INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

A Programmable Display System for Perceptual Stability Research

Fang Wang

Department of Electrical and Computer Engineering

McGill University, Montréal, Canada

March 2001

A thesis submitted to the Faculty of Graduate Studies and Research

in partial fulfillment of the requirements of the degree of

Master of Engineering

To Shaomei

Abstract

A programmable display system for perceptual stability research is presented. The display system involves 96 programmable, independent, two-colored LED outputs, and can display a video with the time resolution of one millisecond. Because of the high time resolution, this system provides a means to study humans' perceptual stability with high precision. The system may display up to 24,576 frames repeatedly after each download. Hardware for video download, saving, and display, and software for video design were developed and implemented. The hardware consists of a control module, a storage and interface module, and a display module. The software offers a graphical user interface (GUI). Using this display system, we carried out two preliminary experiments to test the saccade target theory of how humans perceive stability during eye movements.

Résumé

L'objectif du présent mémoire est de développer un système d'affichage programmable afin d'effectuer des expériences dans le cadre d'un projet de recherche en stabilité perceptuelle. Le système d'affichage consiste en 96 diodes électroluminescentes (LED) dotées de deux couleurs qui sont programmables et indépendantes. Ce système peut afficher une image vidéo avec une résolution-temps d'un millième de seconde. La haute résolution-temps permet au système d'étudier la stabilité de la perception visuelle chez l'humain avec une grande précision. En effet, le système est capable d'afficher jusqu'à 24 576 images après chaque telechargement, et ce, à plusieurs reprises. Nous avons développé le matériel électronique (hardware) pour le telechargement, la sauvegarde et l'affichage d'images vidéo. Aussi, nous avons implémenté des logiciels pour la planification d'images vidéo. Le matériel électronique consiste en quatre modules. Un module de contrôle, un module de mise en mémoire (stockage), une interface et un module d'affichage. Le logiciel offre une interface graphique (GUI). Nous avons utilisé le système d'affichage que nous avons développé dans deux expériences préliminaires afin de tester la théorie de "saccade target" concernant la stabilité perceptuelle durant les mouvements de l'œil.

Acknowledgements

First of all, I would like to thank my thesis supervisor, Professor James Clark, for his advice and encouragement throughout my thesis research and studies.

I thank Ziad Hafed, my colleague in the Visual-Motor Systems Laboratory, for the valuable discussions and his technical supports on the application of serial communication interfaces.

I thank Henry Chan for his help on HC12 evaluation board setup.

Thank goes to Fatima Drissi-Smaili for her translation of the thesis abstract into French.

Thanks are also due to Yuan Zhang, Jianfeng Yin, Muhua Li, Sandra Skaff, Jie Yao, Xiaohui Song, and Zhan Wang, for their patience and cooperations in my experiments.

Finally, I would like to express my deepest gratitude to my family whose support was of paramount importance in completing this thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis describes a programmable display system for perceptual stability research. The background of perceptual stability research, the implementation of the display system, and two experiments by the system follow this chapter. Conclusions are drawn at the end of the thesis.

1.1 Motivation and solution

The problem of humans' perceptual stability has interested researchers for many decades. When an object moves in the visual world, the light pattern on our retinas moves and we perceive motion. On the other hand, if the visual world keeps stable, but our eyes move, the light pattern on our retinas also moves. However, in this case, we perceive stability instead of motion. That is, our eyes receive the same signal but generate different perceptions.

Although there are many popular theories in this area, such as *reafference theory, translation theory, calibration theory*, and *saccade target theory*, they are defective or incomplete. The answer of how we perceive stability during eye movements is not

clear yet and further research is needed.

We designed a programmable display system for the research. Using this system, we can generate a sequence of frames and show it to a subject. Then we may test the subject's perception of a specific visual stimulus. The display panel is a *10 × 10 light-emitting diode* (LED) array with four empty corners. Each LED outputs two colors, green and orange. Our system can display up to *24,576 96*-pixel, two-colored frames repeatedly after each download. The time resolution of this system is *1ms*. This high resolution is very helpful in perceptual stability research. We can generate visual stimulus in high precision and do millisecond-level analysis on humans' perception stability. For example, in the first experiment described in Chapter 5, we changed the flashing period of a group of LEDs from *11ms* to *100ms* with a step of *1ms*, and observed a slope in the *Flashing Period - Visual Perception* curve. The curve tested a hypothesis on how humans perceive stability during eye movements. Without the high resolution, these results were not possible.

We did two preliminary experiments by this display system. These two experiments were designed to test a hypothesis on the detailed steps during humans' eye movements. We suppose that the visual system computes two templates based on the scene in the period of computation before and after each saccade and compares them. If they corresponds to each other, the visual system generates stable perception, otherwise, it generates unstable perception. Our results support the existence of the two computation periods.

1.2 System design

Because we need a high time resolution in display, most popular display methods are not suitable. Phosphor persistence and their display principles confine them in low-frequency (lower than *100Hz*) applications. We therefore used *light-emitting diodes*

(LEDs) as our display media, which can be turned on and off immediately and suitable for high frequency display. We used *96* two-colored LEDs to compose a *display module*. We implemented a *control module*, which used a *MC68HC912B32* microcontroller as the kernel, and a *storage and interface module*, which included a memory and interface chips. These three modules composed the display hardware. Software was also developed. We designed a *graphical user interface* (GUI) using *Microsoft Visual Basic 6.0* to help the users edit a frame sequence and arrange the design to a database. Several format converting programs were developed to convert the database to a format readable to the display hardware. After downloading the file to the hardware, a researcher may show the frame sequence to a subject.

Before the experiments described in Chapter 5, we prepared tens of frame sequences using the software. Then we downloaded the sequences and showed them one by one to the subjects. We recorded the subjects' perception response and analyzed the data.

1.3 Thesis overview

This thesis describes a programmable display system for perceptual stability research.

Chapter 2 introduces the background of the perceptual stability research. The physiological and psychological backgrounds of the area are discussed briefly. Different theories of perceptual stability in this area are explained, compared and evaluated. Specifically, the *saccade target theory* is presented in detail.

Chapter 3 includes the motivation and solution of the system. We compared and evaluated different popular displays in this chapter. Our system solution is described at the end.

Chapter 4 elaborates the detailed implementations of the system, including the display

hardware and the software. The features of the display system is summarized.

Chapter 5 presents two preliminary experiments using this display system. A hypothesis on the detailed steps during humans' eye movements is introduced at first, followed by the setups, data, and analysis of the two experiments.

Finally, Chapter 6 concludes the thesis with a discussion of the contribution of this work, as well as possible directions for future improvements and researches.

Chapter 2

Background

If a scene changes during a fixation, we can detect the difference, because the light pattern on our retinas changes. However, if the changes of the image on our retinas are due to our eye movements, or head and/or body movements, we do not perceive displacements - but we perceive stability. This perceptual stability has puzzled vision researchers for decades. Our system described in this thesis is designed to help researchers solve this problem. In this chapter, we introduce briefly the background of perceptual stability research, including different theories, as well as the physiological and psychological background of this area.

2.1 Perceptual stability

2.1.1 Introduction to perceptual stability

The problem of perceptual stability comes from an observation that seems contradictory. If our eyes are stationary, and the visual world moves, we perceive displacements of the scenes. On the other hand, if the visual world is stable, while our eyes

move, whether by saccades, head movements, or body movements, we do not perceive displacements - rather we perceive stability instead. However, in both cases, the projections of the scenes on our retinas change in the same way. That is, we have different perceptions for the same retinal inputs. Hence, there must be some mechanisms during human beings' eye movements that let us perceive stability.

Since our eyes are almost moving at all the time, perceptual stability is a very common phenomena for every human being in the normal life. Even during a fixation, our heads and bodies are not absolutely immobilized at normal postures [2]. Moreover, saccades, the fast eyeball movements, lead to position changes of our eyes. However, we do perceive stability with the constant eye movements. "How do humans retain stable visual perceptions during eye motions?" is still a question that the vision researchers are facing.

There are at least two kinds of perceptual stabilities. The first is *visual direction constancy* [2], which is defined as the stable perception which occurs when only the eyes move, and the head and the body are stable. The second form of perceptual stability is *visual position constancy* [2], which happens when not only the eyes, but also the head and/or the body change positions. Our system is designed for studying *visual direction constancy* primarily.

Two questions arise from consideration of *visual direction constancy.* The first is why the moving of the light patterns on our retinas is not perceived. The second is why the apparent light pattern differences on our retina between two continuous fixations do not bring out the perception of displacements. The first question is easy to answer. The eye velocity during a saccade can reach up to 1000 degrees per second [2]. This speed is too fast for the visual receptors of the retina to response to because of the relatively long time constants of the neurons in the visual pathways. Moreover, visual sensitivity reduces during and just before the occurrence of a saccade (*saccadic suppression*). Hence, even at the beginning and the end of a saccade, when the

speed of eye motion is not very fast, the mechanism of *saccadic suppression* prevents us perceiving the image sweeping on our retinas. However, on the other hand, the answer to the second question is still not very clear.

Three kinds of information possibilities play important roles in producing stable visual perceptions across saccadic eye movements. Current perceptual stability theories differ from each other by arguing the different roles and degrees that these information play in the phenomena of perceptual stability.

- *Retinal information:* the light patterns sensed by our retina, containing the aspects of the visual world before and after a saccade.
- *Neural outflow:* the output of our neural systems to the extraocular muscles and/or other visual-motor systems controlling the eyeballs. A command from our brain is sent to these muscles before each saccade. This command may involve the expected moving direction, distance, and velocity of a saccade.
- *Neural inflow:* This information is the feedback from the eyes to the neural system. The neural inflow provides to the neural system the position and the direction of the organism.

The *neural inflow* and the *neural outflow* together constitute the so-called *extraretinal signal.*

2.1.2 Theories of perceptual stability

There are many theories explaining the mechanisms of producing perceptual stability. Several popular theories are listed and evaluated below. The *cancellation theory* and the *translation theory* both assume an accurate *neural outflow* signal. The calibration theory emphasizes the role of the *neural inflow* signal.

- *Reafference theory* [17] The assumption of the *reafference theory* is that the human retina senses different images before and after each eye movement and that the *neural outflow* is used to "eliminate", "cancel out", "subtract" or "compensate for" the effects of eye movements. Thus, if the visual world is stable during the two fixations, the neural system may use the information contained in the *neural outflow* to maintain stable perception. Obviously, this solution needs precise *neural outflows.* The cancellation, elimination, or subtraction can work well only if the eye movement is predicted precisely enough. However, physiological evidence shows that this condition cannot been satisfied. The accuracy of the motor-visual system is much lower than that of the human retina.

- *Translation theory* The *translation theory* also depends on a precise prediction of an eye movement. Moreover, it assumes that there is a complete representation of the outside visual world when the scene is stable. This map is established before an eye movement and stored in a memory in the brain. Before or during the eye movement, the neural system calculates an expected retinal image based on the visual map established previously and on the predicted distance and direction of the eye movement. After the eye movement, it compares the new image and the expected image. If the outside world is stable during this process the two images should match each other, and thus the viewer will perceive stability. As with the *reafference theory*, the physiological evidence that the eye movement prediction is not accurate enough challenges this theory. Moreover, the existence of a detailed, complete "visual map" is doubted. Irwin concluded that transsaccadic memory is undetailed and with limited capacity. [11]

- *Calibration theory* Observing that the eye movement prediction is not precise enough, the *calibration theory* emphasizes the role of *neural inflow* in perceptual stability. Bridgeman proposed a *calibration theory* [2].

- *Intra-saccadic stimulus shift* The theory of *intra-saccadic stimulus shift* aims to explain the mechanisms that overcome the imprecision of the extraretinal

signals. The perceptual stability does not come from that we perceive stability, but rather from that we ignore the errors by a mechanism that computes it. If the *intra-saccadic stimulus shift* size is within the range of an uncertainty, the world appears stable [7].

In Section 2.3 we will introduce a newer perceptual stability theory, the *saccade target theory*, which combines the *intra-saccadic stimulus shift* theory and the *attention theory* that is introduced in the next section.

2.2 Attention theory

Traditionally, we believe that our eyes are working like cameras, converting the scenes in front of our eyes to images on our retinas and transmit these images to our brains, where the visual signals, containing all the visible information of the scenes, are analyzed in details. However, this viewpoint is challenged by current visual theories. Because the relatively poor optical characteristics of the human retina, such as the structures of eyeballs and the uneven receptor distribution on the retinas, are inconsistent with the extremely good visual ability of humans, some other mechanisms of our visual system must be involved in perceiving the outside world, and the basic question of *"how do we see?"* arises once again. The *attention theory* proposes that we **see** only the object at the position that we are **paying attention to**. Our eyes move to look at the place where something interests us [14] [15]. This theory is supported by many psychological and physiological experiments and accepted by more and more vision researchers. Undoubtedly the problem of *"how do we see?"* is linked tightly to the problem of *"why do we perceive stability when our eyes are moving?"*. In consequence, solutions to the eye movement problems would greatly help us understand the phenomena of perceptual stability.

This section introduces to the physiological backgrounds of eye movements and some

Anticipations	< 80 ms
Express saccade	90-130 ms
Fast regular	140-170 ms
Slow regular	190-230 ms
Afferent and efferent delays	about 25 ms

Table 2.1: Saccade Reaction Time of Humans *Source: Brain Research Unit, Germany. B. Fischer's web site (http://www.brain.uni-freiburg.de/fischer/intro.html).*

basic ideas of the *attention theory.*

2.2.1 Visual-motor systems

Saccades are very fast, ballistic eye movements, separated by fixation periods during which the eyes are relatively still [2]. Saccades play an important role in spatial attention [6] and bring out the problem of perceptual stability. They are controlled by the visual-motor systems, or the oculomotor systems, in our brains.

The main brain structures involved in the generation of saccades are the brainstem, the primary visual cortex, the prestriate cortex, the parietal cortex, and the frontal cortex. The saccade-related structures in the brainstem, including the superior colliculus, omnipause neurons, medium-lead burst neurons, and the neural integrator, constitute the major low-level "hardwares" for saccade generation and control [8].

The *saccadic reaction time*, the time between the presentation of a target and the start of a prosaccade, may have different values for different types of saccades. Typical values of *saccade reaction time* of humans are listed in Table 2.1. Figure 2.1 shows a histogram of the *saccadic reaction time* of humans. Therefore, mechanisms involved in humans' eye movements, have the time scales in the range of from tens of milliseconds to several hundreds of milliseconds.

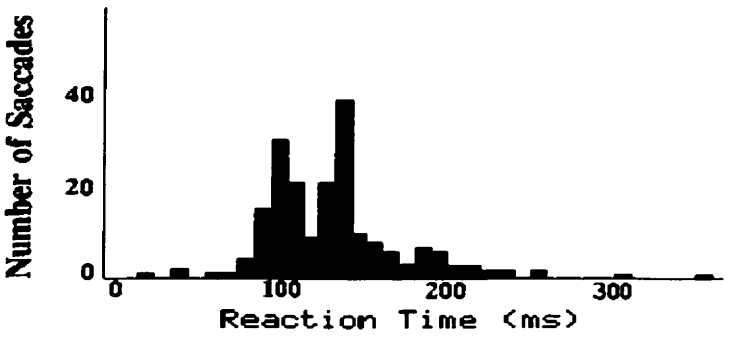

Figure 2.1: Distribution of Humans' Saccadic Reaction Time *Source: Brain Research Unit, Germany. B. Fischer's web site (http://www.brain.uni-freiburg.de/fischer/intro.html).*

2.2.2 Attention

Humans have very high visual abilities and obtain most of their information of the outside world through their visual systems. Our eyes can detect very fine structures and differences. On the other hand, compared with our visual ability, our eyes are apparently badly constructed visual apparatus [14]. Before reaching the photosensitive rods and cones, the light must first traverse a dense tangle of neural matter and a vast web of blood vessels. Other defects of the human retina include its severe nonuniformity and the "blind spot". Therefore, the traditional argument that the human brain gathers the whole image on the retina and "sees" the whole scene is doubted. O'Regan [14] gave an alternative explanation, that we only see the objects at the position we are paying attention to.

> *"The feeling of the presence and extreme richness of the visual world is, under this view, a kind of illusion, created by the immediate availability of the information in this external store."* [14]

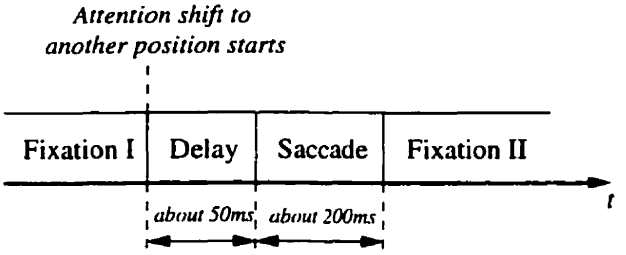

Figure 2.2: Attention, Saccade, and Fixation *During the first fixation, an attention shift is produced. After a short delay, about 50ms, a saccade is triggered, which takes about 200ms. The direction and the distance of the saccade are determined by new locus of attention attained during the first fixation period. A new fixation period starts after the saccade.*

Spatial attention, in biological vision systems, is a process by which the sensitivity of visual feature detectors tuned to a certain area of the visual field are enhanced relative to other areas [5]. Recent psychophysical experiments suggest that there exists a causal connection between spatial shifts in visual attention and the production of saccadic eye movements in humans [4]. The relations between attention, saccades, and visual perception are shown in Figure 2.2.

The winner-take-all model is used in many attention theories [6][18] to explain the strategy of triggering a saccade. The feature map of the peripheral information is computed and combined to a *saliency* map. The features are weighted by different values to produce the *saliencies*. A saliency over the threshold may trigger a saccade and brings the next fixation. Figure 2.3 shows a winner-take-all model in attention signal generation.

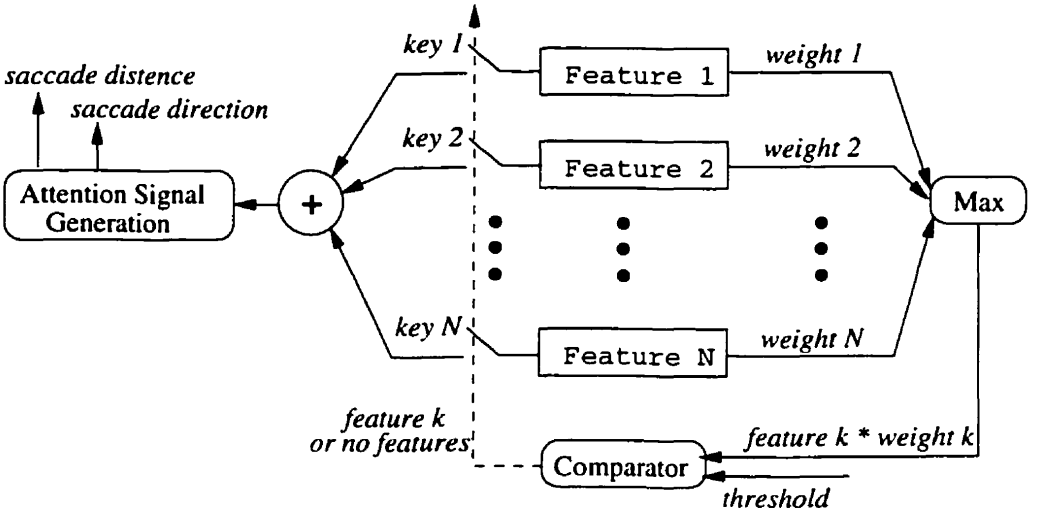

Figure 2.3: Winner-Take-All *The maximum saliency is selected. If the saliency is large enough to produce a new attention shift, compared to the threshold, an attention signal is generated. The direction and the distance of the new attention location depend on the feature that produces the shift in attention.*

2.3 Saccade target theory

Currie et al [7] proposed the *saccade target theory*, which is a newer theory to explain the phenomena of perceptual stability. It proposes that only the critical factors are used to coordinate retinal and perceptual space, and that the entire shifting retinal image is not stored. After a saccade, the eyes do a *"closer inspection"* [7]. This theory solves the "accuracy problem" of many other perceptual stability theories and considers the real processes which occur during the human saccade.

The *saccade target* is an object or a region in the scene that is selected by the human visual system before each saccade, i.e., "the feature that produces the next attention shift" in Section 2.2. Based on this theory, critical features of the target are extracted and then the eyes rotate to bring the target into central vision. During the new

fixation, these critical features are used to "locate the target objects within a limited retinal region". If the locating process succeeds, stability is perceived. Otherwise, the searching region is broadened and instability is perceived.

This theory combines the *intra-saccadic stimulus shift* theory and the *attention theory.*

Obviously, based on this theory, neither accurate *neural input and/or output* nor large memory containing a detailed, complete representation of the scene is needed. It overcomes the difficulties of the first three theories listed in Section 2.1.2. Moreover, it links the problem of perceptual stability to the *attention theory*, which proposes a new and reasonable strategy in human vision.

Currie et al [7] carried out an experiment to prove their *saccade target theory.* They tested their assumption by having subjects make saccades to objects in a scene. During the saccade, either the entire scene, just the target object, or just the background behind the target object was displaced. They found that displacements of the object alone were twice as detectable as the other two kinds of displacements [7].

Their experiment showed that the "critical features", the target objects in the experiment, play an important role in perceptual stability.

Chapter 3

System Design

We need a display system for perceptual stability research. The display system should not only let us observe perceptual stability phenomena, but also let us measure different parameters that may influence humans' perceptual stability. Because the different mechanisms involved in humans' eye movements, as we know so far, have the time scales in the range of from tens of milliseconds to several hundreds of milliseconds (ms), millisecond-level measurements are reasonable for our current research on perceptual stability. Hence, an ideal system should satisfy the following three basic demands: *1) it displays a video pattern frame by frame; 2) it allows us to display different video patterns we design; 3) it displays a video pattern with a timing resolution precise enough for millisecond-level measurements.*

Although there are many kinds of displays, most of them cannot satisfy all three demands. For example, a *cathode ray tube* (CRT) display system satisfies the second condition perfectly, but it does not truly display a video pattern frame by frame, and it cannot produce a *1kHz* image frame sequence for precise analysis for perceptual stability research.

However, *light-emitting diodes* (LEDs) can satisfy all these conditions. Hence, we de-

veloped a *10* × *10* two-color-LED array as our display media. Moreover, we designed hardware and software to complete the system. The software is used to design video patterns for display. And the hardware, with a *MC68HC912B32* microcontroller, controls the LED array based on the designed data. Data representing video patterns and control commands are transmitted from the host computer that runs the software to the hardware through serial communication interfaces.

3.1 Overview of displays

Although there are many kinds of displays, including *cathode ray tube* (CRT) monitors, *liquid crystal displays* (LCDs), *electroluminescent displays* (ELDs), *plasma display panels* (PDPs), and so on, most of them cannot satisfy all three of our demands.

The drawbacks of such displays mainly come from their display control mechanisms and phosphor persistence.

3.1.1 Display by a CRT monitor

CRT monitors are the most frequently used programmable display. However, unfortunately, a monitor does not really display a video "frame by frame" and cannot be used in our system.

The display of a CRT monitor is relatively independent of the computer that produces the display data. A computer sends digital signals of data to the *super video graphics array* (SVGA) adapter. The SVGA adapter then sends the signals through the *digital to analog converter* (DAC) circuit. The DAC compares these values and assigns voltage levels for red, green, and blue. This RGB (Red, Green, and Blue) model uses the three primary colors to create the color of a single pixel. The DAC sends the signals to the *cathode ray tube* (CRT).

A *cathode ray tube* (CRT), is a vacuum tube, which produces an image when an electron beam strikes the phosphorescent surface inside the monitor. An electron gun generates a beam of electrons in each primary color to create an overlapping image. The spot on the screen races from right to left and from top to bottom in a sequence of lines called *raster scanning.* After the entire screen has been swept by the *raster scanning*, the beams return to the upper left-hand corner to begin again. The screen of a monitor is usually redrawn, or refreshed, 60 or 50 times per second.

It is impossible to set the display frequency of a monitor up to *1kHz* to satisfy our measuring demands. And the phosphor persistence (See Section 3.1.3) of CRTs limits their usage in low time resolution applications only.

3.1.2 Flat-panel displays

The principal types of flat-panel displays are *liquid crystal displays* (LCDs), *electroluminescent displays* (ELDs), and *plasma display panels* (PDPs). Each of these displays has its own limitations in our perceptual stability research.

A LCD does not generate any light of its own, but creates an image by controlling whether light passes through it or not. Directions of the liquid crystal molecule are controlled by the voltage applied to the device. When the directions change, the light path through the device is cut off or connected. However, the response speed of LCDs is very slow. For example, the response time of a "fast response time" black-and-white LCD, *Sharp LM4Q30TA*, is *100ms* [21]. Therefore, LCDs do not support *1ms* resolution measurement.

Plasma displays work by sandwiching a neon/xenon gas mixture between two sealed glass plates with parallel electrodes deposited on their surfaces. When a voltage pulse passes between two electrodes, the gas breaks down and produces weakly ionized plasma, which emits ultraviolet radiation. The ultraviolet radiation activates color

phosphors and visible light is emitted from each pixel. Because phosphor persistence exists in the application of plasma display, they are not suitable to our system.

EL displays use thin-film phosphors which emit light when subjected to an electrical field. The advantages of EL displays include their rugged construction, ability to operate in an extended temperature range, and low power consumption. Because EL devices operate at high voltages [3], they are not as popular as LCDs and plasma displays.

3.1.3 Phosphor persistence

Phosphor persistence is the property of a phosphor that determines its ability to emit light for a time after the stimulus has been extinguished. Persistence may extend to over a minute. Phosphor persistence will therefore limit the display frequency of the device.

A CRT pixel is illuminated when an electron beam strikes its phosphor coating. Phosphor persistence of monitors can be detected at or beyond *12ms* [7]. Plasma and EL displays also use phosphor material generate light, and cannot display frames at *1kHz* to satisfy the requirements for our research.

3.2 Two-color-LED array

Since most displays are not suitable for our perceptual stability research, we used *light-emitting diodes* (LEDs) as our display devices. LEDs are fast switching, and may offer different colors. Using an array of these devices, we can realize a system displaying independent frames in each *1ms* time slot. Moreover, because the LEDs we used are two-colored, we can test subjects' responses to different colors.

3.2.1 Light-emitting diode

A *light-emitting diode* (LED) is a type of diode that emits light when current passes through it. Depending on the material used, the color of the emitted light can be different.

The operation of a LED is simple. Applying forward bias to a LED will "turn on" it, while applying backward bias to a LED will "turn off" it.

Because the light emitted from a LED depends only on the current passing through the semiconductor, a LED is a fast-switching device. For example, an optical-isolator, whose input part is a LED, may work with rates up to *2Gbits/s* [16].

Moreover, LEDs have good contrast and brightness. When a LED is turned off, no light is given off. And a typical value of illumination output of a LED with a diameter of *5mm* is *15mcd* [22].

3.2.2 The two-color LED panel in our system

We arranged *96 Sharp LN11WP38* two-color LEDs in a 10×10 (no units at each corner) array as our display (Figure 4.1). The LED is round and has a diameter of *5mm*. Each LED has a common cathode and two separate anodes (Figure 4.5), which controls two basic outputs, orange output and green output, with *630nm* and *565nm* of peak wavelength, and *40nm* and *30nm* of wavelength range, respectively. Because the two anodes are physically independent, a LED may outputs two color together and produce a third output color. For more information about the *Sharp LN11WP38* LED, refer to *LN11WP38 Data Sheet*[22].

3.3 A system solution

We designed hardware that can display pre-designed image frames by the LED panel in each *1ms* interval. The pre-designed data is downloaded to the hardware and saved into a *4Mbit static random accessible memory* (SRAM). A microcontroller, *MC68HC912B32*, is used to communicate with the host computer, on which display data is designed, to arrange the data in the SRAM, and to control interface chips to produce suitable outputs of the LED panel in each *1ms* interval or frame. We define each set of *1024* contiguous frames as a *pattern*. Our system may test up to *16 patterns* one by one cyclicly after each downloading. And each pattern can be repeated a number of times. The user may set the number of the repeating cycles of each pattern and download them to the board. After each hardware reset, a user may choose download and then display or display the data in memory directly.

Software for designing the display patterns has been designed and runs at a host computer. This software is described in Section 4.2.

Serial communication is used for connecting the host computer and the display hardware.

3.3.1 Display hardware

The display hardware consists of three modules, the *controlling module*, the *storage and interface module*, and the *display module*. The pre-designed frames, up to *16* $\times$ *1024*, are downloaded from a host computer and saved into a *512-Kword* $\times$ *8-bit* SRAM, *HM628512B*. During display, the microcontroller reads corresponding data out of the SRAM, and writes the data to the interface circuits, which includes *Intel 8255* peripheral interface chips and buffer chips for driving the *96* LED units.

To simplify the design, we used a *MC68HC912B32* evaluation board as our *controlling*

module. Because the evaluation board does not support expanded bus mode, all of the lines of the *8255* interface chips and the memory chip, including data lines as well as address lines, are controlled by the microcontroller's I/O ports. We used the microcontroller's timer interrupt to produce each *1ms* time slot. Another role of the *controlling module* is to communicate with a host computer through a serial communication port.

The *storage and interface module* is a circuit board including a SRAM chip, eight *8255* interface chips, twenty-six *74LS244* octal buffer chips, and some logic and transceiver chips. This module supplies sufficient current to drive each LED unit.

The *display module* contains *96* two-color LEDs. This module has *192* controlling connections and one common ground connection to the *storage and interface module.*

The three modules are connected together by ribbon cables. A picture of the display hardware is shown in Figure 3.1. Figure 3.2 shows the top view of the hardware, in which the right part is the *control module* and the left part is the *display module.* A picture of the *interface and storage module* is shown in Figure 3.3.

3.3.2 Software for designing the patterns

The pattern design software is a bridge between a user's idea and its corresponding raw data for the hardware.

We designed a *graphical user interface* (GUI), running under *Windows*, for pattern entry using *Microsoft Visual Basic 6.0.* A user can design all possible patterns using this GUI. Moreover, this GUI offers a design strategy suitable for our perceptual stability research.

The software completes the following main tasks: *1) designing a pattern; 2) grouping different patterns and defining corresponding repeating cycles; 3) downloading the data*

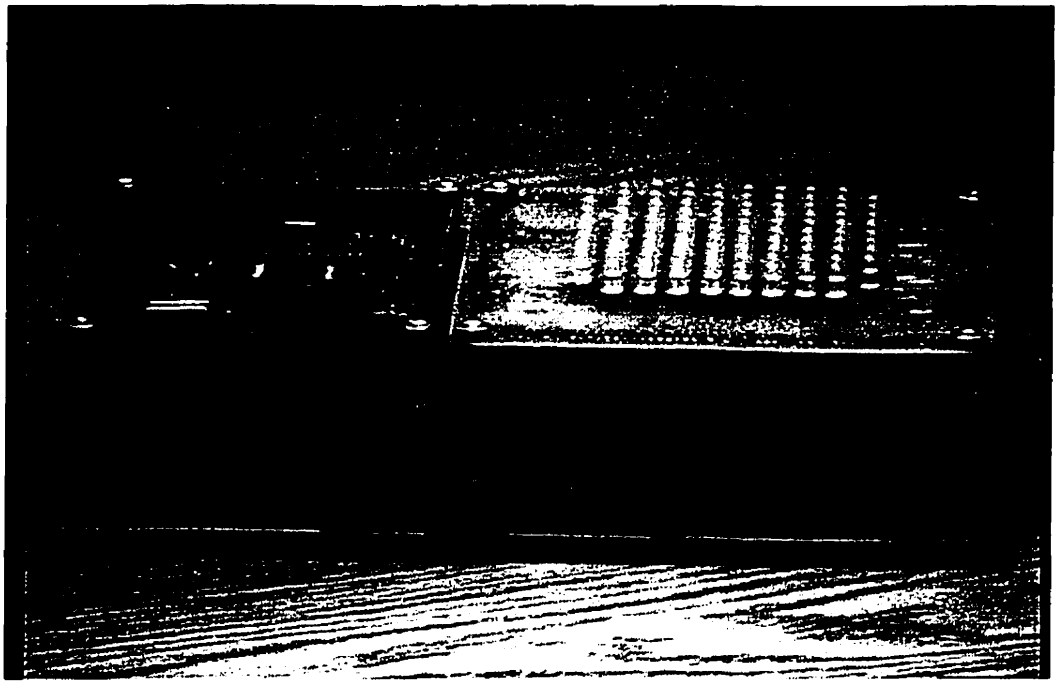

Figure 3.1: The Display Hardware

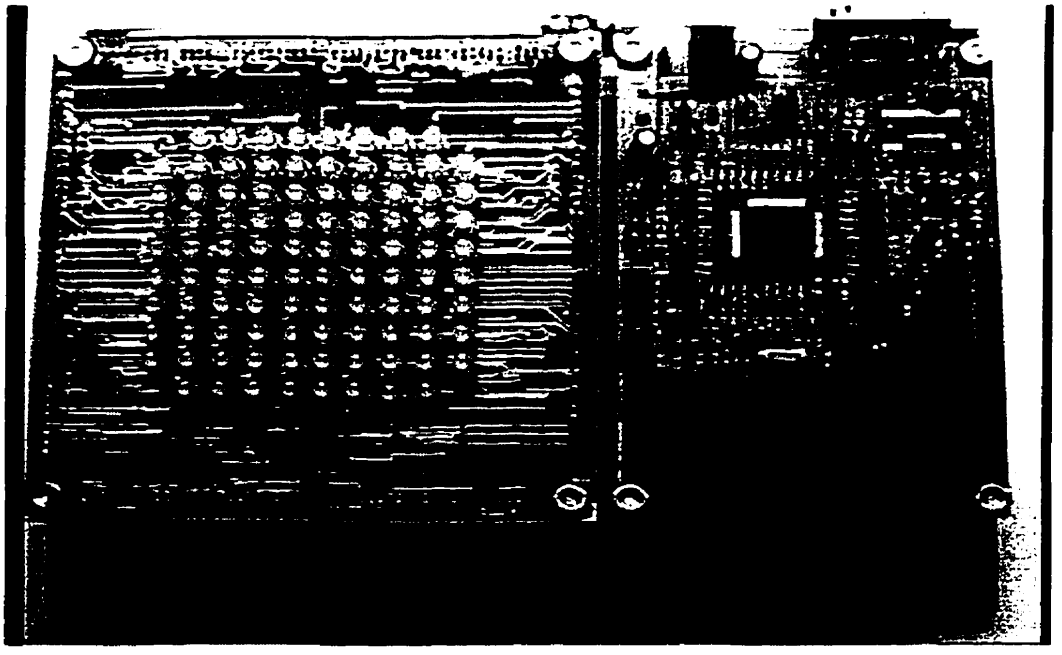

Figure 3.2: Top View of the Display Hardware

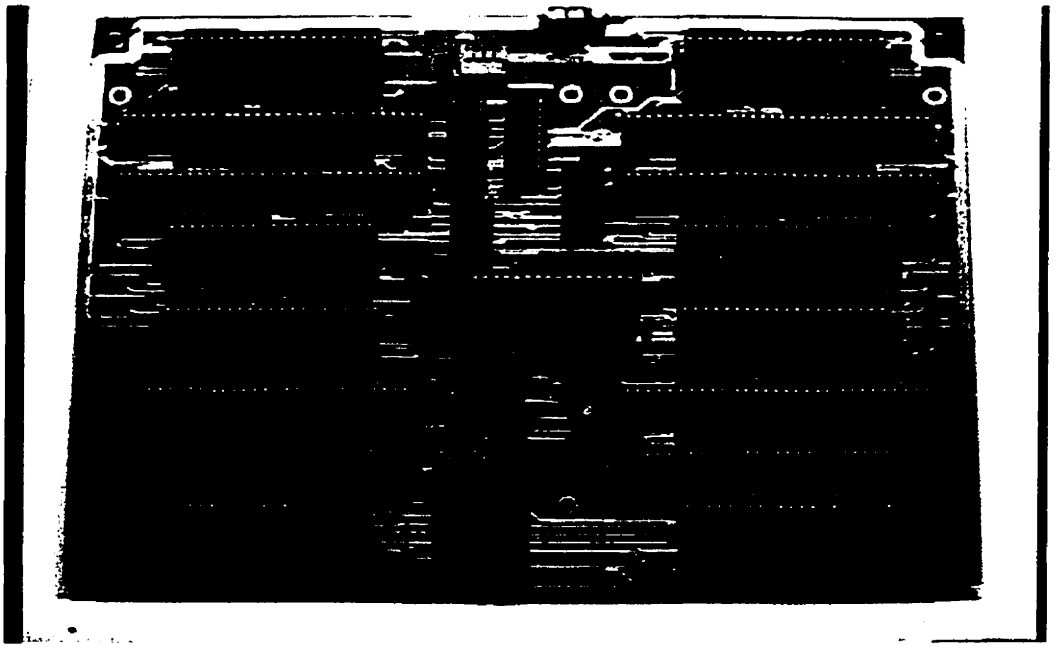

Figure 3.3: The Storage and Interface Module

file to the display hardware through a serial communication port.

A user's design is entered into an *Microsoft Access* database first. Then the database file is converted to a text file for download.

A counterpart running under *UNIX*, programmed with *C* language, was also developed.

Chapter 4

System Implementation

This chapter includes the specific implementation of our system. Our system is composed of two main parts, hardware for display control and software for pattern design, which are described in Section 4.1 and Section 4.2, respectively. Features of our whole system are summarized in Section 4.3.

4.1 Hardware design

The hardware of our design consists of three modules, the *controlling module*, the *storage and interface module*, and the *display module*, each of which is implemented by a double-sided *printed circuit board* (PCB). The kernel of our hardware is a *Motorola MC68HC912B32* microcontroller. The microcontroller operates the chips on the *storage and interface board* through its I/O ports. The *storage and interface board* controls the *display module*, which is a two-color LED array. All the devices used in this hardware are powered at *5 volts*. The overview of the system is shown in Figure 4.1.

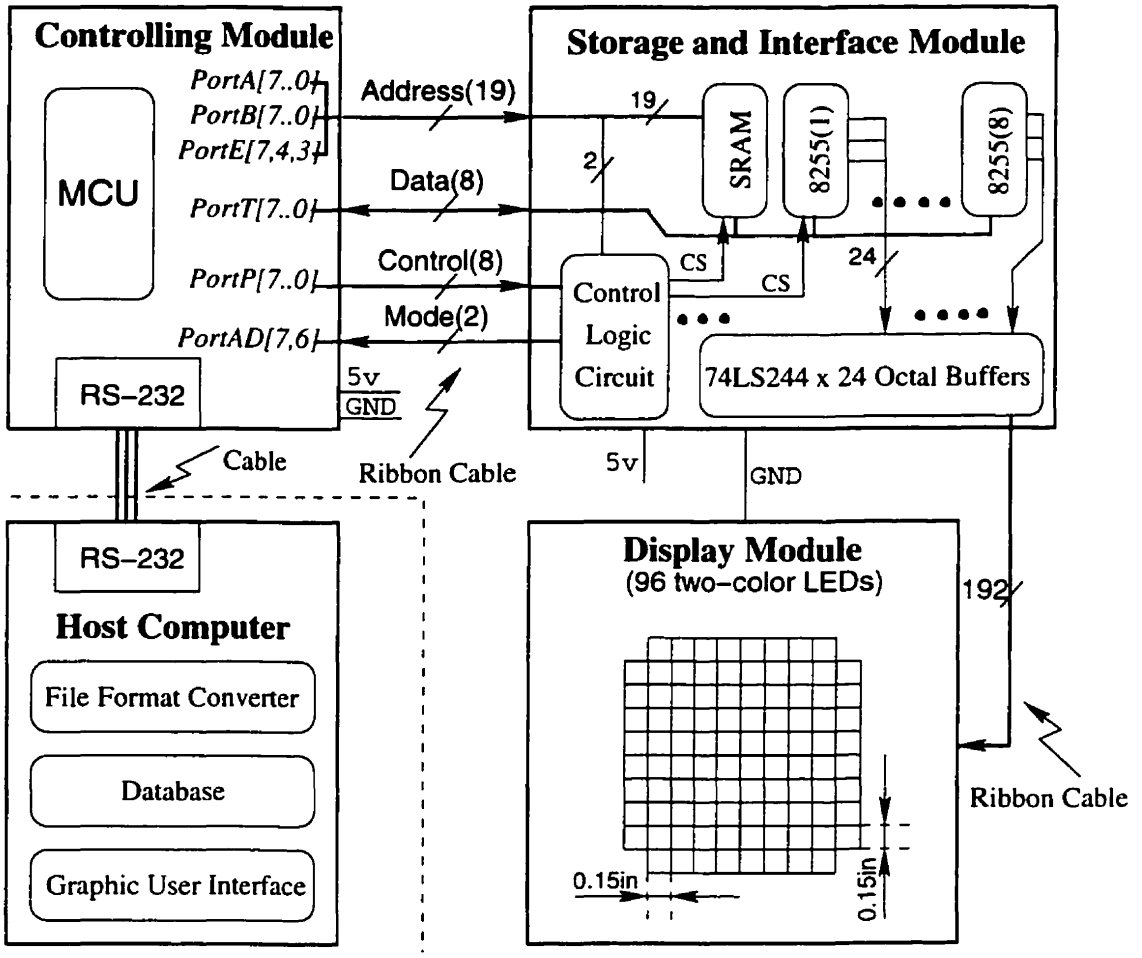

Figure 4.1: System Overview *The system hardware consists of 3 modules, the controlling module, the storage and interface module, and the display module, each of which is implemented by a double-sided PCB. The kernel of our hardware is a Motorola MC68HC912B32 microcontroller. The microcontroller operates the chips on the storage and interface board through its I/O ports. The storage and interface board control the display module, which is a two-color LED array. All the devices used in this hardware are powered at 5 volts. The PCBs are connected through ribbon cables. The controlling module uses an RS-232 port to download data from a host computer.*

4.1.1 Introduction to *MC68HC912B32*

The Motorola *MC68HC912B32 microcontroller unit* (MCU) is a 16-bit device comprised of standard on-chip peripherals including a 16-bit *central processing unit* (CPU12), 32-Kbyte flash *electrically erasable programmable read-only memory* EEPROM, 1-Kbyte *random access memory* RAM, 768-byte byte-erasable EEPROM, an asynchronous *serial communications interface* (SCI), a *serial peripheral interface* (SPI), an 8-channel timer and 16-bit pulse accumulator, an 8-bit *analog-to-digital converter* (ADC), a four-channel *pulse-width modulator* (PWM), and a J1850-compatible *byte data link communications* (BDLC) module. The chip is the first 16-bit microcontroller to include both byte-erasable EEPROM and flash EEPROM on the same device. System resource mapping, clock generation, interrupt control and bus interfacing are managed by the *lite integration module* (LIM). The *MC68HC912B32* has full 16-bit data paths throughout, however, the multiplexed external bus can operate in an 8-bit narrow mode so single 8-bit wide memory can be interfaced for lower cost systems. *MC68HC912B32* uses 64-Kbyte addressing.

The *MC68HC912B32* incorporates eight ports which are used to control and access the various devices and sub-systems. When not used for these purposes, port pins may be used for general-purpose I/O. Additionally, each port consists of a data register which can be read and written at any time. Moreover, almost all of these pins can work in two directions controlled by corresponding direction registers. There are *53* I/O pins at all that can serve as general-purpose I/O pins and work in two directions. They are *Port-A[7:0], Port-B[7:0], Port-E[7:2], Port-P[7:0], Port-T[7:0], Port-S[7:0], Port-DLC[6:0]. Port-AD[7:0]* may work as an 8-bit general input port if not used as an ADC port.

The CPU12 processor supports interrupt service. Each interrupt has an associated 16-bit vector, which points to the memory location where the subroutine that handles the interrupt is located. Vectors are stored in the upper *128* bytes of the standard

64-Kbyte address map. The interrupts include timer interrupts, SCI/SPI interrupts, ADC interrupt, BDLC interrupt, etc. We used one of the timer interrupts to generate the *1ms* intervals in our system. *MC68HC912B32* has an interface for either a crystal or a CMOS compatible clock to control the internal clock generator circuit, which generates the internal clock signals for the CPU and the on-chip peripherals. We used an outside *8-MHz* crystal as the clock standard for the timer interrupt to produce *1ms* time slots precisely.

The serial communication interface on the *MC68HC912B32* is an NRZ format (one start, eight or nine data, and one stop bit) asynchronous communication system, compatible with standard RS-232 systems.

For more information on the *MC68HC912B32*, refer to the *MC68HC912B32 Technique Summary*[24].

4.1.2 Controlling module

To simplify our hardware design, we did not design a PCB and solder a *MC68HC912B32* chip on that, but used an *MC68HC912B32* evaluation board, *M68EVB912B32*, instead, as our *controlling module* and used ribbon cables to connect it and the *storage and interface board.*

Most microcontrollers at this level are high-density packaged. The only available package of *MC68HC912B32* is 80-Pin quad flat pack (QFP). The pins arrangement density is *4 pins per 0.1 inch* and it is surface-mounted. Hence, it is not easy to solder the chip on a board by hand. Because we needed only one system in our perceptual stability research, it was better to use an evaluation board directly. Moreover, a flash EEPROM-resident monitor program, D-Bug12, is factory-configured in the chip. It is ready to use with an RS-232C terminal for writing and debugging user codes. Hence, the evaluation board gives us a way to setup the system quickly - we need not think

about how to write our program into the microcontroller chip or expended memory. Additionally, the evaluation board is very reliable and economical. Connecting the board and a personal computer (PC), which runs *Hyper Terminal*, a communication program, we can download our programs to the chip and debug them easily.

The evaluation board has four operational modes. A user can choose any one of the four operational modes of the evaluation board by configuring two jumpers on the board before the system is reset. The four jumper-selectable modes are *EVB mode, JUMP-EE mode, POD mode, and BOOTLOAD mode* [23]. We used the *EVB mode* to download our programs and to debug them, and use *JUMP-EE mode* to run our program written in the byte-erasable EEPROM of the chip. Set the jumpers *W3* and *W4* as "*1-0*" and then reset the system, execution of the evaluation board will begin directly at location *$0D00* with the user code in the byte-erasable EEPROM.

A program running under DOS, *IASM12*, is used for compiling a text file to a *S-Record* [23] file downloadable to D-Bug12.

However, there are several operational limitations in using the evaluation board. D-Bug12 requires 512 bytes of on-chip RAM for stack and variable storage and it occupies the flash EEPROM. These resources are thus not available for our use. Then we used *768* bytes byte-erasable EEPROM, beginning at *$0D00*, to store our program and *512* bytes on-chip RAM, beginning at *$0800*, during the running of our program. Another operational limitation we have to concern about is the operating mode of the *MC68HC912B32* resided on the board. Although there are seven operating modes of the microcontroller, each of which has an associated default memory map and external bus configuration, the evaluation board was designed for the *Single Chip Mode* of the microcontroller only. That is, it does not support expanded bus operation. In our system, we need to write and read the display data in a SRAM and to write suitable data to the *8255* chips to output. The most convenient way to do this is to connect their data bus and address bus to the microcontroller's data bus and address bus.

Because the microcontroller does not support expanded bus modes, we had to use the microcontroller's I/O ports to operate the devices. Through reading and writing the I/O ports in suitable sequences, the microcontroller emulated right bus operations of all the peripheral chips on the *storage and interface board.*

Ribbon cables are used to connect the I/O ports of the *MC68HC912B32* and the *storage and interface module. Port-E[7,4,3] + Port-A[7:0] + Port-B[7:0]* emulate a 19-bit address bus connecting to the address bus of a 512-Kbyte SRAM. And the lower two bits of the "address bus" connect to the *8255* interface chips as well. *Port-T[7:0]* simulate a 8-bit data bus connecting to the 8-bit data bus of the SRAM as well as the 8-bit data bus of the *8255* chips. I/O pins of *Port-P* are assigned to control lines, including *read/write (R/W), chip-selection (CS), and chip-reset (RST)* lines of the peripheral chips. Two pins of *Port-AD[6,7]* are used as user input switches to expand the program features. All these ports are set as general-purpose I/O ports. The port assignments are shown in Figure 4.1.

4.1.3 Storage and interface module

The *storage and interface module* is controlled by the *controlling module* and drives the *display module.* Ribbon cables are used to implement connections to both sides. There are eight *8255* interface chips, one *HM628512B* 512-Kbyte SRAM chip, twenty-six *74LS244* octal buffer chips, two *74LS245* octal bus transceiver chips, two *74LS138* 3-to-8 line decoder chips, and *268* connections in this module at all. Bypass capacitors are used at all the chips to filter out noise from the power lines.

- **Control word** *Port-P[5:0]* of the microcontroller are configured as output pins and compose a 6-bit control word of the *storage and interface module.* This control word controls directions of the system bus (emulated by *Port-T*), chip selection, and chip reset. Table 4.1 shows the meaning of the control word.

P5-R/W	P4-CSR	P3-CS2	P2-CS1	P1-CS0	P0-RST	Function
X	X	X	X	X	0	Reset all *8255* chips
X	0	1	1	1	1	Stop all chips
0	0	0	0	0	1	SRAM write
1	0	0	0	0	1	SRAM read
0	1	0	0	0	1	8255-0 read
1	1	0	0	0	1	8255-0 write
0	1	0	0	1	1	8255-1 read
1	1	0	0	1	1	8255-1 write
0	1	0	1	0	1	8255-2 read
1	1	0	1	0	1	8255-2 write
0	1	0	1	1	1	8255-3 read
1	1	0	1	1	1	8255-3 write
0	1	1	0	0	1	8255-4 read
1	1	1	0	0	1	8255-4 write
0	1	1	0	1	1	8255-5 read
1	1	1	0	1	1	8255-5 write
0	1	1	1	0	1	8255-6 read
1	1	1	1	0	1	8255-6 write
0	1	1	1	1	1	8255-7 read
1	1	1	1	1	1	8255-7 write
Others						*Wrong function*

Table 4.1: Control Word for the Storage & Interface Module *"P*" means "Port-P[*]". "X" means "don't care". P5 controls the direction of the system data bus, and P0 controls resetting of the 8255 chips. CS bits select chips.*

- **Bus design** There is a 19-bit address bus and a 8-bit data bus in this module. The address bus offers the address of the SRAM and the lowest two bits are

also used to select ports of *8255* chips. The address bus has only one direction, from the *controlling module* to the *storage and interface module.* The data bus is used by the SRAM as well as the *8255* interface chips to transfer data to and from the *controlling module.* The microcontroller writes data to the SRAM during downloading and reads the data out of the SRAM and then writes them to the *8255* chips during display. Hence, the data bus is a bidirectional bus. Because a microcontroller port does not have enough current to drive nine chips, we insert buffer chips between the microcontroller and data bus users. Since the bus is bidirectional, we used *2* octal bus transceiver chips, *74LS245*, to prevent possible bus contentions. We used *Port-P[5]*, the R/W control line, to control the direction of the bus.

- **SRAM** The SRAM *HM628512B* is the storage device in this module. The Hitachi *HM628512B* is a 4-Mbit static RAM organized as 512-Kword × 8-bit. This device has low power consumption and has a short access time of *70ns.* Moreover, it has a plastic DIP package suitable for our board. We used I/O ports of the microcontroller to emulate the writing and reading timing waveforms. The timing waveforms of SRAM reading and writing are shown in Figure 4.2 and Figure 4.3.

- **8255 interface chips** The *8255* is a general purposed programmable I/O device. Each *8255* has twenty-four tri-state I/O ports. We used eight of these devices to produce *192* outputs to drive the two-color LEDs. There is a latch at each output of the chip. The microcontroller writes a desired output value to one of the devices, and the latches keep the value stable while we write values to other *8255* chips. The *8255s* must be reset before use. A positive edge on W/R line during CS line low fires a writing operation to the chip. The timing waveforms of *8255* writing operation are shown in Figure 4.4.

- **LED driving circuits** The working current of each color LED is about *20mA*. Each *8255* controls *24* color outputs. If we let a *8255* drive the *24*

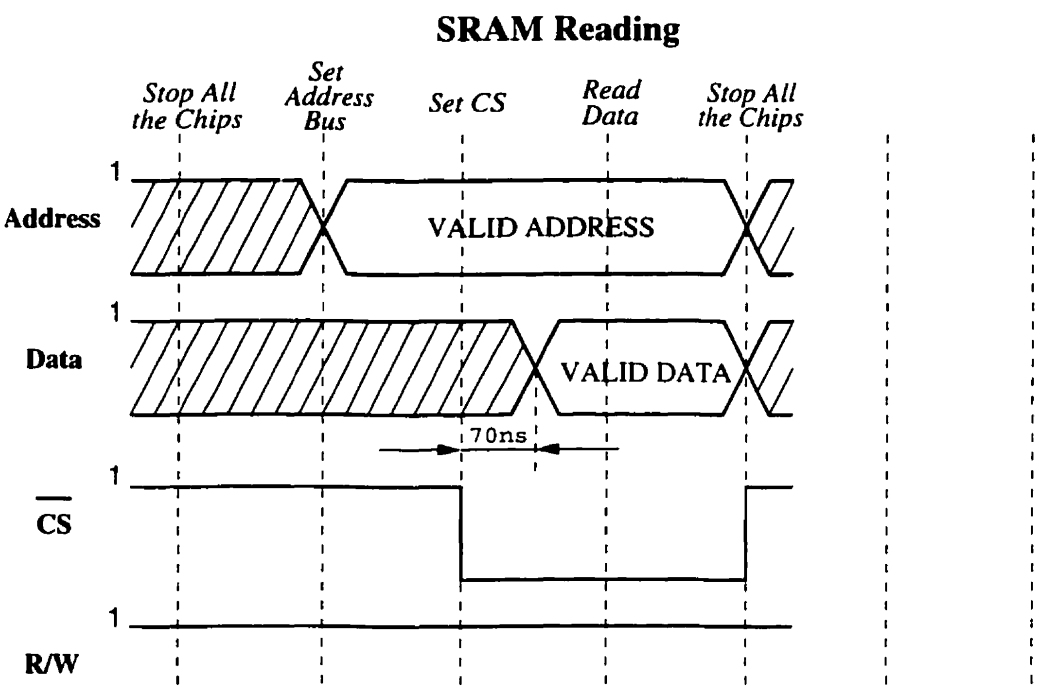

Figure 4.2: Timing Waveform of SRAM Reading *To avoid any possible wrong operations, all the chips are disabled before setting the address bus. When the address bus is stable, we select the chip. After 70ns, the data on the data bus is valid. Because all the operations are done by running instructions of the microcontroller, and the program counter of the microcontroller works at microsecond level, this operation sequence works well in reading data out from the SRAM.*

outputs directly, the output of the chips is up to *480mA*. However, the drive capacity of each *8255* I/O port is only *2.5mA* [20]. Therefore, we use octal buffer chips, *74LS244*, to drive the LEDs. Each *74LS244* drives eight color LEDs. The LED driving circuit is shown in Figure 4.5.

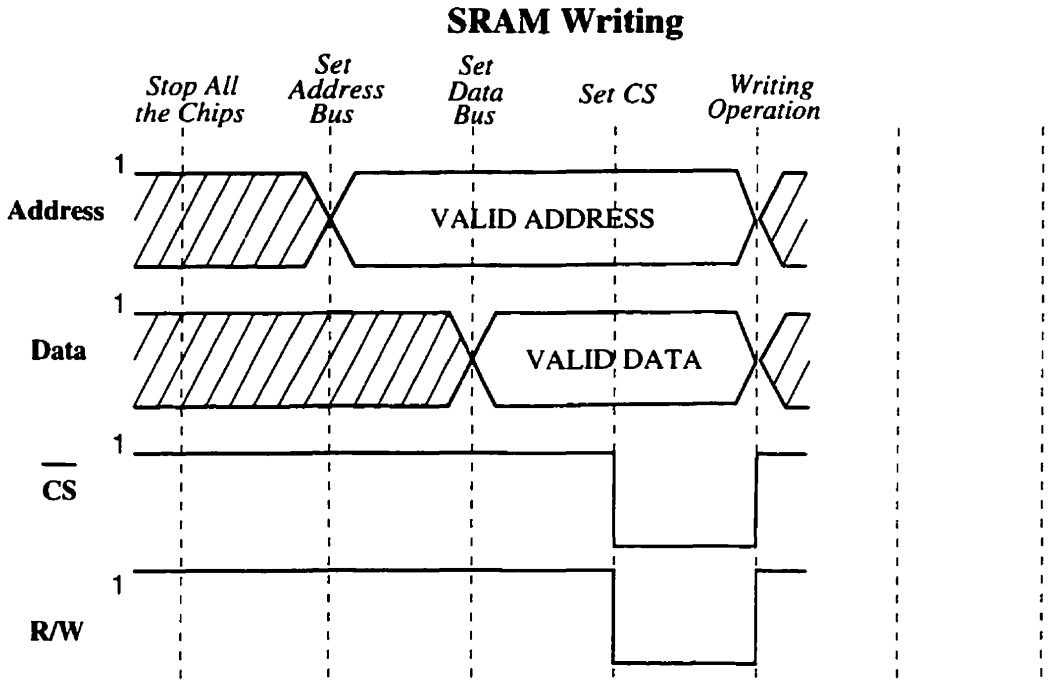

Figure 4.3: Timing Waveform of SRAM Writing *To avoid any possible wrong operations, all the chips are disabled before setting the address bus and the data bus. A positive edge on R/W during CS line low fires a writing operation. And CS and R/W can become high at the same time. Because both CS and R/W are controlled by Port-P of the microcontroller, these two operations can be done at the same time.*

4.1.4 Display module

The *display module* is a separate circuit board on which *96* two-color LEDs are mounted. There is a *200ohm* current limiting resistor between each common cathode and the ground. The *96* LEDs compose a 10×10 square array, with four empty corners. The LEDs are placed in a $0.15\ inch \times 0.15\ inch$ grid. They are grouped by *24* devices, which use a common ground line connected to the *storage and interface module*. Although the LEDs can display three colors, the mixed color is not harmonious with the others. So we recommend using only the green and the orange colors

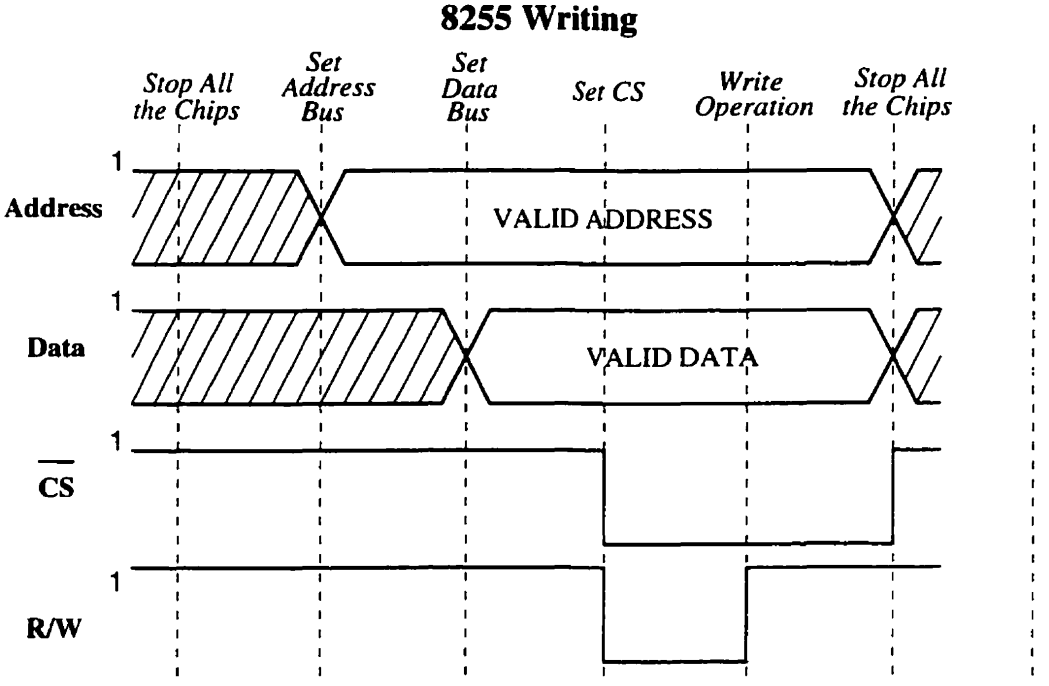

Figure 4.4: Timing Waveform of 8255 Writing *To avoid any possible wrong operations, all the chips are stopped before setting the address bus and the data bus. When the two buses are stable, select the chip. A positive edge on R/W during CS line low fires a writing operation. Different to SRAM writing waveform, these two positive edges cannot happen together. Thus, writing a 8255 needs more operations than writing to or reading from a SRAM.*

in the perceptual stability research. To balance the driving current, the two anodes of a given LED are controlled by the same port of a *8255* chip.

4.1.5 Power supply

All of the PCB boards use the same *5 volts* power supply. The *storage and interface board* and the *display module* consume much more power than the *controlling module*

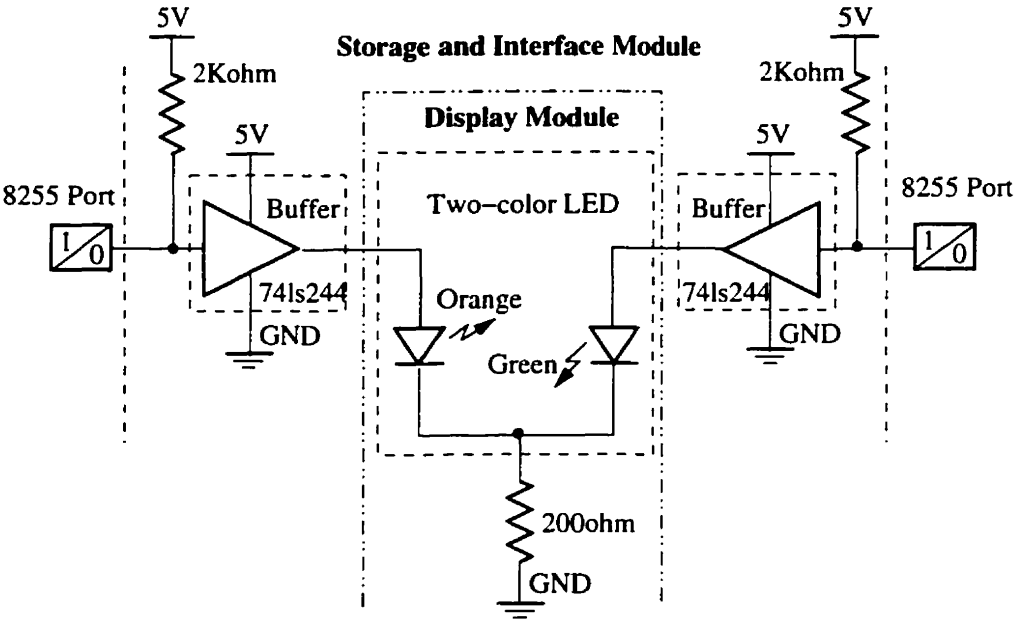

Figure 4.5: Driving Circuit of a Two-Color LED *Each two-color LED is controlled by two 8255 ports. A 1 at the 8255 port lights up the color it controls. The buffer supplies the driving current. A 200 ohm resistor is included to limit the current.*

does. The current fluctuations in these two modules is much serious than that in the *controlling module* as well. Therefore, to improve the working stability of the *controlling module*, it is powered independently to the other two modules. A power line and a ground line are connected to this module from the power supply directly. The other two boards share the same power line and ground line, as shown in Figure 4.1.

4.1.6 Serial communication

The *MC68HC912B32* microcontroller has an asynchronous serial communication module which is compatible to RS-232 interfaces. An RS-232 interface is mounted on the *controlling module*. Our system uses this module to receive data from a host computer. In asynchronous serial communications no clock signal is needed for synchronization.

Rather each character is resynchronized by means of a start bit.

The serial communication interface on *MC68HC912B32* is an NRZ format (one start, eight or nine data, and one stop bit) asynchronous communication system with independent internal baud rate generation circuit and an SCI transmitter and receiver. Using a *8MHz* crystal as external standard clock, the highest baud rate that the *MC68HC912B32* supports is *38,400bps*. We did not use parity generation, and therefore, we need *10* bits to transfer *1* byte of data. Each image frame is expressed by *24* bytes. Hence, at this speed, downloading one pattern, that is *1024* frames, we need $1024 \times 24 \times 10/38400 = 6.4$ seconds.

The configuration of the SCI includes data format register configuration and baud rate configuration. During download, the *MC68HC912B32* waits for the coming data by checking SCI status register. When data comes, it writes the received data into the SRAM. The first byte of the download file is the number of patterns contained in the download file. Followed by this are the bytes indicating the number of repetitions of each pattern. The following bytes are those that express each frame of each pattern. When the system has received all the data it expects, the program jumps out of the download subroutine.

4.1.7 Program control flow diagram

We have a program residing in the byte-erasable EEPROM of the microcontroller. If we set the two jumpers of *W3* and *W4* as *"10"* before resetting the evaluation board, the system will execute this program. There are two modes of this program. If the DIP switch on the *storage and interface board* is set as *"00"*, the program will download a display file to the SRAM and then begin the display. While if the DIP switch is set as *"10"*, the program begins to display the file in the SRAM directly. The program starts after each reset of the *controlling module*. The program flow of

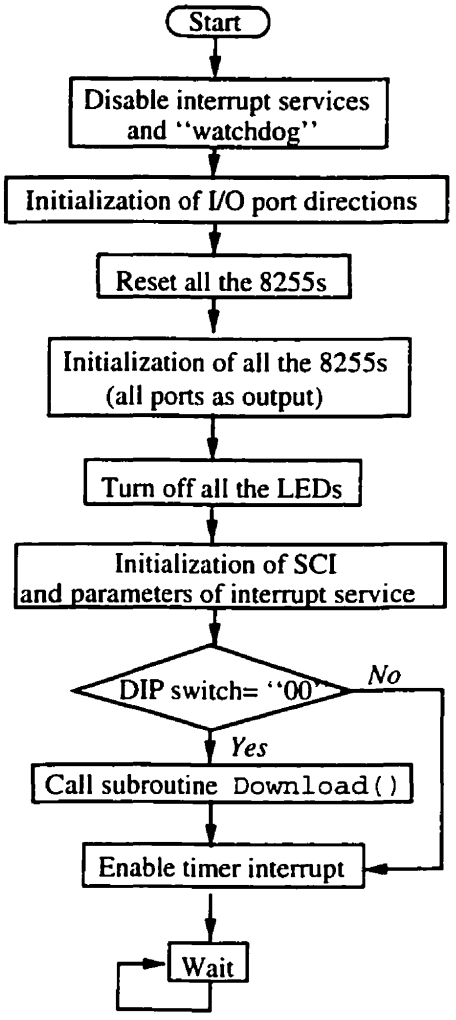

Figure 4.6: Program Flow of the Main Program *The main program will stay in a loop at the end and wait for the timer interrupts. During downloading, the microcontroller writes each byte received from the serial communication port to a suitable address in the SRAM. Several counters in the microcontroller's RAM work as the address pointer. SRAM writing timing waveforms are emulated in the Download() subroutine.*

the main program is shown in Figure 4.6.

After initialization and downloading, the main program enables the timer interrupt

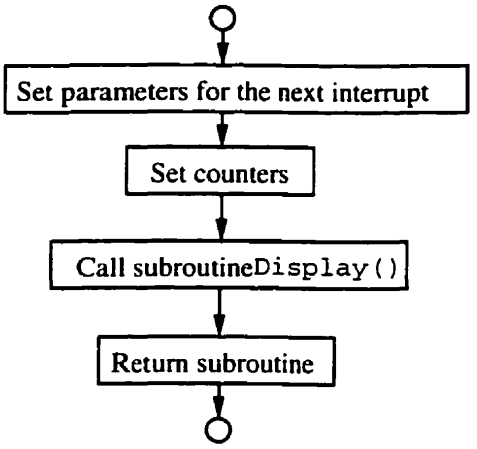

Figure 4.7: Program Flow of the Timer Interrupt Service Program *This subroutine runs at each 1 ms and refresh the display module based on the data read out from the SRAM by calling the subroutine of Display() (Figure 4.8). Several counters in the microcontroller's RAM are used as the address pointer.*

and waits in an endless loop. A timer interrupt occurs, and the interrupt service program runs, every *1ms*. The interrupt service program sets the counters in the microcontroller's RAM and reads twenty-four bytes from the SRAM. The LED array is refreshed based on this data. Then the interrupt service program sets the parameters of the next timer interrupt and the program counter returns back to the main program, waiting for the next interrupt. The program flow of the interrupt subroutine is shown in Figure 4.7.

4.2 Software design

We need software in our system to convert a user's idea to a downloadable file. *Pattern-Maker* is such a software that we designed, running under *Windows*. *Pattern-Maker* helps a perceptual stability researcher express the display patterns in his or

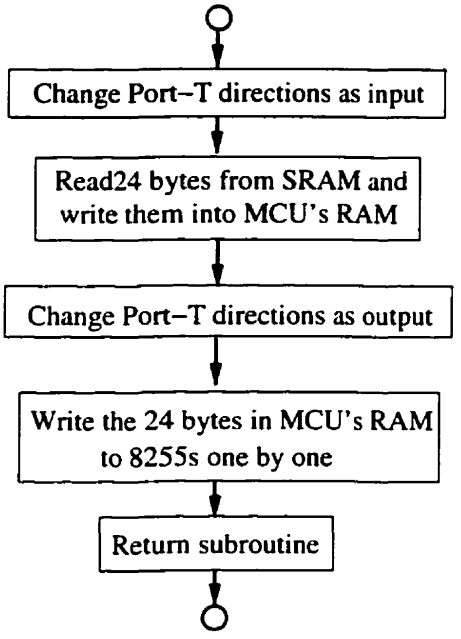

Figure 4.8: Program Flow of the Subroutine of Display() *The direction of the data bus changes in this subroutine. The address pointer is set in the "timer interrupt service program" already when this subroutine is called. The 8255 writing and SRAM reading waveforms are emulated in this subroutine.*

her mind easily and produces a downloadable data file. The user may design any possible display pattern that our LED panel can display, however simple or complex. It also offers a way to organize up to 16 patterns into a file to download. In this step, a user may indicate the number of repetitions of each pattern. *Pattern-Maker* can download the file through a serial communication port at *38,400bps*.

Pattern-Maker uses a *Pattern-Group-Frame* strategy to express a display pattern in the time sequence. The design is stored in a *Microsoft Access Database*. A graphical user interface (GUI) programmed with *Microsoft Visual Basic 6.0* helps the user view or edit a pattern at any time.

4.2.1 Patterns, Groups, and Frames

We use *Pattern*, *Group* and *Frame* to express a video pattern. One *Pattern* consists of some *Groups*, and one *Group* includes one or more *Frames*.

- **Pattern:** is *1024* images in a sequence, with each image having *96* pixels. A pattern is composed of different *Groups*.
- **Group:** is *1024* images in a sequence, where each image is composed of pixels in a subset of the *96* pixels of the LED display panel. All of these pixels change color concurrently. A *Group* is composed of different *Frames*.
- **Frame:** is an image sequence, of which all the images are same and the pixels of the images are the ones of the *Group* it belongs to.

While designing a display pattern, a user should divide the display panel into several *Groups* at first. All of the LED pixels in the same *Group* change colors at the same time during the whole display sequence. The user may then design the *Groups* one by one. Editing the colors of the LEDs in a *Group*, and giving a number indicating how long this image extends, completes the design of a *Frame*. Filling all *1024* images of the group with a number of *Frames* completes the *Group* design. Additionally, in many situations, a researcher may want some or all the LEDs flash repeatedly. To simplify this kind of display, the user may leave some of the *1024* images of a *Group* empty. Then *Pattern-Maker* will finish the whole sequence by repeating the *Frames* already designed.

Obviously, the *Pattern-Group-Frame* expression can express all the possible display sequences clearly. On the other hand, it is convenient to design repeating displays. It is a straightforward top-to-down design method and suitable for the perceptual stability research.

4.2.2 Database for the patterns

Pattern-Maker uses a *Microsoft Access Database* to manage the entry data. The user may view and edit the data easily. The GUI is programmed using *Microsoft Visual Basic 6.0*, which has an interface to *Microsoft Access*.

Using Visual Basic is a quick and easy way to create powerful, full-featured applications that take advantage of the graphical user interface in *Windows* and *Windows NT* operating systems. It reduces time and costs in developing custom applications. Moreover, Visual Basic supports data access features that make it an excellent language for quickly creating full-featured solutions [10] [9]. Using data access features, we can create databases, front-end applications, and scalable server-side components for most database formats. Embedding *structured query language* (SQL) in Visual Basic language, we may do any operations on a database connected to Visual Basic [25]. In our design, we used both SQL embedding technology and ADO *(ActiveX Data Objects) Data control* of Visual Basic to finish the task.

A *Microsoft Access* database may include some *tables*, each of which contains some *records*. And each *record* consists of some *fields*. There is only one *table* in our database. And the number of *records* depends on the complexity of the pattern designed. Generally to say, one *frame*, one *record*. Additionally, adding each group adds a *record* as well. Each "frame *record*" records the color of each LED pixel of the frame, the extending time of this frame, the position of the frame in the display sequence, and the number of the group it belongs to. Each "group *record*" records the pixels in this group and the number of the frames it includes. A special *record*, the "pattern *record*", records the total number of groups in this pattern. And both "frame *record*" and "group *record*" have their own IDs and names for the users to find them. The fields of a record in our database are shown in Figure 4.9.

A simplified example of the database is shown in Table 4.2. There are 7 fields in

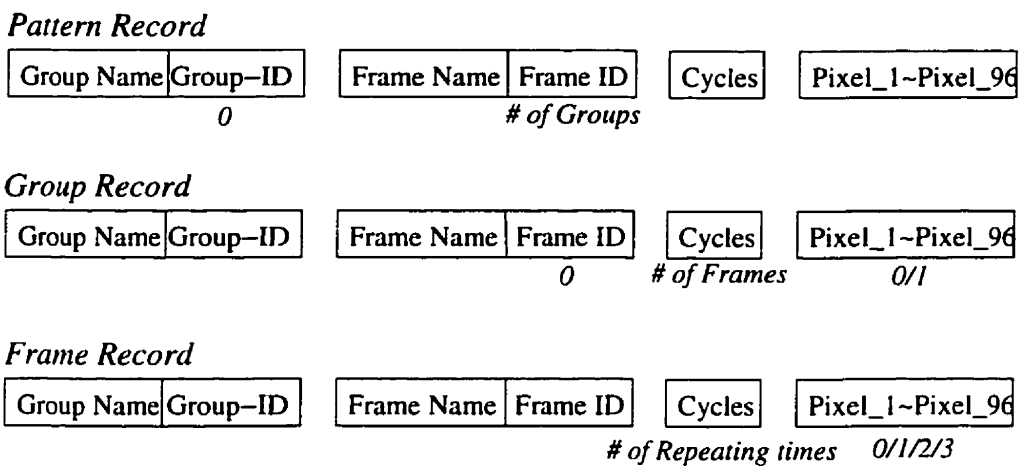

Figure 4.9: Fields of a Record in Our Database *There is only one* ***pattern record*** *through out the database. The "Group-ID" of this record is 0, and the "Frame-ID" of this record indicates the total number of groups in this pattern. A* ***group record*** *represents a group, and its "Frame-ID" field is 0 and its "Cycles" field indicates the total number of frames in this group. Each* ***frame record*** *represents a unique frame. The "Group-ID" indicates the group it belongs to; the "Frame-ID" is the sequence number of this frame, and the "Cycles" is the number of repetitions of this frame. For a* ***frame record****, in the "Pixel" fields, "0" means "white", "1" means "green", "2" means "orange", and "3" means "display two colors". For a* ***group record****, in the "Pixel" fields, "0" means that the pixel is not in this group, and "1" means that it is.*

each record of this database, from "Pattern-ID" to "Pixel-4" in the table. In our real database, each *record* has *92* more "Pixel" *fields*, from "Pixel-5" to "Pixel-96", and *2* more "Name" *fields*. However, their structure and style are similar.

Group-ID	Frame-ID	Cycles	Pixel-1	Pixel-2	Pixel-3	Pixel-4	*Remark*
0	2	X	X	X	X	X	*1:Pattern*
1	0	4	1	1	0	0	*2:G1*
1	1	256	2	1	X	X	*3:G1-F1*
1	2	256	0	0	X	X	*4:G1-F2*
1	3	256	1	1	X	X	*5:G1-F3*
1	4	256	3	0	X	X	*6:G1-F4*
2	0	2	0	0	1	1	*7:G2*
2	1	40	X	X	2	0	*8:G2-F1*
2	2	50	X	X	0	1	*9:G2-F2*

Table 4.2: A Simplified Example of the Database *This is a simplified database for display in which "G" means "Group", and "F" means "Frame". "X" means "Don't care". In the "Pixel" fields, "0" means "white", "1" means "green", "2" means "orange", and "3" means "display both colors". Record 1:Pattern is the* ***pattern record*** *in this example. Record 2:G1 and Record 7:G2 are* ***group records****. The other 6 records are examples of* ***frame record****. This pattern has 2 Groups. Group 1 has 4 frames and Group 2 has 2 frames. Because the 2 frames in Group 2 do not fill out all the 1024 images in the display sequence, the program will repeat frame 8:G2-F1 40 times and frame 9:G2-F2 50 times again and again to produce a sequence containing 1024 images.*

4.2.3 File format conversions

This section introduces the data format conversions in *Pattern-Maker*. We used four file formats through out the software. They are *.mdb*, *.txt*, *.ptn*, and *.dld* format.

The *.mdb* format is the *Microsoft Access* database file format. As we discussed above, a user may record his or her idea to a *Microsoft Access* database first. The data are organized in a format that can be edited easily. When the user finishes his or her

design of a pattern, the database recording the display information is converted to a *.txt* file, which is a normal text file composed of *ASCII* words. We defined this format because when *Windows* is not available, we may design a pattern by writing a text file under any other operating systems.

Then the *.txt* file is converted to a *.ptn* file. *".ptn"* means *"pattern"*. Each of the *1024* images in the display sequence is produced in this step. We use two bits to control one pixel, a two-color LED, as we described in Section 4.1. If we want to display green or orange, we write one *0* and one *1* to the two bits. If we want to display both colors, we write two *1s* to the two bits. If no display, write two *0s*. A *checking table* is used to write the control bits of a two-color LED at the corresponding place of the file. Since each *96-pixel-image* is expressed by *24* bytes, all the *.ptn* files have the same size, that is, $1024 \times 24 = 24576$ bytes.

4.2.4 Data transmission

The downloadable files (*.dld*) are transmitted to the hardware through serial communication interfaces. The program opens an SCI port and then configures the parameters, including *Baud Rate, Parity, Message Length*, and *Transmitting Direction*. For our system, the SCI port at the *host computer* is configured as *38,400bps, no parity, 8-bit message*, and *output*. After configuration, the program transfers a *.dld* file through the serial communication interfaces. The first byte is the total number of patterns containing in this file. Then the repeating times of each pattern are transmitted. After these bytes, the data of the patterns are transmitted one by one. The highest baud rate that our hardware supports is *38,400bps*. The transmitting speed of our system is *6.4 seconds per pattern*.

Although no parity and handshaking is used, the data transmission is reliable. We did an experiment to test the reliability of the serial communication. In this ex-

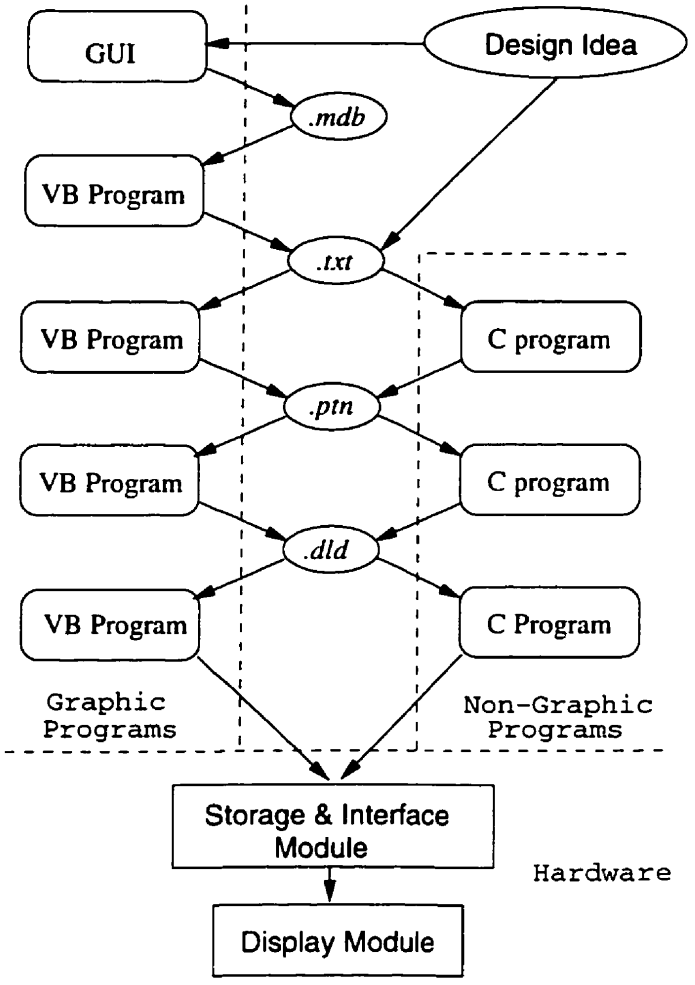

Figure 4.10: File Format Conversions *A user can enter his or her design ideas by "Graphic Programs" or "Non-Graphic Programs". And the user can change programs at any step. The "Graphic Programs" run under Windows or WindowsNT, and the "Non-Graphic Programs" run under Unix.*

periment, we downloaded *1 Mbyte* data to the hardware, and the data were filled with *10101010B*. On the hardware side, the microcontroller tested the received data, and counted the number of transmitting errors. We found no errors. Therefore, this transmission strategy is suitable for our research.

4.2.5 Solution for simple designs

Using the method introduced above we may design any possible display sequence that is *1024* frames long. And all the sequences use *24K* bytes to express and need the same time, *6.4* seconds, to download. However, many redundancies exist in the expression of a simple sequence. We then designed programs to design simple display sequences to save download time.

These programs are used to design the *Patterns* that have exactly two independent LED subsets displaying periodically. Such a sequence needs only *53* bytes to express, two for periods, two for duty times, one for phase, and two 24-byte data sequences indicating the two subsets (Figure 4.11). Then we need to download only *53* bytes. And the user will not perceive the downloading time, which is about *15ms*. In this case, the microcontroller uses another program to receive and display the *simple patterns*.

4.3 System features

This section describes the properties of our system.

4.3.1 Analysis of the timing accuracy

In this subsection we analyze the accuracy of our display. Our hardware displays an image at each *1ms*. At the beginning of each *1ms* interval, the hardware reads *24* bytes of data out of the memory chip and writes them to the *8255* I/O ports one by one. Because the writing procedures consume time, timing differences exist among the displays of the bytes. The eight *8255s* are written one by one. And for each chip, the three ports are written one by one. Therefore, the delay between the first

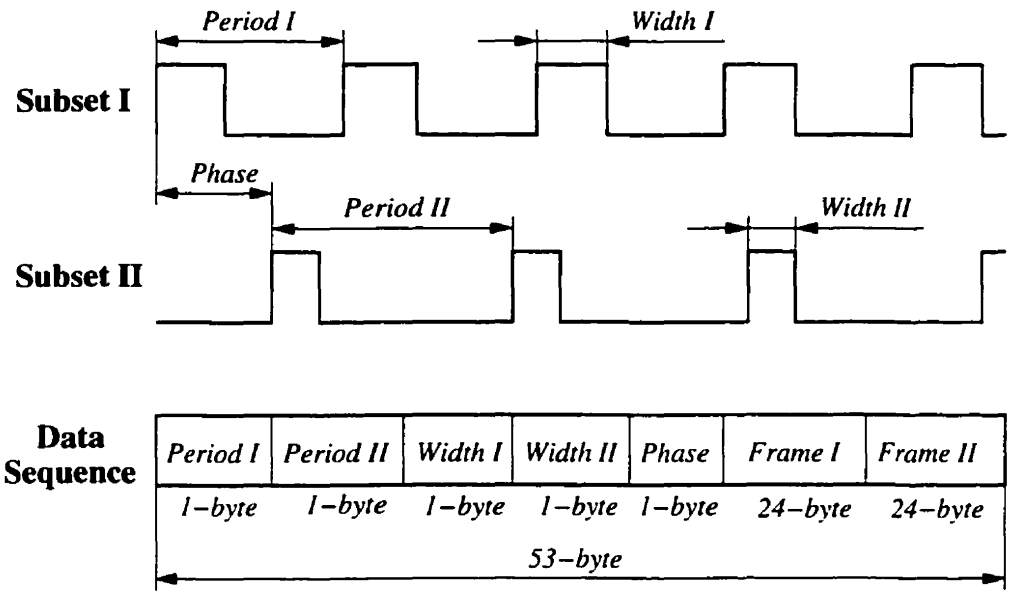

Figure 4.11: Simple Patterns *Frame I and Frame II indicate the two LED subsets. The 192 bits of the 24 bytes of Frame I and Frame II represent the two colors of all the 96 LEDs.*

byte being displayed and the twenty-fourth byte being displayed is approximately the executing time of twenty-three *8255* writing processes. And each *8255* writing process includes processing of several microcontroller instructions. The processing time of an instruction of the *MC68HC912B32* at *8MHz* is at the microsecond level, so this delay may influence the accuracy of the display system. Therefore, at each interval we read all the *24* bytes data from the outside memory to the microcontroller's on-chip RAM first and then display them to reduce the number of instructions executed between the continuous two writing processes of the *8255* I/O ports and increase the accuracy of our system.

We carried out an experiment to test the longest delay at each *1ms* interval. We chose two LEDs to display, one of which was controlled by the first port of the first *8255* chip, and the other one was controlled by the last port of the last *8255* chip.

We let the two LEDs switch at each *1ms* interval and measured the delay. The result was *0.1ms*.

Another timing error source is the mechanism of producing the *1ms* interval. We used the microcontroller's timer interrupt to do this. The timer interrupt of the *MC68HC912B32* uses a standard timer module as its standard clock. The standard timer module consists of a 16-bit software-programmable counter driven by a prescaler. The module clock is divided by the prescaler and then serves as the clock of a 16-bit counter in the module. When the value of the counter is as same as the configured value of a 16-bit comparator, which is set by a timer register, a timer interrupt occurs. For higher resolution and accuracy, we configured the prescaler as *1*. The module clock comes from the outside crystal, which is *8MHz* in our system. And therefore, the accuracy of the *1ms* generation may reach to 1/8 microsecond. The outside crystal is a very stable timing standard device. Hence, even the crystal is not accurate as *8MHz*, we can use a high precision measuring equipment to get the true value, and change the configured value in the 16-bit comparator of the microcontroller's timer module to generate precise *1ms* intervals.

Therefore, the timing error mainly comes from the serial operation of the *8255* I/O ports, and the maximum accuracy of our display is *0.1ms*. However, if the display pattern is simple, we can use the LEDs controlled by the same *8255* to improve the accuracy. In this case, the accuracy can reach to two twenty-thirds of *0.1ms*.

4.3.2 Operation of the system

This subsection reviews the operation of our system.

Firstly, prepare the video patterns to display through *Pattern-Maker* or by editing txt files. And then convert the format of the files to *.ptn*. After that, the user may organize the downloadable file by choosing some patterns and arrange them in a

suitable sequence and setting corresponding repeating times of each pattern. Up to *16* patterns can be arranged in each downloadable file.

Secondly, connect the SCI ports of the hardware and a host computer. Power up the hardware, and reset the boards by click the button on the *controlling module.* Guarantee that the jumpers *W3, W4* are set as "*10*" and the DIP switch on the *storage and interface module* are set as "*00*" before reset. Then the *display module* is cleared to blank and waits for data.

Finally, run a download program in the host computer to download your display file to the hardware. Some LEDs will flash during the download. When the download finishes, the *display module* will display the patterns you designed and organized cyclely. The downloading time for each pattern is about *6.4* seconds.

A user can restart at the display of the first pattern by setting the DIP switch as "*10*" and reset the hardware again. Because the SRAM chip will lose all the data after power down, the user has to download the file to display after each power up.

4.3.3 A summary

Because all the LEDs are independent in our system, a user can display any possible combinations of the colors. Our system consists of two parts, software and hardware. The software is used to design the displays and the hardware is used to execute the display. They connect to each other through serial communication interfaces. The hardware can display up to *16* different patterns after each download.

Chapter 5

Experiments and Analysis

Using the programmable display system described in the previous chapters, we carried out experiments to test a hypothesis on perceptual stability. The high time resolution and flexibility of our display system offer us a way to do precise measurements on humans' perceptual stability. Based on these measurements, we studied the phenomenon of perceptual stability in detail. Although the experiments introduced below are preliminary, they move us closer to an understanding of the mystery of humans' visual perception.

Our hypothesis on how humans perceive stability is introduced in Section 5.1. This hypothesis is derived from the *saccade target theory* and supposes the detailed strategy that humans perceive stability during eye movements. Two experiments were designed to test our hypothesis. The results and analysis of the experiments are described in Section 5.2 and Section 5.3. These results show the firm relationship between humans' visual attention and the problem of perceptual stability, and form a foundation of the perceptual stability research in the future.

5.1 Hypothesis

Our hypothesis on humans' perceptual stability is derived from the *saccade target theory* (Section 2.3). Before an eye movement, the human visual system gathers some important features of the *saccade target* and produces a *template*. After the eye movement, the *saccade target* is inspected and a new *template* is produced. This new *template* is compared to the old one. If the new *template* corresponds to the old one, we perceive stability, if not, we perceive instability.

Moreover, we suppose a more detailed model of human eyes in perceiving stability. In this model, the human visual system computes the first *template* after it makes the decision to move the eyes, and computes the second *template* immediately after the completion of the saccade. Figure 5.1 shows the model we supposed.

Based on the attention theory, an eye movement begins from the generation of a new attention shift. After the attention shift (period T_A), a *saccade target* is identified and the visual-motor system makes a decision to move the eye to the target (period T_D). Before the saccade, some key features of the *saccade target* are extracted and a template of these key features is computed and saved in a memory (period T_T). After this, the visual-motor system prepares for a saccade (period T_{Tr}) and then a saccade occurs (period T_M). At the end of the saccade, the visual system extracts the features of the new scene and computes a new template based on these informations (period T_C). The two templates are compared. If they correspond to each other, under some tolerance, the human perceives stability; otherwise, the human perceives instability.

We studied our hypothesis in the case that a scene appears periodically. An object appears in the visual world periodically and is visible for only a short period in each cycle. The attention shift period (T_A) may be long and cover several flashing periods. After the attention shift finishes, the visual-motor system waits for a while (T_D) before starting the eye movement until the target that triggers the attention

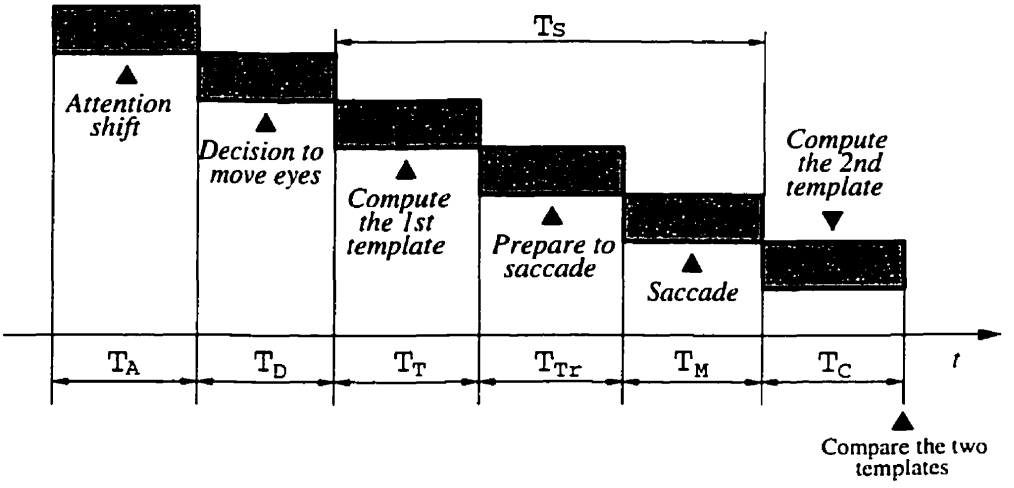

Figure 5.1: Perceiving Stability I (Model) *This figure shows the supposed steps involved in humans eye movements. T_A is the period of attention shift. T_D is the time for making a decision to move the eyes. T_T is the time for computing the first template. T_{Tr} is the time for the visual-motor system to prepare for a saccade. T_M is the executing time of the saccade. This value depends on the amplitude of the eye movement. T_C is the time for computing the second template during the next fixation. Then the two templates are compared and the human perceives stability or instability based on the result of this comparison. $T_S = T_T + T_{Tr} + T_M$.*

is visible again. Because of the mechanisms that the human brain uses to process the data, when the visual system computes the templates (during T_T and T_C), it extracts the average values of the key features. Hence, different flashing periods may lead to different perceptions, stability or instability. Figure 5.2(a) shows a case where humans perceive instability and Figure 5.2(b) shows a case where humans perceive stability.

We used our programmable display system to simulate these cases and carried out some preliminary experiments as described in Section 5.2 and Section 5.3.

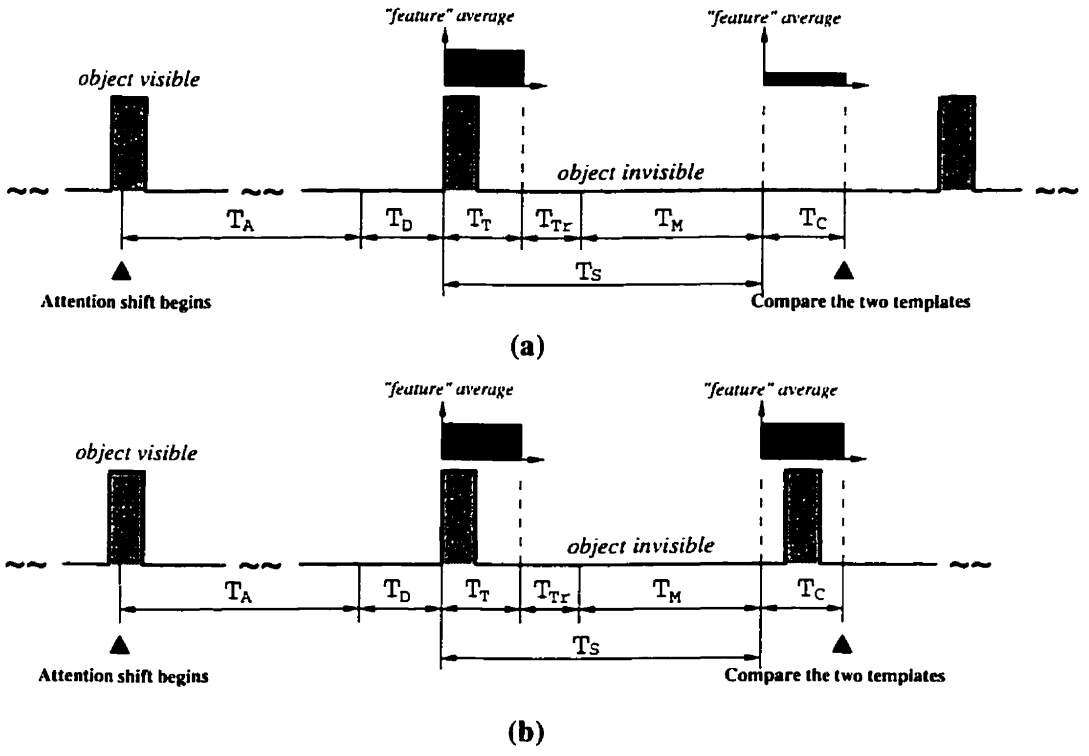

Figure 5.2: Perceiving Stability II (Case Study) *A scene appears periodically and the visual scene extends for a short period each time. (a) The case of long periods. If the object does not appear immediately after the saccade, the average values of the key features computed as the second template during T_C are quite different to the first template. Thus, the human perceives instability. (b) The case of shorter periods. If the flashing period is shorter, the object will appear again when the visual system computes the second template during T_C after the saccade. Then the second template corresponds to the first one and the human perceives stability.*

It should be noticed that the above hypothesis and analysis also apply on the cases that the display patterns are not periodic. Using an eye-tracker to record movements of the eyeballs during showing display patterns to the subject, we may do more elaborate researches. For preliminary experiments, we only used simple periodic display patterns. In these cases, we knew saccades happened, while we did not know

the exact time they happened.

5.2 Experiment A

This experiment was based on the case study introduced in Section 5.1. It was designed to observe the influence of the flashing period on the visual perceptions. The flashing period boundary of perceiving stability or perceiving instability was measured for each subject.

5.2.1 Motivations

Based on the case study in Section 5.1, the stability perception is sensitive to the period of the visible signal. Thus we designed an experiment to study the relationship between stability perceptions and the signal periods. For each subject, we changed the flashing periods of two LED groups in the *display module*, and let the subject say whether the display was stable or not. We recorded the responses and drew a *Flashing Period - Perception* curve for each subject.

5.2.2 Experiment setup

The experiment was done in a dark room. The *display module*, on which a periodic flashing pattern displays, was located *20-25cm* in front of a subject's eyes. The subject was asked to report whether the visual pattern was stable or unstable within *5* seconds and was permitted to guess among the only two possible choices if he or she could not make a clear decision. The subject started a new display pattern by pressing a "reset" key on the *display module*. The period of the flashing pattern and the subject's response were recorded.

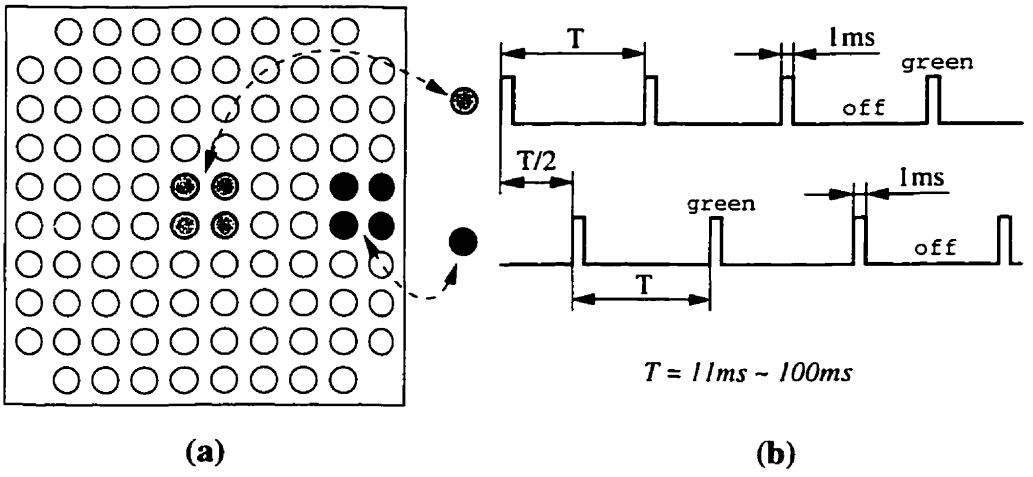

Figure 5.3: Experiment A Setup *(a) All the display patterns included two 2×2 LED groups, as shaded in this graph. The four LEDs in each of the two groups output green color and were turned on and off periodically with the same mode as shown in (b). (b) The two LED groups flashed with the same period but different phase. The phase difference between the two groups was always 180°. The turn-on time was always 1ms, whatever the period was. The period varied from 11ms to 100ms.*

All the display patterns included two two-by-two LED groups in the LED array, as shown in Figure 5.3(a). The two LED groups were turned on and off periodically with the same period varying from *11ms* to *100ms* and a constant phase difference of *180°*. The turn-on time was always *1ms* of both LED groups of all the display patterns, as shown in Figure 5.3(b).

Because some display patterns we tested were periodic with low frequencies, subjects would detect flicker. However, perception of flicker and perception of movements are different. To avoid possible confusions between flicker and movements, we prepared some warming-up display patterns and showed them to the subjects before formal tests. The warming-up patterns are composed of the same two 2 × 2 LED groups and are periodic with low frequencies with a *0°* phase or a *180°*. All the subjects could

clearly distinguish the two phenomena after several warm-up displays.

We prepared *90* display patterns, with flashing periods as *11ms, 12ms, 13ms, ... , 100ms*, respectively, to test the subjects. From a preliminary test to some subjects, we found that motions were always visible when the flashing period of the display is long. We showed the display patterns with a flashing period longer than *45ms* to each subject only one time, while showed the display patterns with a flashing period shorter than *45ms* to each subject eight times. The display patterns were showed randomly. Each of the six subjects were shown all of the flashing patterns, for a total of *335* trials.

5.2.3 Data and results

For each subject, we calculated the *normalized frequency of detecting movements* of each display pattern and generated a plot of the relationship of the normalized frequency to the flashing period. All the subjects perceived motions when the flashing period was long and perceived stability when the period was short. A typical plot is shown in Figure 5.4. The location of the slope varied from *20ms* to *40ms* for the six subjects.

5.2.4 Experiment analysis

Our analysis is based on the case study in Section 5.1. Only if the turn-on period of the object is coincident with the time period that the visual system is computing the second template after the saccade, stability is perceived. Therefore, the relationship of stability perception to the flashing period should be that shown in Figure 5.5(a). The length of the right most *stable period* (shaded in Figure 5.5(a)) equals to T_C and the right most negative edge is at $T = T_S$. Because a kth ($k > 1$) turn-on period after T_T may locate at T_C as well, there is a kth *stable period*. The kth *stable period*

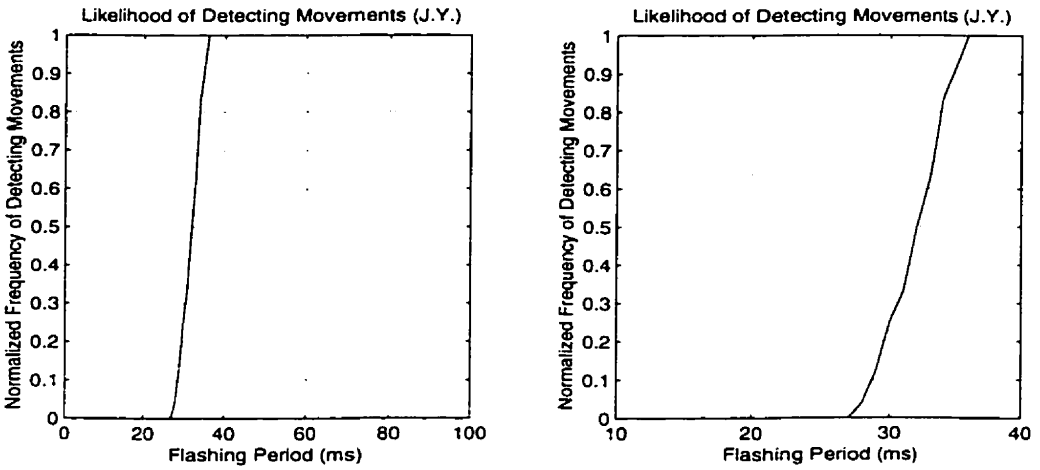

Figure 5.4: Typical Data of Experiment A *Subject J. Y.'s data of the likelihood of detecting movements. The data was filtered with [1 1 1] and normalized to numbers between 0 and 1. The horizontal axis represents the flashing period and the vertical axis represents the normalized frequency of detecting movements. The left graph shows all the data; the right one only shows the data for flashing periods shorter than 40ms.*

begins at $T = T_S/k$ and is T_C/k long.

However, we only found one slope in each data graph, as shown in Figure 5.4. There are two possible reasons for this observation.

First, different *stable periods* merge when k is big enough. The condition that a *unstable period* is observable is show in Equation 5.1:

$$\frac{T_S}{k-1} - \frac{T_S}{k} > \frac{T_C}{k}, \quad or, \quad k < \frac{T_S}{T_C} + 1 \qquad k \in N \tag{5.1}$$

For example, in Figure 5.5(a), $T_S/T_C = 4$. Then the fifth, sixth, ... , etc. *stable periods* merge together. And there are only four *unstable areas* in the plot. Therefore, if T_C is too big, for example, bigger than $T_S/2$, we can only observe one slope in our experiment.

Second, the duration of eye movements is not constant, and T_S is randomly valued.

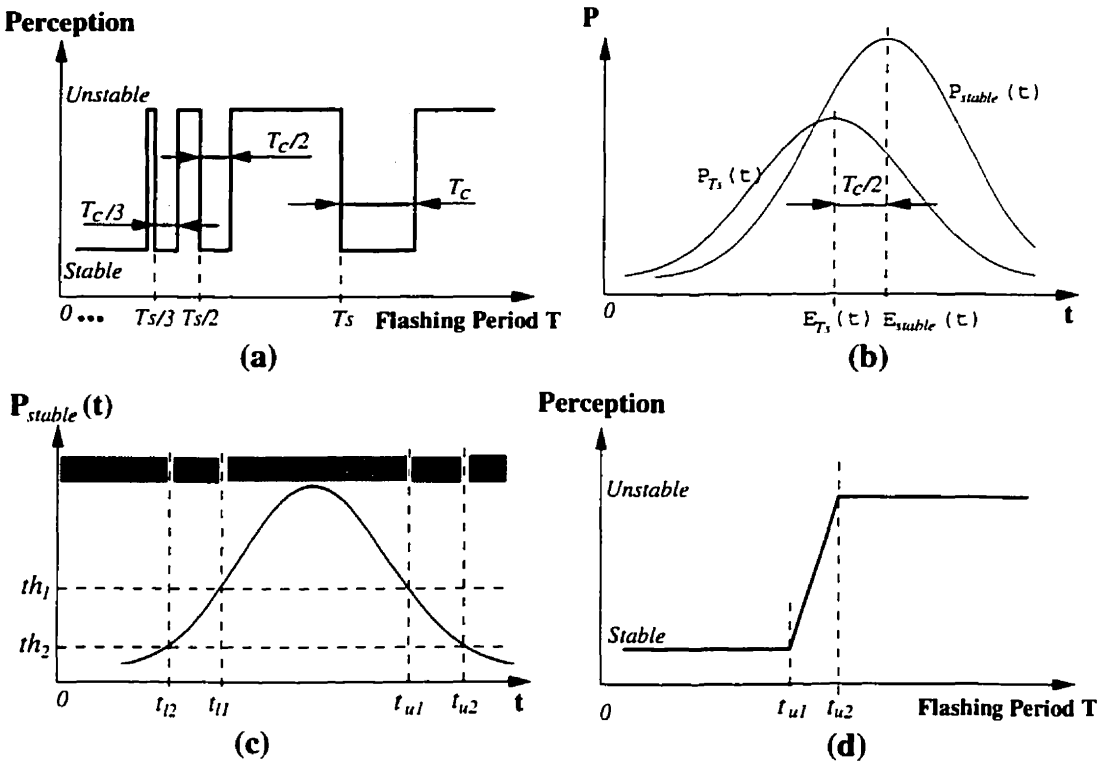

Figure 5.5: Flashing Period - Visual Perception Curve *(a) In the case that T_S is constant, there should be several "stable periods" in the Flashing Period - Perception plot. $T_S = 4T_C$, the fifth, sixth, ..., etc. "stable periods" merge together, and a subject only perceives stability at low flashing periods. (b) $P_{stable}(t)$ is the convolution of P_{T_S} and T_C. (c) Two thresholds in $P_{stable}(t)$ produce three cases of perception. If a turn-on period locates in Area-I, the subject perceive stability; if turn-on periods only locate in Area-III, the subject perceive motion; if turn-on periods only locate in Area-II, the subject reports differently in different trials. (d) Only one edge is observed in the plot if $T_C' > 0.5T_S$, and then $k' = 1$.*

This may expand the *effective value* of T_C, which we refer to T_C'. Figure 5.5(b)(c) shows the analysis of the expansion of T_C'. Without loss of generality, suppose that T_S is a random variable and the probability distribution function of T_S is $P_{T_S}(t)$, where t is the time from the beginning of the computation of the first template.

The probability of perceiving stability if a turn-on period edge is located at time t, $P_{stable}(t)$, satisfies Equation 5.2:

$$P_{stable}(t) = \int_{t-T_C}^{t} P_{T_S}(t)dt \tag{5.2}$$

Two thresholds, th_1 and th_2, exist in $P_{stable}(t)$ for each subject, as shown in Figure 5.5(c). If a turn-on period is located at time t, where $P_{stable}(t) > th_1$, it is hard for the subject to perceive movements during the test period, and the result should be "stable". If a turn-on period is located at time t, where $P_{stable}(t) < th_2$, it is hard for the subject to perceive stability during the test period, and the result should be "unstable". In other cases, the subject's answer should be different in different trials at this flashing period. Then we get the *effective value* of T_C:

$$T_C^{'} = t_{u_1} - t_{l_1} \tag{5.3}$$

Therefore, $T_C^{'}$ can be increased a lot. Additionally, we can see that T_S is constant as a special case, in which $P_{stable}(t)$ is constant from T_S to $T_S + T_C$ and $T_C^{'}$ equals to T_C.

Under the consideration that T_S is a random variable, the number of areas on the T-axis that a subject perceive instability, k', satisfies

$$\frac{T_S}{T_C^{'}} - 1 < k' <= \frac{T_S}{T_C^{'}} \qquad k \in N \tag{5.4}$$

Based on the discussion above, if $T_C^{'}$ is big enough, for example, bigger than $T_S/2$, there is only one slope in the *Flashing Period - Perception* plot, as shown in Figure 5.5(d).

5.3 Experiment B

With the same experimental setup, we did experiment B to study in what cases the subjects perceived separate movements, while in other cases they perceived the two

LED groups moving together.

5.3.1 Motivations

In experiment A, we found that if a subject could perceive movements, he or she felt that the two LED groups moved separately. However, if we changed the phase to 0°, instead of 180°, that is, let the two LED groups flash in the same mode, the subjects always reported that the two groups move together if they perceived instability. This phenomenon interested us because they relate to the strategies that humans use to define a *saccade target* during eye movements. If a subject perceives the two LED groups move separately, he/she must define only one of the two LED groups as the *saccade target*, whereas if the subject perceives concurrent movements, he/she must feel the two LED groups comprise a single large *saccade target*. By changing the phase between the two LED groups, and recording the subjects' reports on whether the objects are perceived to move separately, we could study the production of a *saccade target*.

5.3.2 Experiment setup

The setup of experiment B was as same as that of experiment A except the display patterns shown to the subjects. The subject was asked to report whether the visual pattern was moving together or separately within 5 seconds and was permitted to guess among the only two possible choices if he or she could not make a clear decision. Warm-up patterns were shown to each subject until he or she could distinguish "flicker", "moving separately", and "moving together" accurately and confidentially.

We still chose the same two LED groups, and the four LEDs of the each group were turned on and off periodically in the same mode. The flashing periods of the two LED groups were same in each trial. For each subject, we selected two flashing periods,

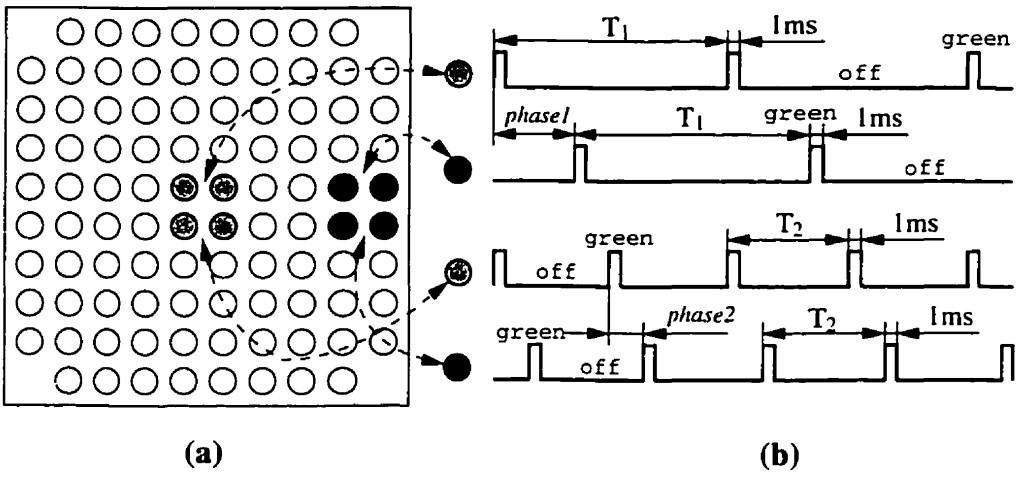

Figure 5.6: Experiment B Setup *(a) All the display patterns included two 2×2 LED groups, as shaded in this graph. The four LEDs in each of the two groups output green color and were turned on and off periodically with the same mode as shown in (b). (b) In each trial, the two LED groups flashed with the same period that is T_1 or T_2, where $T_1 = 2T_2$. The phase difference between the two groups changes in different trials, from 0° to 180°. phase1 $= 0, 1ms, 2ms, ..., T_1/2$ and phase2 $= 0, 1ms, 2ms, ..., T_2/2$. The turn-on time was always 1ms, whatever the period and the phase were.*

one of which is two times of the other, and established that the subject could perceive movements at both flashing periods. At each flashing period, we changed the *phase* between the two periodical signals of the two LED groups, from *0°* to *180°*, and kept the turn-on time as *1ms* in all trials. For example, if we chose a flashing period of *40ms*, the absolute delay of two periodic signals were *0ms*, *1ms*, *2ms*, *3ms*, ... , *20ms*. A longer flashing period involved more display patterns. Figure 5.6 shows the design of the display patterns.

We repeated each display pattern three times and showed them to each subject randomly. Four university students were selected as subjects. The lower flashing periods for these subjects were from *30ms* to *40ms*.

5.3.3 Data and results

For each subject, we calculated the normalized frequency of *detecting separate movements* (DSM) of each display pattern and generated a plot of the relationship of the normalized frequency to the phase difference. We used two ways to express the phase difference, relative value as well as absolute value. Figure 5.7 shows the typical data of this experiment.

A slope exists in each graph. The subjects perceived separate visual movements when the phase difference was great, and perceived concurrent visual movements when the phase difference was small. However, the cut-off edges were not constant. For some subjects, the cut-off edges depended on the absolute phase difference. For example, subject M. L. perceived separate visual movements when the phase difference was greater than *5ms*, whatever the flashing period was. In the *absolute phase* plot, as shown at the upper-right corner of Figure 5.7, the two curves match each other. While for some other subjects, the cut-off edges depended on the relative phase, such as S. S., who perceived separate visual movements when the relative phase difference was greater than 0.3π. In the *absolute phase* plot, the down-left graph in Figure 5.7, the cut-off edges of $T = 80ms$ is on the right of that of $T = 40ms$.

5.3.4 Experiment analysis

The subjects could perceive two kinds of visual movements during the experiment, moving together, or moving separately. The two kinds of perceptions come from how humans recognize a *saccade target*. If a subject felt that the two LED groups belonged to the same *saccade target*, when the subject loses stability of this *target*, he/she would perceive that the whole *target* composed of the two LED groups move together. If the visual system of a subject considered one LED group itself as the whole *saccade target*, when the subject loses stability of the *target*, he/she only perceived this LED

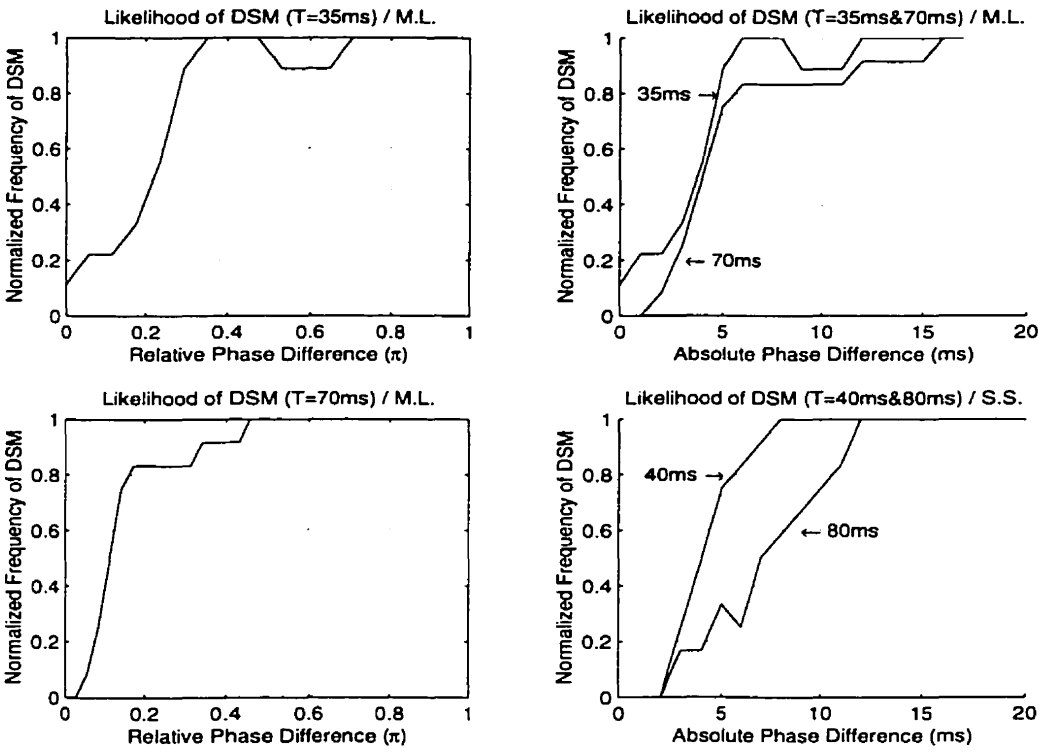

Figure 5.7: Typical Data of Experiment B *Subjects M. L. and S. S.'s data of the normalized frequency of detecting separate movements (DSM). The data was filtered with [1 1 1] and normalized to numbers between 0 and 1. The horizontal axes of the two graphs on the left represent the relative phase differences of the two LED groups. The horizontal axes of the two graphs on the right represent the time delay of the two LED groups. The vertical axes of all the graphs represent normalized frequency of detecting separate movements (DSM). The two graphs on the left and the up-right graph come from the same subject who was tested at $T = 35ms$ and $T = 70ms$. The down-right graph comes from a subject who was tested at $T = 40ms$ and $T = 80ms$.*

group moving, that is, separate visual movements were perceived.

We suppose that the *saccade target* is generated during the visual system computing the first template (period T_T). If the turn-on periods of both the two LED groups happened in T_T, the visual system counted them as an entire *saccade target*, and then

the subject could not perceive separate visual movements. Therefore, based on the model we supposed in Section 5.1, the condition of perceiving separate movements is that the *phase difference* of the two LED groups is longer than T_T. Under this condition, if only one LED group was perceived during the computation of the first template, the visual system considered this LED group itself as a separate *saccade target.*

We did observed the relationship between the *phase difference* and the perception. Figure 5.7 shows that the subjects perceived separate visual movements when the *phase difference* was greater than some value. This location of the cut-off edge should be the time period that the subject computed the first template (T_T).

However, we found two kinds of results. For some subjects, the cut-off edges at different flashing periods were similar, such as subject M. L. in Figure 5.7. While for some other subjects, the cut-off edges at different flashing periods were different, such as subject S. S. in Figure 5.7. The reason may be that some factors, such as brightness (brightness was different at different flashing period), may influence the time period of T_T. These influences may be different to different people. Trying more subjects may lead to more clear results.

Chapter 6

Conclusions

This chapter summarizes and evaluates the display system as well as the preliminary experiments. Possible improvements and further researches are suggested.

6.1 Summary of the system

A programmable display system for perceptual stability research is successfully developed and used in some preliminary perceptual stability experiments. This system consists of a display hardware and a software for design display patterns.

The display hardware is composed of a *control module*, a *storage and interface module*, and a *display module*. A *MC68HC912B32* microcontroller is used in the *control module* to control the entire hardware. A 4M-byte static RAM in the *storage and interface module* stores the frame sequences for display. Two-color LEDs are used as the display media. The system can display up to *24,576 96*-pixel, two-colored, different frames repeatedly after each download.

The distinctive feature of our system is the high time resolution and accuracy in

display. The display is refreshed at every *1ms*, and the refreshment is done in less than *0.1ms*. It realizes the millisecond-level study on perceptual stability.

A software for design display patterns was also developed. A *graphical user interface* (GUI) offers the users an amiable way to edit a display pattern quickly and distinctly.

The main limitation of this system is that it does not support real time operation. The system uses *serial communication interfaces* (SCIs) to download display patterns from a host computer to the display hardware. The display hardware saves the data and displays it. The download speed is *38,400bps*, and the download time of *1024* frames is *6.4* seconds. Because a download must be done before each new display, it does not support real time operation.

A possible improvement of this limitation is using a faster communication interface, such as *universal serial bus* (USB), instead of the traditional SCIs. The lowest data transmission speed for real time operations is *192,000bps*. We also designed some programs for simple displays (Section 4.2.5) to save the download time. These programs were used in the preliminary experiments. Additionally, if the display sequence is simple, it is quite possible to realize real time operation by compressing the original data and decompressing the package in the display hardware.

6.2 Summary of the experiments

Two preliminary experiments were done to test the hypothesis on the detailed steps during humans' eye movements involved in keeping perceptual stability.

We suppose that the human visual system computes a template before and after a saccade. The contents of the two templates depend only on the visual informations during the periods of computations. The visual system generates stable or unstable perception based on the comparison results of the two templates. A model of six

steps in this process was presented in the thesis.

We showed two two-by-two LED groups to the subjects in the preliminary experiments. The LED groups flashed periodically with the same period and turn-on time of *1ms*.

In Experiment A, we fixed the relative phase between the two LED groups as *180°*, but changed the flashing periods. We found the relationship of the subjects' perceptual stability to the flashing periods as we expected and we analyzed the results. These results supported our model.

In Experiment B we fixed the flashing period and changed the phase between the two LED groups. We recorded the subjects' unstable perceptions of "the two LED groups move together" or "the two LED groups move separately" at each phase. We found that when the phase was small, the subjects perceived "the two LED groups move together", and when the phase was large, the subjects tended to perceive "the two LED groups move separately". This observation can be explained by our model also.

In Experiment B, based on our hypothesis, the cut-off phase (absolute value) for a subject perceiving "the two LED groups move separately" instead of perceiving "the two LED groups move together", should be the period during which the subject computes the first template. We tested the subjects at two flashing periods, one of which is two times long of the other. We found that for some subjects, the computation periods were same in both cases, however, for some other subjects, the computation periods were different in the two cases. This result maybe come from the small subject set in our preliminary experiment. Because the cut-off edge is located at small phases (several milliseconds), small noise may lead to confusing results. Moreover, the computation periods may be different in different people and depend on other factors, such as brightness. We suggest using a larger subject set in a formal experiment and expect statistical analysis to bring us more clear answers.

It should be emphasized that even we only tested periodical display patterns, our hypothesis applies on non-periodical patterns as well. In the future experiments, eye-trackers are highly recommended to be used, which can record the exact time when saccades happen. And more elaborately designed display patterns may help us do more deeply researches in perceptual stability.

Bibliography

[1] Breitmeyer, Bruno G., *Visual Masking: An Integrative Approach*, New York: Oxford University Press, 1984

[2] Bridgeman, B., van der Heijden, A. and Velichkovsky, B., *A theory of visual stability across saccadic eye movements*, Behavioral and Brain Sciences, vol. 17, 247-292, 1994

[3] Brennan, K.F., *The Physics of Semiconductors*, Cambridge University Press, 1999

[4] Clark, J.J., *Spatial attention and saccadic camera motion*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 3247-3252, 1987

[5] Clark, J.J., *Spatial attention and the maintenance of representations of a robot's environment*, Proceedings of the Workshop on Perception for Mobile Agents, June 1998

[6] Clark, J.J., *Spatial attention and latencies of saccadic eye movements*, Vision Research, vol. 39, 585-602, 1999

[7] Currie, C., McConkie, G.W., Carlson-Radvansky, L.A., and Irwin, D.E., *Maintaining visual stability across saccades: Role of the saccade target object*, Human Perception and Performance Technical Report UIUC-BI-HPP-95-01, University of Illinois, Feb. 1995

[8] Hafed, Z.M., *A Neural Network Model of the Primate Saccadic System,* M.Eng. Thesis, Dept. of Electrical and Computer Engineering, McGill University, Mar. 1999

[9] Holzner, S., *ADO Programming in Visual Basic 6,* Prentice-Hall Inc., 1999

[10] Iqbal K. et al., *Visual Basic 6.0 Fundamentals,* Microsoft Corporation, 1999

[11] Irwin, D.E., Zacks, J.L., and Brown, J.S., *Visual memory and the perception of a stable visual world,* Perception and Psychophysics, vol. 47, 35-46, 1990

[12] Li, W. and Matin, L., *The influence of saccade length on the saccadic suppression of displacement detection,* Perception and Psychophysics, vol. 48, 453-458, 1990

[13] Matin, L., *Visual localization and eye movements,* Handbook of Perception and Human Performance, vol. 1, chapter 20, New York: Wiley, 1986

[14] O'Regan, J.K., *Solving the "real" mysteries of visual perception The world as an outside memory,* Canadian Journal of Psychology, vol. 46, 461-488, 1992

[15] Rensink, R., O'Regan, J.K., and Clark, J.J, *To see or not to see: the need for attention to perceive changes in scenes,* Psychological Science, vol. 8, No. 5, 368-373, 1997

[16] Shur, M., *Physics of Semiconductors,* Prentice Hall, 1990

[17] Sperry, R. W., *Neural basis of the spontaneous optokinetic response produced by visual inversion,* Journal of Comparative and Physiological Psychology, vol. 43, 482-489, 1950

[18] Tsotsos, J.K., *Analyzing vision at the complexity level,* Behavioral and Brain Sciences, vol. 13, 423-469, 1990

[19] Yarbus, A.L., *Eye Movements and Vision,* New York: Plenum, 1967

[20] *82C55A CMOS Programmable Peripheral Interface Data Sheet,* Intersil Corporation, 1999

[21] *LM4Q30TA Data Sheet,* Sharp Cooperation

[22] *LN11WP38 Data Sheet,* Sharp Cooperation

[23] *M68EVB912B32 Evaluation Board User's Manual,* Motorola Inc., 1997

[24] *MC68HC912B32 16-Bit Microcontroller Technical Summary,* Motorola Inc., 1997

[25] *Mastering Microsoft Visual Basic 6 Development,* Microsoft Corporation, 1998