rightarrow S_3$	0	0	0	0	0	0	0	0	1	X
$S_9\rightarrow S_1\rightarrow S_4$	0	0	0	0	0	0	0	1	0	X
$S_9\rightarrow S_1\rightarrow S_5$	0	0	0	0	1	0	0	0	0	X
$S_9\rightarrow S_1\rightarrow S_6$	0	1	0	0	0	0	0	0	0	X
$S_9\rightarrow S_1\rightarrow S_7$	0	0	0	0	0	0	1	0	0	X
$S_x\rightarrow S_9\rightarrow S_2$	0.6	0	0	0	0	0	0	0	0	tp_2
$S_9\rightarrow S_2\rightarrow S_5$	1	0	0	0	0	0	0	0	0	X
$S_x\rightarrow S_9\rightarrow S_4$	0	0	0.6	0	0	0	0	0	0	tp_3
$S_9\rightarrow S_4\rightarrow S_7$	0	0	1	0	0	0	0	0	0	X
$S_x\rightarrow S_9\rightarrow S_5$	0	0	0	0	0	0.6	0	0	0	tp_4
$S_9\rightarrow S_5\rightarrow S_1$	0	0	0	0	0	1	0	0	0	X
$S_x\rightarrow S_9\rightarrow S_7$	0	0	0	0.6	0	0	0	0	0	tp_5
$S_9\rightarrow S_7\rightarrow S_4$	0	0	0	1	0	0	0	0	0	X
Others	0	0	0	0	0	0	0	0	0	*X*

Table 3.13 example of recurrent classifier

k	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Triggered Sequence	1	9	1	4	9	1	3	9	1	9	2	5	9	7
$g(1)$	0	0	0	0	0	0	0	0	0	0	0.6	1	0	0
$g(2)$	0	0	0.6	0	0	0.6	0	0	0.6	0	0	0	0	0
$g(3)$	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$g(4)$	0	0	0	0	0	0	0	0	0	0	0	0	0	0.6
$g(5)$	0	0	0.6	0	0	0.6	0	0	0.6	0	0	0	0	0
$g(6)$	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$g(7)$	0	0	0.6	0	0	0.6	0	0	0.6	0	0	0	0	0
$g(8)$	0	0	0.6	1	0	0.6	0	0	0.6	0	0	0	0	0
$g(9)$	0	0	0.6	0	0	0.6	1	0	0.6	0	0	0	0	0
Transient Pattern	X	X	tp_1	X	X	tp_1			tp_1		tp_7			tp_{13}

The initial value of outputs $g(i)$, i=1,2,…,9 is set to zeros. The state transient S_9→S_1 has a transient pattern tp_1 as target of hand gesture. The state transient S_9→S_1 →S_5 will cause the Right around hand gesture is triggered for process. The state transient condition S_9→S_3 isn't defined in state transient condition, thus the value of output of hand gesture will clear as zeros. It is obvious that the training sequence, trigger sequence and target of hand gestures, can be obtained from above example for recurrent classifier. For the recurrent classifier, there is triggered state at time k, denoted by $q_k(1)$, sent sequentially to the input layer of the recurrent classifier for analyzes the nine hand gestures represented as a vector $\boldsymbol{g}_k$ shown in Figure 3.8. The recurrent classifier adopted a switch control to decide whether the switch is 'on' or 'off' described as

$$Sc = \begin{cases} 1 & , \text{for } q_k(1) \neq 0 \\ 0 & , \text{for } q_k(1) = 0 \end{cases} \tag{3-16}$$

the switch signal Sc is triggered whenever $q_k(1)$ is non zero. The output of hand gesture $\boldsymbol{g}_k$ perform one calculation according to the outputs of hidden layers h_{m-1} and triggered state $q_k(1)$ when the switch is 'on'.

The architecture of recurrent classifier as shown in Figure 3.9, eight input of neuron is indexed as n_1, n_2,…, n_8. If input state is not zero then the value one will be set to n_j and others will set to zeros. The outputs of recurrent classifier $\boldsymbol{g}_k$ are represented as the nine gestures.

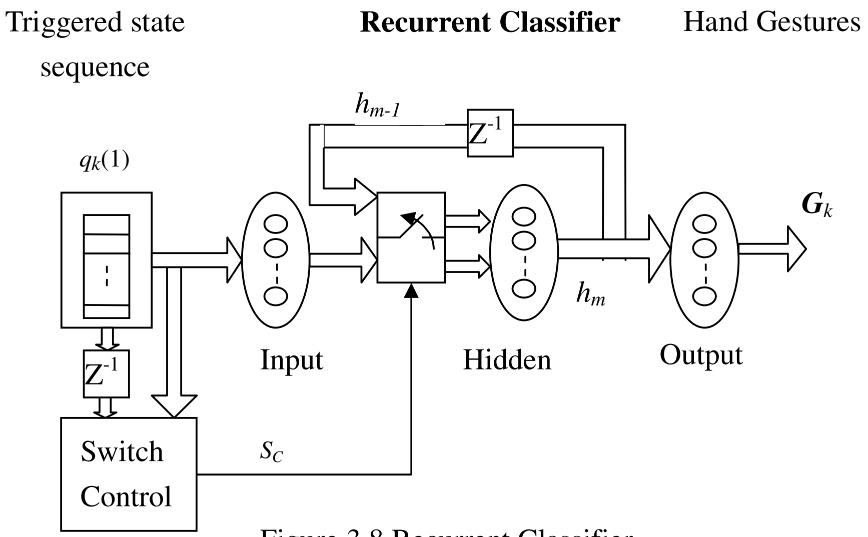


Figure 3.8 Recurrent Classifier

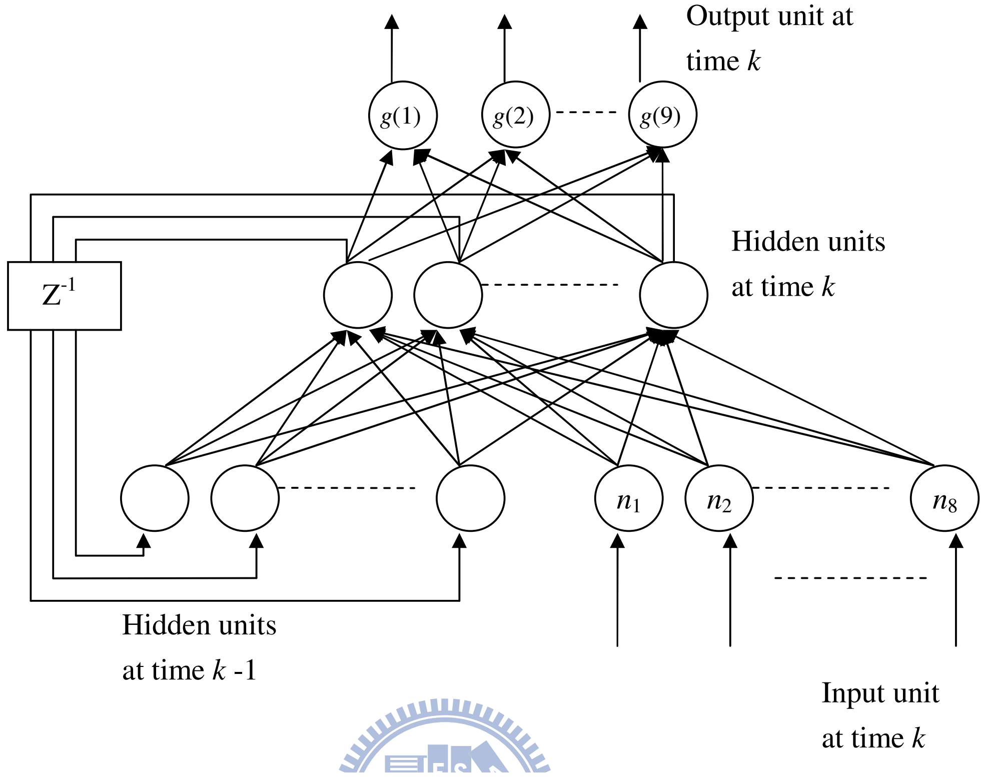


Figure 3.9 the architecture of recurrent classifier

Chapter 4

Experimental result

The experiment results of hand gestures learned by Cascaded Neural Network and Simple Recurrent Network are presented in Section 4.1 and 4.2 respectively. The comparison between these two architectures networks will be discussed in Section 4.3 and a sequence of image with hand gesture recognition were showed in Section 4.4.

4.1 Repeated State Retriever

4000 state sequence is generated according to following equation.

$$s_{k+1} = \begin{cases} s_k & rd \geq 0.6 \\ S(i) & rd < 0.4 \end{cases} \tag{4-1}$$

Where $S(i)$ is the state i which is a random integer ranging from 1, 2, 3…, 9 and rd is a random value range from 0~1. According to equation (3-8) and (3-11), the training sample can be obtained. Input layer of repeated state retriever contains s_k and s_{k-1} entity for input node and output layer of repeated state retriever contains $q_k(1)$, $q_k(2)$,… , $q_k(N)$ entity for output node as shown in Figure 3.5.The Table 4.1 shows the architecture of repeated state retriever and training parameter.

Table 4.1 Architecture of repeated state retriever

	Single State Eliminator	Repeated State Processor	Trigger Net
# of neuron input /output	2/1	3/1	4/3
Transfer function of hidden layer	Hyperbolic tangent		
Training algorithm	Levenberg-Marquardt		

Although, the number of trigger output *N*=3 is used for feed-forward classifier which used to classification of hand gesture. The number of trigger output which *N* is range from 3~8 was implemented here to proving this system is extendable. The decrease of testing-error is followed by increase of number of neuron of hidden layer for different number of trigger output as shown in Figure 4.1.

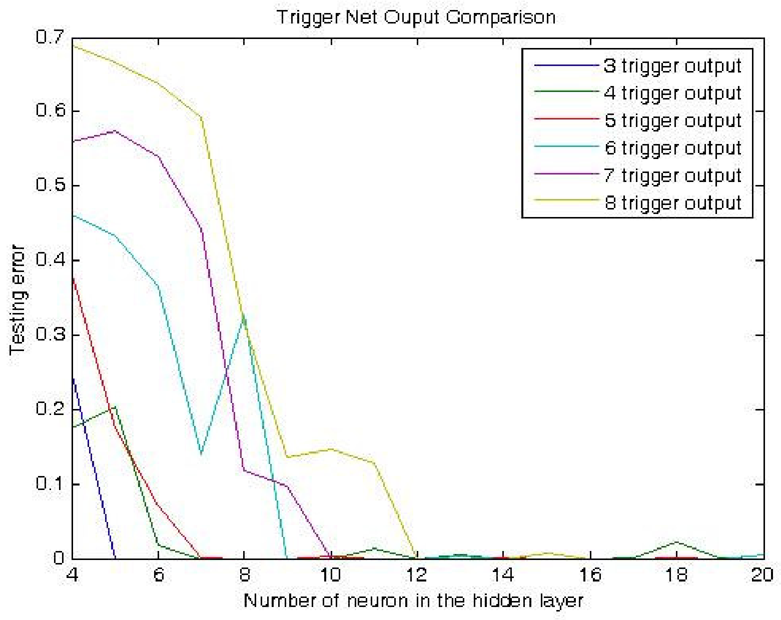


Figure 4.1 testing results with different number of trigger output

Different trigger output number at least needs a neuron number to reach 100% success rate given in Table 4.2 according to Figure 41. Table 4.2 tells that the increase of number of trigger output lead to the increase of neuron number of hidden layer.

Table 4.2 minimum neuron for different number of trigger output

# of trigger output	3	4	5	6	7	8
# of neuron at least	4	7	7	9	11	12
Training samples	4000					
Test samples	1000					
Success rate	100					

4.2 Feed-forward classifier results

The hand gesture machine will be set up and used to generate a string of hand gestures. Each hand gestures will generate triggered state sequence with different probability given in Table 4.3. Up to 4000 length of trigger state sequence will be generated and used for hand gesture classification task.

Table 4.3 training sample of nine hand gesture

Trigger State Sequence	Probability	Hand Gesture
$S_9 \rightarrow S_2 \rightarrow S_5$	0.04	Upward
$S_9 \rightarrow S_1 \rightarrow S_6$	0.04	Downward
$S_9 \rightarrow S_4 \rightarrow S_7$	0.04	Turn right
$S_9 \rightarrow S_7 \rightarrow S_4$	0.04	Turn Left
$S_9 \rightarrow S_1 \rightarrow S_5$	0.04	Right around

$S_9 \rightarrow S_5 \rightarrow S_1$	0.04	Left around
$S_9 \rightarrow S_1 \rightarrow S_7$	0.04	Warming
$S_9 \rightarrow S_1 \rightarrow S_4$	0.04	Following
$S_9 \rightarrow S_1 \rightarrow S_3$	0.04	Learning
others	0.64	Don't Care

Table 4.4 the nine hand gestures with trigger state sequence

Trigger State Sequence	g_k	Hand Gesture
$S_9 \rightarrow S_2 \rightarrow S_5$	[1 0 0 0 0 0 0 0 0]	Upward
$S_9 \rightarrow S_1 \rightarrow S_6$	[0 1 0 0 0 0 0 0 0]	Downward
$S_9 \rightarrow S_4 \rightarrow S_7$	[0 0 1 0 0 0 0 0 0]	Turn right
$S_9 \rightarrow S_7 \rightarrow S_4$	[0 0 0 1 0 0 0 0 0]	Turn Left
$S_9 \rightarrow S_1 \rightarrow S_5$	[0 0 0 0 1 0 0 0 0]	Right around
$S_9 \rightarrow S_5 \rightarrow S_1$	[0 0 0 0 0 1 0 0 0]	Left around
$S_9 \rightarrow S_1 \rightarrow S_7$	[0 0 0 0 0 0 1 0 0]	Warming
$S_9 \rightarrow S_1 \rightarrow S_4$	[0 0 0 0 0 0 0 1 0]	Following
$S_9 \rightarrow S_1 \rightarrow S_3$	[0 0 0 0 0 0 0 0 1]	Learning
others	[0 0 0 0 0 0 0 0 0]	Don't Care

Where S_j is a triggered posture state, for j=1, 2 … 9. Table 4.4, Testing-data is possessed from recording an image sequence captured by camera. The output $\boldsymbol{g}_k$ will be triggered decided by feed-forward classifier. Table 4.5 shows testing results according to Table 4.4.

Table 4.5 testing result for feed-forward classifier

Testing-sample		Target	Testing-sample		Target
$q_k(3)\ q_k(2)\ q_k(1)$	g_k		$q_k(3)\ q_k(2)\ q_k(1)$	g_k	
[1 9 1]	X	X	[9 1 4]	$g_k(8)$	$g_k(8)$
[1 4 1]	X	X	[4 1 9]	X	X
[1 9 1]	X	X	[9 1 3]	$g_k(9)$	$g_k(9)$
[1 3 1]	X	X	[3 1 9]	X	X
[1 9 1]	X	X	[9 1 9]	X	X
[1 9 2]	X	X	[9 2 4]	X	X
[2 4 5]	X	X	[4 5 9]	X	X
[5 9 1]	X	X	[9 1 9]	X	X
[1 9 1]	X	X	[9 1 2]	$g_k(7)$	X
[1 2 4]	X	X	[2 4 5]	X	X
[4 5 9]	X	X	[5 9 2]	X	X
[9 2 4]	X	X	[2 4 5]	X	X
[4 5 9]	X	X	[5 9 1]	X	X
[9 1 6]	$g_k(4)$	$g_k(4)$	[1 6 9]	X	X
[6 9 1]	X	X	[9 1 9]	X	X
[1 9 1]	X	X	[9 1 4]	$g_k(8)$	$g_k(8)$
[1 4 7]	X	X	[4 7 9]	X	X
[7 9 1]	X	X	[9 1 9]	X	X
[1 9 1]	X	X	[9 1 5]	$g_k(5)$	$g_k(5)$
[1 5 1]	X	X	[5 1 9]	X	X
[1 9 1]	X	X	[9 1 6]	$g_k(4)$	$g_k(4)$

[1 6 1]	X	X	[6 1 9]	X	X
[1 9 1]	X	X	[9 1 4]	$g_k(8)$	$g_k(8)$
[1 4 7]	X	X	[4 7 1]	X	X
[7 1 9]	X	X	[1 9 4]	X	X
[9 4 7]	$g_k(1)$	$g_k(1)$	[4 7 9]	X	X
[7 9 7]	X	X	[9 7 4]	$g_k(2)$	$g_k(2)$
[7 4 9]	X	X	[4 9 2]	X	X
[9 2 5]	$g_k(3)$	$g_k(3)$	[2 5 7]	X	X
[5 7 9]	X	X	[7 9 1]	X	X

Table 4.6 testing performance

	All Testing samples	motion Sequence	Undefined sequence
# of test samples	60	10	50
# of misclassified	1	0	1
Success Rate	98.33%	100%	98%

4.3 Simple Recurrent Neural Network results

Table 4.6 provides training sequences for the task of hand gesture classification. Input sequence of Table 4.6 is generated by posture state generator according to Figures 4.2 and Table 4.3. Target of nine gestures are designed according to Figures 3.10. The 10 hand gestures are consisting of 60 sequence length generated for sequential learning in the following table.

Table 4.7 Training samples for nine hand gestures

Input	$\boldsymbol{g}_k$								
$q_k(1)$	$g_k(1)$	$g_k(2)$	$g_k(3)$	$g_k(4)$	$g_k(5)$	$g_k(6)$	$g_k(7)$	$g_k(8)$	$g_k(9)$
1	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0
1	0	0.6	0	0	0.6	0	0.6	0.6	0.6
4	0	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0
1	0	0.6	0	0	0.6	0	0.6	0.6	0.6
3	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0
1	0	0.6	0	0	0.6	0	0.6	0.6	0.6
9	0	0	0	0	0	0	0	0	0
2	0.6	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0
1	0	0.6	0	0	0.6	0	0.6	0.6	0.6
9	0	0	0	0	0	0	0	0	0
1	0	0.6	0	0	0.6	0	0.6	0.6	0.6
2	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0

5	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0
2	0.6	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0
1	0	0.6	0	0	0.6	0	0.6	0.6	0.6
6	0	1	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0
1	0	0.6	0	0	0.6	0	0.6	0.6	0.6
9	0	0	0	0	0	0	0	0	0
1	0	0.6	0	0	0.6	0	0.6	0.6	0.6
4	0	0	0	0	0	0	0	1	0
7	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0
1	0	0.6	0	0	0.6	0	0.6	0.6	0.6
9	0	0	0	0	0	0	0	0	0
1	0	0.6	0	0	0.6	0	0.6	0.6	0.6
5	0	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0
1	0	0.6	0	0	0.6	0	0.6	0.6	0.6
6	0	1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0

1	0	0.6	0	0	0.6	0	0.6	0.6	0.6
4	0	0	0	0	0	0	0	1	0
7	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0
4	0	0	0.6	0	0	0	0	0	0
7	0	0	1	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0
7	0	0	0	0.6	0	0	0	0	0
4	0	0	0	1	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0
2	0.6	0	0	0	0	0	0	0	0
5	1	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0
1	0	0.6	0	0	0.6	0	0.6	0.6	0.6

The recurrent classifier is implemented by recurrent neural network with Levenberg-Marquardt learning algorithms given in Figure 4.7, where input layer contains 4 nodes, two hidden layer contains 17 neurons and 8 neurons with hyperbolic tangent function and output layer contains 3 neurons with saturation function.

Table 4.8 the architecture of recurrent classifier

Architecture of SRN			
	Input layer	Hidden layer	Output layer
Number of layer	1	2	1
Number of neuron	4	[17 8]	3
Transfer function		Hyperbolic tangent	Linear
Training algorithms	Levenberg-Marquardt		

The neural network's training function that updates weight and bias values according to Levenberg-Marquardt learning algorithms. This learning has better performance than gradient descent learning algorithms. Test output value can't exceed the threshold 0.8 by using gradient descent learning algorithms when that hand gesture has occurred. Table 4.8 showed that testing results for nine hand gestures by using previous training table. A threshold 0.8 is used to judge nine gestures is triggered for out. The testing performance given at Table 4.9 shows success result for hand gestures classification.

Table 4.9 test samples for nine hand getures

$q_k(1)$	Test Output		Target	
	g_k	HG	g_k	HG
1	[0 0 0 0 0 0 0 0 0]		[0 0 0 0 0 0 0 0 0]	
9	[0 0 0 0 0 0 0 0 0]		[0 0 0 0 0 0 0 0 0]	
1	[0 0 0 0 0 0 0 0 0]		[0 0.6 0 0 0.6 0 0.6 0.6 0.6]	
4	[0 0 0 0 0 0 0 1 0]	*following*	[0 0 0 0 0 0 0 1 0]	*following*
1	[0 0 0 0 0 0 0 0 0]		[0 0 0 0 0 0 0 0 0]	

9	[0 0 0 0 0 0 0 0 0]		[0 0 0 0 0 0 0 0 0]	
1	[0 0 0 0 0 0 0 0 0]		[0 0.6 0 0 0.6 0 0.6 0.6 0.6]	
3	[0 0 0 0 0 0 0 0 1]	*learning*	[0 0 0 0 0 0 0 0 1]	*learning*
1	[0 0 0 0 0 0 0 0 0]		[0 0 0 0 0 0 0 0 0]	
9	[0 0 0 0 0 0 0 0 0]		[0 0 0 0 0 0 0 0 0]	
1	[0 0 0 0 0 0 0 0 0]		[0 0.6 0 0 0.6 0 0.6 0.6 0.6]	
9	[0 0 0 0 0 0 0 0 0]		[0 0 0 0 0 0 0 0 0]	
2	[0 0 0 0 0 0 0 0 0]		[0.6 0 0 0 0 0 0 0 0]	
4	[0 0 0 0 0 0 0 0 0]		[0 0 0 0 0 0 0 0 0]	
5	[0 0 0 0 0 0 0 0 0]		[0 0 0 0 0 0 0 0 0]	
9	[0 0 0 0 0 0 0 0 0]		[0 0 0 0 0 0 0 0 0]	
1	[0 0 0 0 0 0 0 0 0]		[0 0.6 0 0 0.6 0 0.6 0.6 0.6]	
9	[0 0 0 0 0 0 0 0 0]		[0 0 0 0 0 0 0 0 0]	
1	[0 0 0 0 0 0 0 0 0]		[0 0.6 0 0 0.6 0 0.6 0.6 0.6]	
2	[0 0 0 0 0 0 0 0 0]		[0 0 0 0 0 0 0 0 0]	
4	[0 0 0 0 0 0 0 0 0]		[0 0 0 0 0 0 0 0 0]	
5	[0 0 0 0 0 0 0 0 0]		[0 0 0 0 0 0 0 0 0]	
9	[0 0 0 0 0 0 0 0 0]		[0 0 0 0 0 0 0 0 0]	
2	[0 0 0 0 0 0 0 0 0]		[0.6 0 0 0 0 0 0 0 0]	
4	[0 0 0 0 0 0 0 0 0]		[0 0 0 0 0 0 0 0 0]	
5	[0 0 0 0 0 0 0 0 0]		[0 0 0 0 0 0 0 0 0]	
9	[0 0 0 0 0 0 0 0 0]		[0 0 0 0 0 0 0 0 0]	
1	[0 0 0 0 0 0 0 0 0]		[0 0.6 0 0 0.6 0 0.6 0.6 0.6]	
6	[0 1 0 0 0 0 0 0 0]	*downward*	[0 1 0 0 0 0 0 0 0]	*downward*
9	[0 0 0 0 0 0 0 0 0]		[0 0 0 0 0 0 0 0 0]	

1	[0 0 0 0 0 0 0 0 0 0]		[0 0.6 0 0 0.6 0 0.6 0.6 0.6]	
9	[0 0 0 0 0 0 0 0 0 0]		[0 0 0 0 0 0 0 0 0 0]	
1	[0 0 0 0 0 0 0 0 0 0]		[0 0.6 0 0 0.6 0 0.6 0.6 0.6]	
4	[0 0 0 0 0 0 0 0 1 0]	*following*	[0 0 0 0 0 0 0 0 1 0]	*following*
7	[0 0 0 0 0 0 0 0 0 0]		[0 0 0 0 0 0 0 0 0 0]	
9	[0 0 0 0 0 0 0 0 0 0]		[0 0 0 0 0 0 0 0 0 0]	
1	[0 0 0 0 0 0 0 0 0 0]		[0 0.6 0 0 0.6 0 0.6 0.6 0.6]	
9	[0 0 0 0 0 0 0 0 0 0]		[0 0 0 0 0 0 0 0 0 0]	
1	[0 0 0 0 0 0 0 0 0 0]		[0 0.6 0 0 0.6 0 0.6 0.6 0.6]	
5	[0 0 0 0 1 0 0 0 0 0]		[0 0 0 0 1 0 0 0 0 0]	
1	[0 0 0 0 0 0 0 0 0 0]		[0 0 0 0 0 0 0 0 0 0]	
9	[0 0 0 0 0 0 0 0 0 0]		[0 0 0 0 0 0 0 0 0 0]	
1	[0 0 0 0 0 0 0 0 0 0]		[0 0.6 0 0 0.6 0 0.6 0.6 0.6]	
6	[0 1 0 0 0 0 0 0 0 0]	*downward*	[0 1 0 0 0 0 0 0 0 0]	*downward*
1	[0 0 0 0 0 0 0 0 0 0]		[0 0 0 0 0 0 0 0 0 0]	
9	[0 0 0 0 0 0 0 0 0 0]		[0 0 0 0 0 0 0 0 0 0]	
1	[0 0 0 0 0 0 0 0 0 0]		[0 0.6 0 0 0.6 0 0.6 0.6 0.6]	
4	[0 0 0 0 0 0 0 0 1 0]	*following*	[0 0 0 0 0 0 0 0 1 0]	*following*
7	[0 0 0 0 0 0 0 0 0 0]		[0 0 0 0 0 0 0 0 0 0]	
1	[0 0 0 0 0 0 0 0 0 0]		[0 0 0 0 0 0 0 0 0 0]	
9	[0 0 0 0 0 0 0 0 0 0]		[0 0 0 0 0 0 0 0 0 0]	
4	[0 0 0 0 0 0 0 0 0 0]		[0 0 0.6 0 0 0 0 0 0 0]	
7	[0 0 1 0 0 0 0 0 0 0]	*training*	[0 0 1 0 0 0 0 0 0 0]	*training*
9	[0 0 0 0 0 0 0 0 0 0]		[0 0 0 0 0 0 0 0 0 0]	
7	[0 0 0 0 0 0 0 0 0 0]		[0 0 0 0.6 0 0 0 0 0 0]	

4	[0 0 0 1 0 0 0 0 0]	*turnleft*	[0 0 0 1 0 0 0 0 0]	*turnleft*
9	[0 0 0 0 0 0 0 0 0]		[0 0 0 0 0 0 0 0 0]	
2	[0 0 0 0 0 0 0 0 0]		[0.6 0 0 0 0 0 0 0 0]	
5	[1 0 0 0 0 0 0 0 0]	*upward*	[1 0 0 0 0 0 0 0 0]	*upward*
7	[0 0 0 0 0 0 0 0 0]		[0 0 0 0 0 0 0 0 0]	
9	[0 0 0 0 0 0 0 0 0]		[0 0 0 0 0 0 0 0 0]	
1	[0 0 0 0 0 0 0 0 0]		[0 0.6 0 0 0.6 0 0.6 0.6 0.6]	

Table 4.10 Testing performance

	All Testing samples	motion samples	Undefined motion samples
# of test samples	60	10	50
# of misclassified	0	0	0
Success Rate	100%	100%	100%

The two experiments including feed-forward and recurrent type of gestures classifier shows that good performance in hand gesture recognition task. Test sequence consists of lots of undefined gesture samples need to design training samples additionally for better performance in feed-forward type of gesture classifier. However, it is easy to design gesture contribution tree in recurrent classifier. Feed-forward classifier must have fixed-three input samples for motion classification. However, recurrent classifier doesn't limit input sequence length for motion classification task. In other word, recurrent classifier is more extendable for sequence classification task because there is no limitation about the length sequence of a hand gesture. The performance between feed-forward and recurrent motion classifier is compared shown in Table 4.10.

Table 4.11 Comparison of Performance

	Size of neural network	Complexity of Training	Motion event Extendable	Test error
Feed-forward Classifier	Bigger	Fast	Bad	Good
Recurrent Classifier	Smaller	Slow	Batter	Good

4.4 System Configuration and operation experiments

This thesis developed an experimental human interface system using the proposed had gesture recognition system. The current system uses PC executes on Matlab r2006a and Humanoid Vision robot for image extraction, hand posture state encoder and hand gesture recognition. The operation experiment structure is shown at Fig.4.2, firstly the image sequence of hand gestures is continuously captured by Human Vision robot CCD, and then PC will accept the image sequence then extract these images sequence to the task of hand posture data and processes this data by the proposed hand gesture recognition system. The hand gesture recognition system will recognize the input hand gesture and this hand gesture were the command that will be execute by the Human Vision robot. The description for PC and Human Vision robot is describe at following section.

4.4.1 Humanoid Vision Description

The HVS is built with two cameras and five motors to emulate human eyeballs as shown in Fig. 4.3. These five motors, FAULHABER DC−servomotors, are used to drive the two cameras to implement the eye movement, one for the conjugate tilt of two eyes, two for the pan of two eyes, and two for the pan and tilt of the neck correspondingly. The control of DC−servomotors is executed by the motion control card, MCDC 3006 S, in a positioning resolution of 0.18°. With these 5 degrees of freedom, the HVS would track the target whose position is determined from the image processing of the two cameras. In addition, these two cameras, QuickCam^{TM} Communicate Deluxe, have specifications listed below.

- 1.3-megapixel sensor with RightLight™2 Technology
- Built-in microphone with RightSound™ Technology
- Video capture: Up to 1280 x 1024 pixels (HD quality) (HD Video 960 x 720 pixels)
- Frame rate: Up to 30 frames per second
- Still image capture: 5 megapixels (with software enhancement)
- USB 2.0 certified
- Optics: Manual focus

In this proposed system structure, the control and image process are both implemented in personal computer with 3.62 GHz CPU.

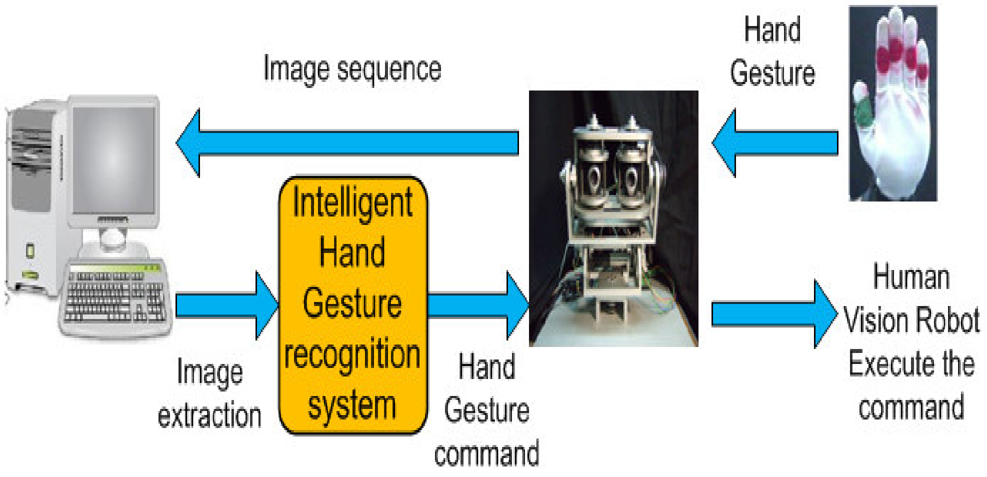


Image sequence
Hand Gesture
Intelligent Hand Gesture recognition system
Image extraction
Hand Gesture command
Human Vision Robot Execute the command

Fig. 4.2 operation experiments structure

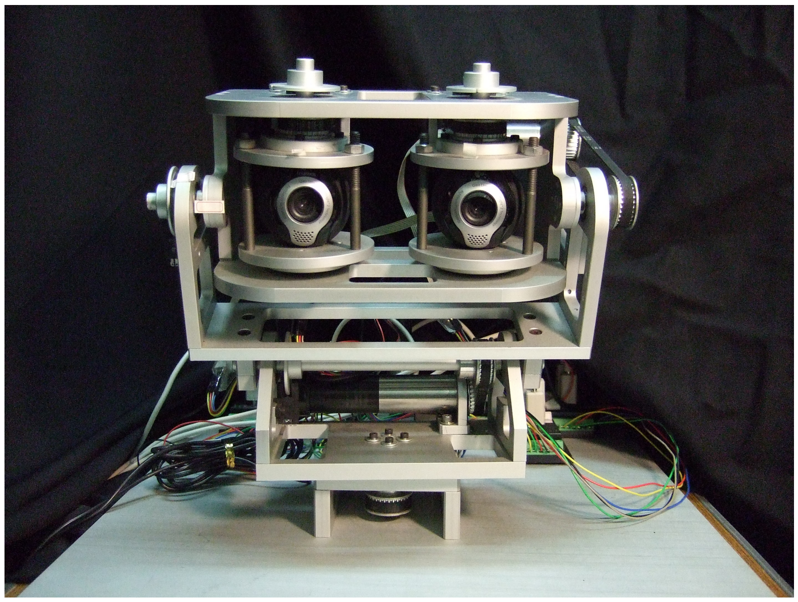


Fig. 4.3 Humanoid vision system

Chapter 5

Conclusion and future work

In this thesis, we provide an intelligent hand gesture recognition system which contains Image Data Retriever, Posture Data Generator, Posture States Encoder and Hand Gesture Analyzer. Image sequence of posture data can effectively encode to hand posture state by Posture States Encoder. Hand Gesture Analyzer contains two stages, Repeated State Retriever and Gesture Classifier, there are two kinds of Classifier are used in this thesis, feed-forward classifier and recurrent classifier. Based on trained gesture classifier, Hand Gesture Analyzer is implemented to classify different hand gestures. Each subsystem is serial cascaded and implemented by neural network. The experimental results show that this structure can achieve satisfactory real-time performance and high classification accuracy. The proposed intelligent hand gesture recognition system contained several advantages described below:

- Another application of hand gesture can be implemented by increasing hand posture states
- There is no limit of length of state sequence for a recognized hand gesture (trigger net).
- This system can't be influenced by undefined hand gesture (recurrent classifier).

A system has been developed to demonstrate the proposed approach the proposed approach. The experimental result shows that the system can correctly recognize the hand gestures in real time. The image sequences are processed in real time: at 8fps for the hand detection and 4fps for color tracking and hand posture recognition.

In the future, addition of new gesture through the “learning” hand gesture will make the system is more intelligent and flexible. This self learning behavior makes the system much more closely to human brain behavior. However there are a lot of conditions and restrictions need to be cleared to realize this concept. To solve this problem is the main goal in the future.

References

[1] Y. Wu and T. S. Huang, "Hand modeling analysis and recognition for vision-based human computer interaction," *IEEE Signal Process Mag.—Special Issue on Immersive Interactive Technology*, vol. 18, no. 3, pp. 51–60, May 2001.

[2] N. Badler, "Temporal Scene Analysis - Conceptual Descriptions of Object Movements," Report TR 80, 1975.

[3] R. Brooks, "Symbolic reasoning among 3D models and 2D images," *Artificial Intelligence*, pp, 285-348, 1981.

[4] A. Cohn, D. Magee, A. Galata, D. Hogg, S. Hazarika, "Towards an Architecture for Cognitive Vision using Qualitative Spatio-Temporal Representations and Abduction," *Spatial Cognition III*, Springer, 2003.

[5] M. Erwig and M. Schneider, "Spatio-Temporal Predicates," *IEEE Trans. on Knowledge and Data Engineering* (TKDE), vol. 14, no. 4, pp. 1–42, 2002.

[6] P. Muller, " A qualitative theory of motion based on spatiotemporal primitives," In A.G. Cohn, L.K. Schubert, and S.C. Shapiro, editors, Principles of Knowledge Representation and Reasoning: *Proceedings of the Sixth International Conference* (KR'98). Morgan Kaufmann, 1998.

[7] P. Sebastiani, M. Ramoni, P. Cohen, "Sequence Learning via Bayesian Clustering by Dynamics, Sequence Learning: Paradigms, Algorithms, and Applications," Springer, 2000.

[8] M. Mohnhaupt, B. Neumann, "Understanding Object Motion: Recognition, Learning and Spatiotemporal Reasoning, Toward Learning Robots," MIT Press , 65-92, 1993.

[9] A. Galata, A.D.M Cohn, D. Hogg, "Learning Temporal and Qualitative Spatial Components of an Interaction Model," Proc. *ECCV Workshop on Vision and*

Modelling of Dynamic Scenes (VAMODS), 2002.

[10] K.S. Fu, “Syntactic Pattern Recognition and Applications”, Prentice Hall, 1982.

[11] A. Tijsseling, and L. Berthouze, “A neural network for temporal sequential information.” *Proceedings of the 8th International Conference on Neural Information Processing*, Shanghai (China), pp. 1449–1454, 2001.

[12] J.L.Elman, “Finding structure in time,” *Cognitive Science.*, vol. 14, pp.179–211, 1990.

[13] R.J. Williams and D. Zipser, “A learning algorithm for continually running fully recurrent neural networks,” *Neural Computa.*, vol. 1, pp.270–280, 1989.

[14] S.E. Fahlman, “The recurrent cascade-correlation architecture,” Carnegie Mellon Univ., Pittsburgh, PA, Tech. Rep. CMU-CS-91-100, 1991.

[15] W.C. Omlin, K.K. Thornber, and C.L Giles," Fuzzy Finite-State Automata Can Be Deterministically Encoded into Recurrent Neural Networks", *IEEE Transactions on Fuzzy systems*, Vol. 6, no. 1, FEBRUARY 1998.

[16] C.L. Giles, S. Lawrence and A.C. Tsoi, “Noisy Time Series Prediction using a Recurrent Neural Network and Grammatical Inference”, *Machine Learning*. 2000.

[17] C.W. Omlin and C.L.Giles, “Extraction of Rules from Discrete-Time Recurrent Neural Networks”, *Neural Networks*, vol. 9, no. 1, pp. 41-52, 1996.

[18] S.E. Fahlman and C. Lebiere, “The cascade-correlation learning architecture,” *in Advances in Neural Information Processing Systems*, D. S. Touretzky, Ed. San Mateo, CA: Morgan Kaufmann, vol. 2, 1990, pp. 524–532.

[19] C.L. Giles, D. Chen, G.Z. Sun, H.H. Chen, Y.C. Lee, and M.W. Goudreau, “Constructive learning of recurrent neural networks: Limitations of recurrent casade correlation and a simple solution,” *IEEE Trans. Neural Networks*, vol. 6, pp. 829–836, 1995.

[20] L.B. Almeida, “A learning rule for asynchronous perceptrions with feedback in a combinatorial environment,” *in Proc. IEEE 1st Annu. Int. Conf. Neural Networks*, M. Caudil and C. Butler, Eds. New York: IEEE Press, pp. 609–618, 1987.

[21] L. Atlas et al., “A performance comparison of trained multilayer perceptrons and trained classification trees,” *Proc. IEEE*, vol. 78, pp. 1614–1619, 1992.

[22] A. Sperduti, A. Starita, and C. Goller, “Learning distributed representations for the classification of terms,” *in Proc. Int. Joint Conf. Artificial Intell.*, pp. 509–515, 1995.

[23] A. Sperduti, A. Starita, and C. Goller, “Fixed length representation of terms in hybrid reasoning systems, report i: Classification of ground terms,” Dipartimento di Informatica, Universit`a di Pisa, Italy, Tech. Rep. TR-19/94, 1994.

[24] A. Sperduti and A. Starita, “An example of neural code: Neural trees implemented by LRAAM’s,” *in Proc. Int. Conf. Neural Networks Genetic Algorithms*, Innsbruck, Austria, pp. 33–39, 1993.

[25] A. Sperduti, “Encoding of labeled graphs by labeling RAAM,” *in Advances in Neural Information Processing Systems*, J. D. Cowan, G. Tesauro, and J. Alspector, Eds. San Mateo, CA: Morgan Kaufmann.

[26] J.A. Sirat and J.P. Nadal, “Neural trees: A new tool for classification,” *Network*, vol. 1, pp. 423–438, 1990.

[27] A. Sankar and R. Mammone, “Neural tree networks,” *Neural Networks: Theory and Applications*. New York: Academic, pp. 281–302, 1991.

[28] J. B. Pollack, “Recursive distributed representations,” *Artificial Intell.*, vol. 46, nos. 1–2, pp. 77–106, 1990.

[29] S. Muggleton and L. De Raedt, “Inductive logic programming: Theory and methods,” *J. Logic Programming*, vol. 19, no. 20, pp. 629–679, 1994.

[30] Y. Hochreiter, Bengio, P. Frasconi, and J. Schmidhuber, “Gradient flow in

recurrent nets: The difficulty of learning long-term dependencies." In S. C. Kremer and J. F. Kolen,editors, *A Field Guide to Dynamical Recurrent Neural Networks. IEEE Press*, 2001.

[31] Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[32] Pearlmutter, "Gradient calculations for dynamic recurrent neural networks: A survey," *IEEE Transactions on Neural Networks*, vol. 6, no. 5, pp. 1212–1228, 1995.

[33] J. Williams and J. Peng, "An efficient gradient-based algorithm for on-line training of recurrent network trajectories" , *Neural Computation*, vol. 2, no. 4, pp. 490–501, 1990.

[34] C.H. Lan , "Design of Intelligent Motion Description System", 2008

[35] S. Wagner, B. Alefs, C. Picus, "Framework for a portable gesture interface", *Automatic Face and Gesture Recognition, 2006. FGR 2006. 7th International* Conference , pp. 275 – 280, 2006

[36] Q. Chen, N.D Georganas, . and E.M. Petriu, "Hand Gesture Recognition Using Haar-Like Features and a Stochastic Context-Free Grammar", *Instrumentation and Measurement, IEEE Transaction*, Volume 57, Issue 8, pp.1562 – 1571, 2008

[37] M. Vafadar, A. Behrad, "Human hand gesture recognition using spatio-temporal volumes for human-computer interaction", *Telecommunications, 2008. IST 2008. International Symposium*, pp.713 – 718, 2008