

WEBCAM MOTION DETECTION USING VISUAL BASIC

by

Ilyana Bt Abdullah

Dissertation Report submitted in partial fulfillment of
the requirements for the
Bachelor of Technology (Hons)
(Information Technology)

JULY 2005

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

t

QA

76.66

.J27

2000

- 1) system programming (computer science)
- 2) Application software
- 3) IT/IS - Trends

CERTIFICATION OF APPROVAL

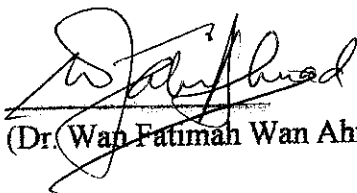
WEBCAM MOTION DETECTION USING VISUAL BASIC

by

Ilyana Bt Abdullah

**A Project Dissertation Report submitted to the
Information Technology Program
Universiti Teknologi PETRONAS
In partial fulfillment of the requirements for the
Bachelor of Technology (Hons)
(Information Technology)**

Approved by,



(Dr. Wan Fatimah Wan Ahmad)

**UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK**

November 2005

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and the original work contained herein has not been undertaken or done by unspecified sources or persons.

ILYANA ABDULLAH

ABSTRACT

Watch Eyes Motion Detection System is a project used to enhance student's interest in multimedia system development, but at the same time the project can be used as motion detection system that will help any person in doing research in motion detection purposes. This system is able to be upgraded to be a monitoring system for security purposes. The project adds the number of motion detection collection programs that available in the market. Motion detection helps a lot in daily activities that involve security and good monitoring system. Studies have shown that motion detection helps in improvement of the security system and especially in doing a research about motion. But it is not the main focus in this project because it focuses more on how motion detection works.

Watch Eyes is developed using skin tones algorithm which is use in the system to detect the skin tones that present in the captured image. The main focus of the system is to ensure that the motion can be detected only using the skin tones. Combination of several related algorithms helps in developing the skin tones algorithm. Many researches have been carried out to identify the needs of the system to make it works as expected. The development of this system done in stages, determined by the planning processes which follow the Water-Spiral model. Early analysis of the existing motion detection system falls under the earlier stage which include information gathering. Through the development of such system, it is hoped that it will help to increase student interest in computer vision that involve multimedia development and as well as to individual who interest in doing motion detection research.

ACKNOWLEDGEMENT

Alhamdulillah (Praises be to Allah) for with His blessing the project and this report to be completed. Peace and blessing be upon our Grate Prophet Muhammad Sallallahu'Alaihi Wasalam.

First and foremost, lots of gratitude and grateful thanks to Dr. Wan Fatimah Wan Ahmad (Supervisor) for sharing her knowledge and experience, and providing feedback during completing the project.

And also a special thanks to friends and VB Experts (friends and Visual Basic Experts from online discussion board through Internet) that gave advices and helps to ensure this project completed in the expected period of time. Not forgotten both parents whose gave support throughout the process of completing the project.

This appreciation also extended to those who responsible in giving helped and supported, especially for Information Technology Lecturers.

TABLE OF CONTENTS

ABSTRACT	i
ACKNOWLEDGEMENT	ii
CHAPTER 1: INTRODUCTION	
1.1 Background of Study.	1
1.2 Problem Statement	4
1.2.1 High Development Cost	4
1.2.2 Difficult to Understand How Motion Detection Works	4
1.3 Objective	5
1.4 Scope of Study	5
1.5 Feasibility Study	7
1.5.1 Difficult to ensure consistence factor	8
1.5.2 Visual Basic as New Tools	8
1.5.3 Limited Time to Research.	8
1.5.4 Slow Internet Connection	9
1.5.5 Lack Knowledge	9
1.5.6 Web Camera Compatibility	9
CHAPTER 2: LITERATURE REVIEW/THEORY	
2.1 What is Motion	10
2.2 What is Detection	11
2.3 Changes of Images	11
2.4 Research of Motion Detection History	11
2.5 Slippery Motion Research.	12
2.6 Target Set	12
2.7 Motion Analysis I	13
2.8 Automatic Detection and Matching of Moving Objects	16

	2.9 3D Camera Motion Estimation and Separation for Stereoscopic Image Sequence	17
	3.0 Motion Analysis II	19
	3.1 Image Motion	20
	3.2 Low Bit Rate Video Coding On Moving Object	21
	3.3 Comparison of Technique.	21
CHAPTER 3:	METHODOLOGY	
	3.1 System Development Life Cycle	24
	3.2 Waterfall Model	24
	3.3 Spiral Model	25
	3.4 Water-Spiral Model	26
	3.5 Tools	27
CHAPTER 4:	RESULTS AND DISCUSSION	
	4.1 Project Development	29
	4.2 Application Development.	29
	4.3 Human Engineering	31
	4.3.1 Target User Group	31
	4.3.2 Web Camera Motion Detection Interface.	32
	4.4 System Application	32
	4.4.1 Start and Stop Button	33
	4.4.2 Current Image and previous Image Box	33
	4.4.3 Fovea Box.	34
	4.4.4 Motion Detection Box	34
	4.5 Capturing Image from Web Camera	35
	4.6 Algorithm	36
CHAPTER 5:	CONCLUSION AND RECOMMENDATION	
	5.1 Conclusion	40

5.2 Recommendation	41
REFERENCES	42
APPENDIX	44

LIST OF FIGURES

Figure 2.1 : Detection Algorithm

Figure 3.1 : Waterfall Model

Figure 3.2 : Spiral Model

Figure 3.3 : Water-Spiral Model

Figure 4.1 : Main Interface (Stop Mode)

Figure 4.2 : Main Interface (Running Mode)

Figure 4.3 : Start and Stop Button

Figure 4.4 : Current Image and previous image

Figure 4.5 : Fovea image

Figure 4.6 : Motion Detection image

Figure 4.7: System architecture-image processing

ABBREVIATIONS AND NOMENCLATURES

VB 6	= Visual Basic Version 6
3D	= Three Dimension
PC's	= Personal Computers
GHz	= Gigahertz
UTP	= University Technology Petronas
STVM	= Short Term Visual Memory
SIMM	= software use for modeling body segments and associated muscles
AK & BK	= Term of process for optimal fit
2-D Projection	= Two Dimension projection
DCT	= Discrete Cosine Transform
JPEG	= Joint Photograph Expert Group
SDLC	= System Development Life Cycle
SDM	= Software Development Methodology

CHAPTER 1

INTRODUCTION

1 INTRODUCTION

Chapter 1 features the basic information of the project which consists of its background, problem statement, objectives and the scope of the study involved. This report contains of two main areas of interest that will be discussed furthermore: first, to explore the research issues involved in motion detection and second is to produce a sample of prototype of motion detection software in the development using Visual Basic 6 (VB 6).

1.1 Background of Study

Motion detection relate to the 3D movement of objects which being recorded using digital detector such as video recorder and digital camera for example web camera. It is because digital devices are good in accuracy, organization, performance and speed. For example digital video motion detection takes integration of security into the video system to new levels. Digital certainly increases accuracy and gives the ability to analyze an image in much better detail than ever before. In term of security, it should be able to ensure that the person monitoring the system does not miss anything; it can automate response and reaction to intrusion and or compromise to protected areas. It is to ensure that evidence and visual information is recorded in proper proportions.

Digital devices that use digital video motion detection systems are getting famous in the security industry. Digital video motion detection takes integration of security

into the video system to new levels. Digital is better than analog because it is compatible with today's computer system and ability. Digital certainly increases accuracy and gives us the ability to analyze an image in much better detail than ever before, but it does not afford us error-proof, hassle-free video motion detection. It is important to understand that almost everyone offers one form or another of digital video motion detection built into their equipment these days but, it doesn't mean that not all digital video motion detection is equal or applicable in all circumstances.

One of the reason is two digital video motion detection within capable units cannot be assigned to the same video image and digital video motion detection is not the sole or necessarily even the best option for all circumstances of integration and motion detection. But to identify whether the digital video motion detection is right as the selection, it is depends on the purpose of the system. The digital video motion detection is most suitable for individual movement detection because it has it ability to monitor and highlight the focus image. It is difficult to pick and select a person out of a crowd. Motion and crowd combination make motion detection more difficult.

The study of the different between normal digital motion detection such as digital camera and web camera that use in this project shows that the abilities of web camera to record and differentiate the motion will be more difficult compared to normal digital camera. It is because the ability of the web camera to differentiate between color tone of the human skin and it background is difficult. Web camera gives the image captured in orange base color. It will be difficult to identify the skin tone and the background of the image captured. So it is important to use the highest quality equipment for the application to support more high quality motion detection.

All video motion detection is based first upon change of contrast within the video signal or image. The simpler the digital video motion detection, the simpler the style of detection. Most units that are built into cameras, multiplexers, switchers and such are based upon simple contrast change within the image. Therefore any variation of light within the coverage area, e.g., shadow over the sun, lights turning on or off, headlights sweeping an area at night, will be detected as motion. More sophisticated units give us the ability to define the amount, type, size and even the direction of the

contrast change within a defined area. The size of the area and the general sensitivity to change that the unit is set for will determine what the unit sees as motion and what it sees as nuisance.

The uses of the web camera motion detection only suitable for indoor only compared to the outdoor. It is less cheap than using normal digital motion detector. High quality digital motion detector is needed for outdoor purposes because the environment keeps changing. It will be easily affected by the weather that might be the reason the detector failed to work and capture clear images. Another factor that might effect the different between indoor and outdoor area motion detection are lighting and the number of potential moving objects. It is not likely in an indoor situation that we would have extreme shadow motion such as a cloud passing over the sun.

In addition, in most indoor applications it does not have a lot of stray animals, blowing debris, rain, snow, camera shake or other such offenses that require some form of intelligent discrimination. Digital video motion detection can be added easily to any existing or new video system with the installation of a unit at the main control point. Since the installation is done inside and is performed by looping the video signals through the digital video motion detection, there is little time required. One key factor here is the overall quality of the video image that the motion detection is being attached to. Digital video motion detection will have problems when attached to images with a high degree of outside interference as caused by ground loops or radio frequency noise; it is because of the inability to discriminate between noise patterns and contrast interruptions.

Motion detection needs multiple areas of programming. Some give the user the ability to define the size of a target within an area, which can be useful when trying to discriminate between a forklift and a person. Obviously, since the forklift takes up more space within the detection area, it can be discriminated or not seen by the detection unit. It is also because the object detection becomes smaller. It will be easier to motion detection capability if the object can be highlighted and circled on

the screen for quick identification. Motion detection ability between indoor and outdoor is different because of the some form of contrast discrimination built in that involves. Normally outdoor motion detection used filtering features to identify any movement but it is more expensive than indoor. The expensive technology is usually reserved only the best for outdoor units.

Motion detection is easier when there is the ability and possibilities of tracking the movement through specific features, for example a man walking across a field and the system is able to literally track the movement through automated pan/tilt features. It is even possible that once detected and tracked by one camera to the edge of its field, it can be handed off to the next camera in succession. Actually motion needs to enter the "invisible box" of protection. Through the inter-copulation of the images, a computer is able to determine an actual three-dimensional area to be monitored. This is called "museum coverage." It is a nice concept that allows for motion around, over and under a specific item or area. Only when the item or area of coverage is touched or entered the system will detect the motion.

1.2 Problem Statement

1.2.1 High development cost

This system was developed to reduce the development cost required to have the same functional system available in the market. Motion detection system that currently available in the market sometime will be too costly. An example system present in the market is a system used in monitoring system. Some system in motion detection analysis field will also too costly. It will be difficult to some researchers, students and any related person who want to make research about motion detection to have a system that will help their research activities. This system was designed to help any individual to make research related to motion detection.

1.2.2 Difficult to understand how motion detection works

As mention before, this system was design to help people who interested to do any research about motion detection. This system is one of the solutions to

understand how motion detection works. Motion detection system development will be different with different function. Different algorithms and ways are used in motion detection. For example developer can use pixels comparison between current image and previous image to detect if motion presents.

1.3 Objectives

- 1.3.1 To improve the skin tones algorithm for motion detection.
- 1.3.2 To develop a simple system in detecting motion using VB 6.
- 1.3.3 To design simple motion detection system as far as monitoring system for future monitoring application software.

1.4 Scope of Study

The scope of the study consists of different research to find the different ways of motion detection that are used in the technology and some of it was about the research to improve the skin tones algorithm. This is the first part of the research which leads to the understanding of motion detection process. Different ways are used for different functions. Motion detection normally comes from an electrical or electromagnetic signal which then sends to the receiver as optical signal and then detected by the optical communication receiver such as web camera. Motion detection includes methods by which motion has been identified. Motion detection sensors that currently available are able easily to detect sound and produce an alarm when the motion being detected. There are motions detectors which employ cameras connected to a computer and this detect certain moving objects. The ability of the application to work properly when capturing and record the moving image will be different in different capacity of the computer. Computer with Pentium I processor will be able have slower image capturing ability compared to the Pentium III processor. More important in motion detection is the algorithm that implement in the programming development. It will identify what type of ability the systems provide. The study also expended to the motion research in science of Biomechanic, ergonomic and computer game. It is more to research how motion detection is implementing. Some of the algorithms are used in motion detection nowadays, an example:

1) Randomized Hough Transform (RHT)

Motion detection using Randomized Hough Transform is one method use to find global feature from an image for example circle, line, ellipse and etc. new algorithm arise from this algorithm is Motion Detection Randomized Hough Transform (MDRHT). This algorithm is use for solving machine vision problem like tracking of moving object. In 2D motion detection that use this algorithm, the motion object are assumed to be rigid and a number of moving object to be unknown. Actually object can be partially non rigid or distorted.

2) Detection by stereo shape matching algorithm

When a person get within range of stereoscopic vision they present a very characteristic 3D head and shoulder type of shape to the system. By searching the stereo image for large blobs with smaller blobs on the top the system can detect face without any reliance at all upon color or motion. The match is often noisy, with phanthom blobs appearing and disappearing periodically even after the gaussian smoothing (noise filtering). The stereo correspondence does actually seem to work for objects within a range less than 2 meters.

3) Region growing algorithm

In this algorithm, the edges of close image are identified or detect as one object. This algorithm normally used in object detection in an image. It is use to detect whether there are any object present in the images. To detect motion, the algorithm use by comparing the changes of object present in the images.

The main focus to this system is the development of the motion detection by using the skin tone algorithm. Researches have been done to get close to the idea and to combine the ideas of the current algorithm used in the motion detection development into the system. The study involve the RGB color management which to find out

how RGB can relate to the human skin tone. Human eyes and monitor use same color management to detect images. So due to this similarities the system act as the watch eyes of the computer.

The background or history of the motion detection started from the early year when people find the movement between atoms and particles in the environment. Since motion detection technique to broad and present in many purposes. Due to that the research about video motion detection more relate to the project. This research involves on how the image can be capture into the system using Visual Basic. The research also includes how to make the system automatically detect the device connected to the system without to make selection for the device and install the device driver.

Stereo vision also relate to the development of the project. Stereo algorithm is good in detecting whether or not the object close to the camera. Face detection use this type of algorithm. This system also use face detection algorithm. The effect performance for stereo matching involves resolution of the camera and also the camera itself whether it can calibrate.

It is also important to have temporary location for the image to be keep before next process take place. The study also includes the way how asking the system to keep the image in the temporary memory. Here the visual memory in motion detection activities used to keep temporary image in the buffer by the computer. It is important in comparing difference frames from capturing images to identify whether the motion exist. Visual memory also known as short term visual memory (STVM) uses to compare against a database of existing exemplar image within short term memory.

1.5 Feasibility of Study

Motion detection project is an interesting area if we want to learn about computer vision. Computer vision has same function like the human eye. If human have its vision, it's also the same for computer. The 'eye of the computer' has limited view

compared to human. In starting to develop this project those of the problems had occurs:

1.5.1 Difficult to ensure consistence factor (Of practicable and cost effective solution for developing the system or application).

The advantages of devising automatic systems capable of accurately locating and recognizing people are obvious if we are to have computer systems interacting unobtrusively within a social context. Many low cost video capture devices producing reasonably good quality images are now available, together with PCs having processing speeds in excess of 1 GHz. general purpose machine vision systems will at last become a practicable and cost effective solution to many previously difficult problems requiring visual identification or real time visual interaction.

Because of the large amount of computation typically required in order to process visual data, most vision systems today have been confined to industrial environments where special lighting conditions and other contextual invariants drastically reduce the scope of possible interpretations of the image only to those which are useful for achieving a particular task.

1.5.2 Visual Basic as New Tools.

The main development tools used in this project is VB 6, to learn and complete this project will be in some difficulties. Learning VB in only within four months and at the same time to develop the application will be some hard time to finish it in given time. Some features of the project need to narrow down to ensure that the project will only focus to the main objective.

1.5.3 Limited time to research (research about motion detection)

Nowadays monitoring system is very important to monitor every movement of suspected person and as long as security purposes. The development of variety of monitoring system allowed users to choose any system that suitable with

their environment. This project focuses on how motion detection analysis is done. There are many ways to analyze motion detection. Motion detection features mostly important in monitoring activities especially in security matter. So in developing this project provide alternative way to understand the motion detection clearly. But to ensure the perfect analysis to be done under this project system will be more difficult because of the limited time. Full attention is needed to ensure this project have ability to cover all the required features about motion detection.

1.5.4 Slow internet connection

Due to the slow internet connection in the UTP campus give some barriers to find information and make research for the project. Another alternative to get the information was paid for cyber café for surfing the internet.

1.5.5 Lack knowledge (About motion detection)

To design monitoring system using VB 6 will be a little bit difficult because of the lack of knowledge of that programming language. Involve in internet discussion with VB expert gave extra advantage beside of reading books and internet resources.

1.5.6 Web camera compatibility.

Using web camera as one of the selection of digital video device compare to another digital video device such as video recorder comes with new way of motion detection features. But it gives less satisfaction to the quality image recorded. Using web camera as the detection device also come with advantage which is no expensive digital video device is needed, then low cost in the development. The only problem is the quality image. In order to increase the image quality some sort of filtering is needed to reduce the noise in the image which interfere the image quality during the image capturing process.

CHAPTER 2

LITERATURE REVIEW

2 LITERATURE REVIEW

Chapter 2 contains the acknowledged findings on motion detection topic that consist of relevant theories, hypothesis, facts and information which are relevant to the objective and the research of the project.

2.1 What is Motion?

Motion means a change in the position of a body with respect to time, as measured by a particular observer in a particular frame of reference. Until the end of the 19th century, Newton's laws of motion, which he posited as axioms or postulates in his famous *Principia*, were the basis of what has since become known as classical physics. "Movement of an object in relation to its surroundings", [Zakhor, A et al, 1993] .Calculations of trajectories and forces of bodies in motion based on Newtonian or classical physics were very successful until physicists began to be able to measure and observe very fast physical phenomena. Motion normally can be detected when there are different frames occurs in every captured image. In the project, the motion (movement) detected by web camera and then being analyze by a simple program to detect whether the motion occurs. As object moved and enters the targeted area, motion then is detected. When it said movement of object relate to the surrounding, it means that the movement of object will also effect its surrounding. When capturing images, there will be some noise captured and make blur to the image. In order to get clear image, filtering is needed to remove the noise and improve the image quality.

2.2 What is Detection?

Motion detection includes methods by which motion has been identified. Motion detection sensors are available which can easily detect sound and produce an alarm. There are some motions detectors which employ cameras connected to a computer and this detects certain objects moving. The project needs to ensure the ability of detection works. "Detection is the act of detecting something; catching sight of something". [Zakhor, A et al, 1993]. The project use web camera as the motion detection sensor. Some algorithm was used to ensure that the program be able to detect motion within certain features. An example of the algorithm was face detection algorithm that detect using face and shoulder features.

2.3 Changes of images.

There is a stream of things entering into being, and time is a raging torrent; for no sooner does each thing enter our sight than it has been swept away, and another is passing in its place, and that too will be swept away. [Aurelius, 2001]. In capturing image it involve many frames. Each frame consists of different motion of images (if motions have been detected). Human eyes have good sensitivity in motion detection because the ability of human mind to captured and compared the images cross in front the eyes. It is different to digital video motion detection device (web camera that use in the project) which less sensitivity in detecting the motion especially in outside environment.

2.4 Research of motion detection history.

The research for motion detection started around the year 1900. It begun with the research about atoms, molecules, ions, bacteria and all sorts of minute organisms. Then it comes to the Brownie Motion, it is a motion of particles suspended in a solution. In the 1900s it came to encapsulate the fundamental issues at stake in early-twentieth century physical sciences: the nature and structure of matter, the relationship between statistical mechanics, kinetic theory and thermodynamics, and more broadly the validity of hypotheses and mechanical models in science. It is the

basic way of the motion research which detects motion from the movement of the atoms and molecules. [Bigg, 1870-1920]

2.5 Slippery Motion Research

As we move through the natural world, we use a variety of Visio-motor reflexes to move around objects in our path. For some time "first person view" video games have successfully simulated this movement in the virtual world, causing the user to slide around virtual objects in their path, along walls and around corners. The effect is so natural, that few players complain about it and many don't even notice. [Jacobson and Lewis, 1997]. In implementing the motion detection in the project, the understanding about the motion itself is important. Implementing the concept in the virtual world will help in increasing the information and knowledge of how motion detection can be implementing in the project and the feature present is as like in the real world.

2.6 Target Set

Proposed that when attend to a target set of moving dots, intermixed with a second set, shifts in the targets' perceived direction are attributable to relative motion computation. This computation uses the mean direction and (weighted) mean speed of all motion signals as an assumed background. It then adjusts observed motion signals to take into account the contribution of this background to the observed motion. During testing the relative speed and density of target and distractor, the following results were found; [Kim, et al, 1996]

- During examining dot patterns when the directions of target and distractor differ by as much as 135 degrees, we still observe reliable, although smaller, shifts in the perceived direction of the target.
- Increasing the density of the distractor set, increases the magnitude of the illusion.
- Speed differences between the two sets do not simply attenuate the effect and do not produce shifts that are a simple linear function of the distractor set's speed . Instead we find a small effect for slow distractors, which increases

with distractor speed and then plateaus when the distractor is traveling at the same speed as the target.

The first two results are in close accord with computation of the relative motion of the target set. The third result suggests that an additional constraint be imposed on the model: that a ceiling be placed on the influence the speed of the distractor set has on calculation of the background motion. The question of why the full speed of the background is not taken into account and why the influence of the second set plateaus as its speed is increased are possibly related to one another and to an attempt by the visual system to limit any error on its estimate of the background.

These findings have significant implications for the development of models of the integration of motion signals. There is a strong intentional component to the processing of multi-vectoral motion. Depending on whether the subject (web camera) is asked to judge the overall motion, or to judge the motion of one set, very different estimates of "perceived direction of motion" arise. This suggests that any model of motion transparency/coherency cannot produce a unique estimate of perceived direction without taking attentional factors into account. Relative motion computation is a normally effortless and fundamental component of motion processing in natural scenes. When that computation is underconstrained, as is the case of two intermixed sets of dots, attentional processes seem to direct relative motion computation. This information is not use in the project development but it is also important to understand how motion relates with its surrounding. Motion relate to the distractor speed and multi vectoral motion which will effect the movement in real environment.

2.7 Motion analysis I

Quantitative gait analysis is useful in objective documentation of walking ability as well as identifying the underlying causes for walking abnormalities in patients with cerebral palsy, stroke, head injury and other neuromuscular problems. The results of gait analysis have been shown to be useful in determining the best course of treatment in these patients. This motion detection technique is use in the medical field to detect the disability of the patient in body movement. It is one of the motion detection researches that useful in motion detection analysis for the project. Even the

objective and function of the research is different, but the information still relate to the motion detection process.

The Motion Analysis systems offer state-of-the-art, high resolution, accurate motion capture systems to acquire, analyze and display three dimensional motion data on patients while walking. The system is integrated with analog data acquisition system to enable simultaneous acquisition of force plate and electromyographic data. Clinically validated software analysis packages use OrthoTrak to analyze and display kinematic, kinetic and electromyographic data in forms that are easy to interpret. Both systems can be integrated with SIMM software which is used for modeling body segments and associated muscles. Using RealTime and SIMM, it is now possible to monitor dynamic changes in muscle length as well as muscle moment arms as the subject is walking. Motion Analysis Corporation is the undisputed worldwide leader and provider of optical motion capture systems in movement analysis, animation production, and industrial measurement and control, with over 600 systems installed worldwide. [Motion Analysis Corporation, No date].

Today's professional needs accurate physical measurement tools that do not restrict or limit patient motion. Optical motion capture offers unrestricted movements within the limitations of the patient. Whether assessing spine and back disorders in ergonomics or determining a patient's function, Motion Analysis allows to quickly and easily measure performance to determine the patient's functional outcome.

Features

- RealTime capture and digitization of immediate 3D data analysis
- Integrated kinematic or kinetic reports
- Segmental of full body reports and measurements
- Bilateral and/or unilateral testing functions

Benefits

- Identify symptom magnifier
- Objective, qualitative reports
- Ability to create and use functional normative data

- Minimal staff and space requirements

Quantitative measurement and analysis of walking is useful in the design and fabrication of orthotics and prosthetics. Orthotists and prosthetists have been reluctant to use video motion capture systems for improving the design and fitting of orthoses and prostheses due to the time necessary for processing motion data. The Motion Analysis RealTime motion capture system offers exciting possibilities in the design of orthoses and prostheses as well as interactive fitting of AK or BK prostheses for optimal fit and performance by providing real-time kinematic and kinetic data with minimal or no time delay.

Sports medicine and performance analyses go hand in hand with analyses aimed at preventing injury. While the protocols are often very similar, the analysis of sports related movements often entails analyzing a variety of highly dynamic movements. The magnitudes for angular velocities and accelerations associated with sports activities are often large and occur while translating through large fields of view.

Motion Analysis provides the tools for the Sports Medicine & Performance professional to perform accurate functional evaluations/analyses for clinical and research oriented purposes such as:

- Performance enhancement
- Injury prevention
- Return to activity
- Effects of orthopedic/athletic devices

This information provides valuable information for physicians, therapists, trainers, and coaches in designing:

- Treatment methods
- Training methods
- Rehabilitation protocols

The synergy of Biomechanics with education in Engineering Mechanics and Engineering Design has led to a number of research projects in which engineering

design principles and numerical methods are applied to biomechanical problems. Mechanics of materials and structural mechanics are applied in bone research and in rehabilitation and prevention biomechanics. Design Engineers are prepared to effectively participate in a design environment, generate conceptual design sketches and drawings, create complex design layouts, perform static and dynamic analyses, create models and prototypes, create and define complex surfaces and shapes and understand and integrate manufacturing principles into design.

The Motion Analysis RealTime system offers state-of-the-art, high resolution, accurate motion capture systems to acquire, analyze and display three dimensional motion data.

Medical robotics spans the broad areas of science and engineering to realize intelligent machineries that can be applied to clinical practice. Robotic technologies can improve existing clinical procedures as well as provide innovative new approaches to current clinical problems.

Motion Analysis works with the medical community to provide cost-effective solutions in applications such as patient positioning, virtual reality simulations for interactive medical training, and simulation of positions and forces for assessment.

The Motion Analysis RealTime system offers state-of-the-art, high resolution, accurate motion capture systems to acquire, analyze and display three dimensional motion data. [Motion Analysis Corporation, No date].

2.8 Automatic Detection And Matching Of Moving Objects

“The detection of moving objects is important in many tasks. Motion in image sequences acquired by a video camera is induced by the movement of objects in 3D scene and by camera motion. In this paper we describe automatic detection and matching algorithm when camera motion was assumed to be zero. A region growing technique in addition to change detection is used for object extraction. Object association is done after extraction and classification of moving objects. The algorithm was tested and experimental results are presented” [Prabhabar and Kadaba , 2003]

Moving object detection is an important problem in image sequence analysis. It is necessary for surveillance application, for guidance of autonomous vehicles, for efficient use video compression, for smart tracking of moving objects, and many other applications. Several methods use a change detection algorithm [Ramesh, No Date] for detection of the changed pixels. Region growing technique is used for distinguishing the change caused by the object movement and noise. Object based classification technique is used for classifying the change-detected regions into the two frames. Further, moving objects are associated in the two frames and subsequently motion parameters are estimated.

A change detection algorithm has been used for detecting the changes in two consecutive frames. Changes occur due to motion of the objects, drifts in illumination, and noise addition. To eliminate the changes caused by the noise addition and small drifts in illumination region growing technique is used. All the pixels in the change-detected image are marked with a label and given as input to the region growing algorithm. The region growing technique starts around any marked pixel and grows until all the new pixels in the neighborhood are unmarked pixels. Once the whole scan is complete all grown regions smaller than a threshold size are unmarked. This greatly reduces the unwanted regions caused by noise and small drifts in illumination. Even when the noise addition is more between two frames the noise patches can be eliminated in identification process.

Motion can be detected when if the displacement of an object from the previous frame is greater than a certain threshold value, then we can safely assume that they are different objects. The horizontal and vertical movement of the same object within two frames shows that there are movements in the scene.

2.9 3D Camera motion estimation and separation for stereoscopic image sequence.

The goal of a foreground/background extraction procedure is to estimate the camera movement (global motion) that affects both moving and stationary points of the

original scene. In many cases, global motion is the larger portion of the total apparent motion in an image sequence. Thus, a foreground/background extraction scheme may be a very efficient preprocessing step in a motion compensation coding scheme, leading to significant bit rate savings. Specifically, camera motion parameters can be effectively used to predict the stationary parts in the scene, thus saving the bandwidth needed to transmit motion vectors for the moving part. The difficulty of foreground/background extraction in monoscopic image sequence processing, relates to the low reliability of the depth estimation techniques used [Zakhor and Lari, 1993].

In order to separate background from foreground, the camera motion must be compensated. The 3D camera motion is first projected to the image plane, providing a dense motion field describing the camera movement at each point in the image plane. A camera motion compensated estimate of the second image frame is produced using the first frame and the projected camera motion information. A motion detection procedure based on thresholding follows, using the intensity difference at each pixel between the two consecutive frames. Tests have shown that a good threshold value lies between 10 and 25. The error image is then passed through a median filter to remove the small noisy regions. A procedure follows which eliminates all small regions that were not removed by the median filter. Finally, a dilation procedure was performed in order to group the small areas detected into moving objects. A block diagram of the change detection algorithm used is shown in Figure 2.1

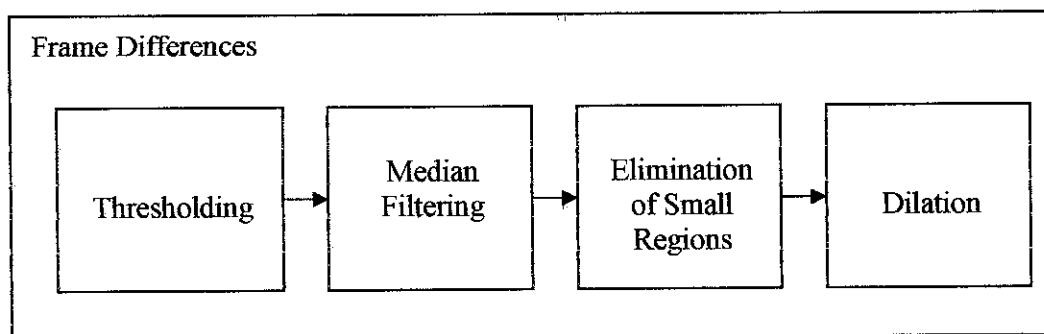


Figure 2.1: Detection Algorithm

3.0 Motion Analysis II

The versatility of computers and the precise measurements that they give have facilitated their use in the understanding of the visual world. Computer vision, the machine representation and analysis of visual information is a combination of fields spanning several disciplines such as image processing, artificial intelligence, psychology, and computer graphics. Motion provides useful information about the visual world. Dynamic scene or time-varying imagery is a sequence of images taken at closely related time i.e. two-dimensional projections (time-dependent intensity values $I(x,y,t)$) of a three dimensional physical world. Changes in these images are caused by movement of objects in the scene or the movement of the camera or both under varying lighting conditions. Intrinsic to these images is a worth of parameters which can be used to estimate motion of objects in the scene, describe 3D structures in the scene, or estimate the motion [Owusu, Willis & Jennings, No Date].

During the 1960s and 1970s enormous efforts was expended by researchers to understand static scenes. This was partly due to limited technological advancements specifically, the difficulties related to the huge amount of data produced by dynamic scenes and partly due to a paradigm adopted from the movie world an example dynamic scene imagery consists of static images (frames), thus could be understood by analyzing the frames which constitute the scene. Dynamic scene analysis differs from static scene analysis in that not only must information be extracted from each frame, but also information must be extracted from the sequence as such and integrated into a coherent whole. This integration is not just compiling of facts since change occurs in the appearance of the scene between frames due to object or camera motion. This is a non-trivial task and is further complicated where noise is present. There is also a problem where objects occlude themselves or are partially occluded by other objects.

Three main phases in motion perception: peripheral, attentive and cognitive. The peripheral process detects changes (using features such shape, motion, texture, and color) which occur between frames and directs the attentive process to it. An example to the peripheral process is a motion detector. Motion detection is the simplest of motion-related problems and usually involves the use of a single static

camera. A practical example of motion detection is surveillance which is for security purposes. The attentive process, on the other hand, tracks areas of interest in order to investigate them in more detail. There are two types of trackers which are background trackers and target trackers. For background trackers, the camera moves whilst objects are static thus effectively monitoring the location of the objects as the camera moves. Target tracking occurs when objects move relative to their background and the camera is in a static location. A complex form of tracking occurs when both camera and objects are moving. Some of the problems associated with target tracking are the detection and prediction of an object's trajectory of motion. Examples of target tracking include weather forecasting and city traffic analysis. The cognitive process relates the information derived from the attentive process to knowledge about the domain being investigated. This process focuses on deriving a 3-D description of an object from a set of 2-D projections acquired at different times of the object's motion. [Owusu, Willis & Jennings, No Date]

3.1 Image Motion

One of the most striking aspects of visual experience is the ease with which we make sense of motion in the world. The apparent ease with which motion is perceived can obscure the fact that the visual system is faced with a formidable set of problems in processing visual motion. At the physical level, movement in the world causes changes in the distribution of light across the retina. Somehow the visual system extracts information about the environment from the pattern of these changes. In order to study how this occurs, we need a description of the information available at the retina, for which we will make some simplifications. Ignoring wavelength information and adopting a ray model for light, the light distribution on the retina can be described by specifying the intensity at each point (x, y) on the retina at each moment of time, t : $I(x, y, t)$. We refer to the instantaneous light distribution on the retina $I(x, y)$ as the retinal image, which is formed by the projection of light from points in the world onto the retina. When objects in the world move relative to the observer, the projected points move within the retinal image. The movement of the projected points on the retina is referred to as the motion field [Schrater, No Date]

3.2 Low bit rate video coding on moving object.

Video is incredibly stimulating in electronic communications and multimedia applications. Researchers into very low bit-rate digital video coding face a daunting challenge of meeting two inherently conflicting requirements—reducing the transmission bit-rate as well as increasing the image quality—that are sometimes inversely proportional due to bandwidth of communication medium and computational complexity [Paul, Murshed, and Dooley, No Date].

3.3 Comparison of shot boundary detection techniques.

The following 6 methods are feasible for shot boundary detection:

1. Pixel differences: count the number of pixels that change in value more than a certain threshold. An image difference measure is calculated. Unfortunately, this technique is sensitive to camera and object motion.
2. Statistical differences : This method expand on the idea of pixel differences by breaking the image into regions and comparing statistical measures of pixels in those regions. It also generates many false positives.
3. Histograms: This method computes the gray level or color histograms of two images. If the bin wise difference between the two histograms is above a threshold, a shot boundary is assumed.
4. Compression differences: This uses differences in discrete cosine transform (DCT) coefficients of JPEG compressed frames as their measure of frame similarity and then using a form of binary search to determine the actual boundary.
5. Edge tracking: The percentage of edges that enter and exit between the two frames is computed. Shot boundaries are detected by looking for large edge change percentages.
6. Motion vectors: Motion vectors may be used to detect whether or not the shot is a zoom or a pan. Because shots with camera motion may be incorrectly classified as gradual transition, detecting zooms and pans increases the accuracy of a shot boundary detection algorithm.

Five algorithms were implemented and compared for their results. These were:

- 1) Histograms
- 2) Region histograms
- 3) Running histograms
- 4) Motion compensated pixel differences
- 5) DCT coefficient differences

The algorithms that produced better results were region based comparisons, running differences, and motion vector analysis. The algorithms compared are color histogram differences, edge change ratio, standard deviation of pixel intensities, and contrast.

1. The color histogram-based shot boundary detection algorithm is based on the fact that the color content does not change rapidly within but across shots. Thus, hard cuts and other short lasting transitions can be detected as single peaks in the time series of the differences between color histograms of contiguous frames or of frames a certain distance k apart.
2. Edge Change Ratio. This is calculated from the number of edge pixels entering and exiting in consecutive frames. The edges are calculated by the canny edge detector. Hard cuts, fades, dissolves and wipes exhibit a characteristic pattern in the ECR time series.
3. Standard deviation of pixel intensities. During video production fades are produced by a linear monotone and usually linear scaling of the pixel intensities over time. This intensity scaling is clearly visible in the time series of standard deviation of pixel intensities.
4. Edge based contrast. The basic idea of edge based contrast feature is to capture and emphasize the loss in contrast and/or sharpness to enable dissolve detection.

The comparison procedure is based on the hit rate, the miss rate and the false hits. It turned out that the performance of universal shot detection boundary algorithm based on the edge change ratio cannot justify the great computational burden. Its performance was always inferior to that of the specialized shot boundary detectors

based on color histogram differences, standard deviation of pixel intensities and edge based contrast. Fade recognition not only worked extremely reliably but also very precisely. [Tiwari and Jain, 2002]. All detection algorithms are negatively influenced by motion in the video. Future approaches should concentrate particularly on identification of local and global motion.

CHAPTER 3

METHODOLOGY

3 METHODOLOGY

Chapter 3 contains details about the description of the methodologies and procedures use to complete this project. New methodology that will be introduced later in this report is the combination of two basic of methodologies which are Waterfall model and Spiral model. This new methodology is namely as 'Water-spiral methodology'. Waterfall model used in traditional project development which it helps to provides orderly sequence of development steps.

3.1 System Development Life Cycle (SDLC)

Software engineering is the practice of using selected process techniques to improve the quality of a software development effort. This is based on the assumption, subject to endless debate and supported by patient experience, that a methodical approach to software development results in fewer defects and, therefore, ultimately provides shorter delivery times and better value. The documented collection of policies, processes and procedures used by a development team or organization to practice software engineering is called its software development methodology (SDM) or system development life cycle (SDLC).

3.2 Waterfall Model

All projects can be managed better when segmented into a hierarchy of chunks such as phases, stages, activities, tasks and steps. In system development projects, the simplest rendition of this is called the "waterfall" methodology. In this type of methodology, work is done in stages, content reviews are conducted between stages and reviews represent quality gates and decision points for continuing. The waterfall

provides an orderly sequence of development steps and helps ensure the adequacy of documentation and design reviews to ensure the quality, reliability, and maintainability of the developed software. While almost everyone these days disparages the "waterfall methodology" as being needlessly slow and cumbersome, it does illustrate a few sound principles of life cycle development. Figure 3.1 shows the Waterfall model:

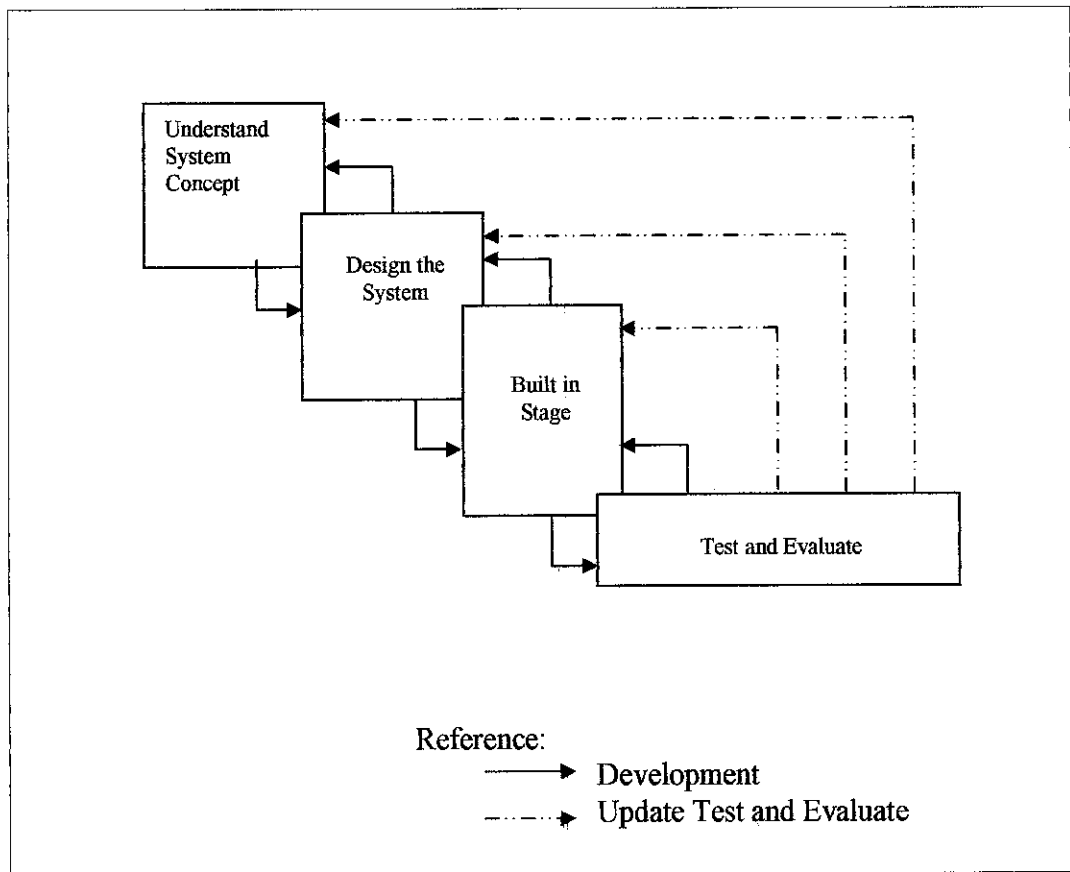


Figure 3.1: Waterfall Model

3.3 Spiral Model

The waterfall methodology offers an orderly structure for software development, demands for reduced time-to-market make its series steps inappropriate. The next evolutionary step from the waterfall is where the various steps are staged for multiple deliveries or handoffs. The ultimate evolution from the waterfall is the spiral, taking advantage of the fact that development projects work best when they are both incremental and iterative, where the team is able to start small and benefit

from enlightened trial and error along the way. Spiral methodology reflects the relationship of tasks with rapid prototyping, increased parallelism, and concurrency in design and build the activities. The spiral method should still be planned methodically, with tasks and deliverables identified for each step in the spiral. Figure 3.2 shows the Spiral Model:

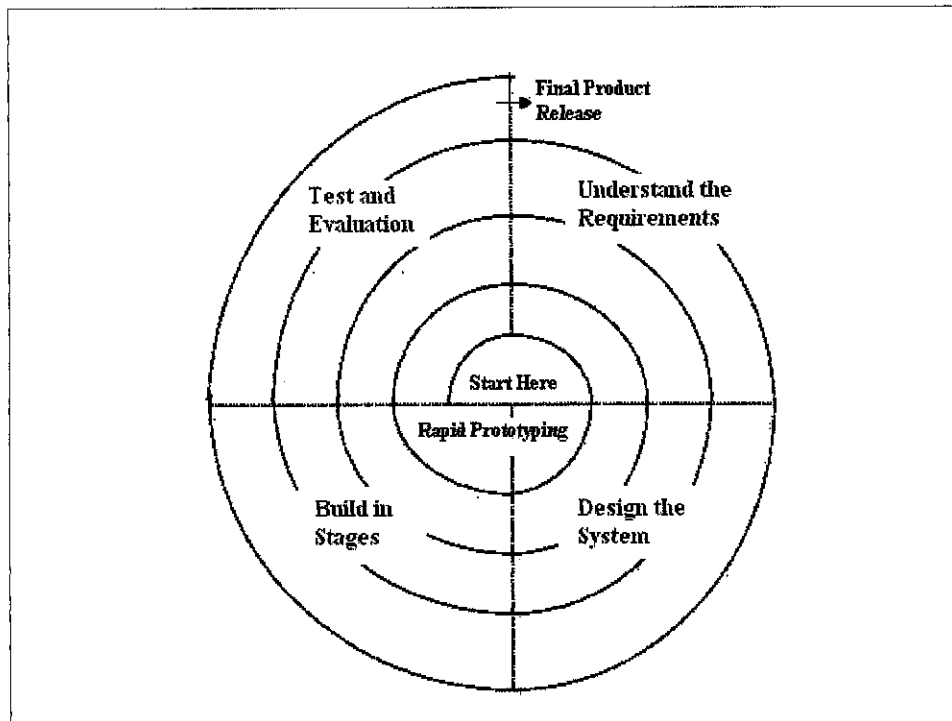


Figure 3.2 : Spiral Model

3.4 Water-Spiral Model

Water-spiral model as shown in Figure 3.3, takes advantages of both waterfall model and spiral model. With this combination the development process will be more helpful to the designer in developing their project planning. Since the waterfall provides an orderly sequence of development steps and helps ensure the adequacy of documentation, the spiral provides a lifecycle where the design, develop, test phases are repeated several times before the end product is complete. There are several different flavors of this methodology. Practitioners of this methodology like to describe this visually with a spiral diagram where each phase repeats as the spiral goes in toward the center (or sometimes out).

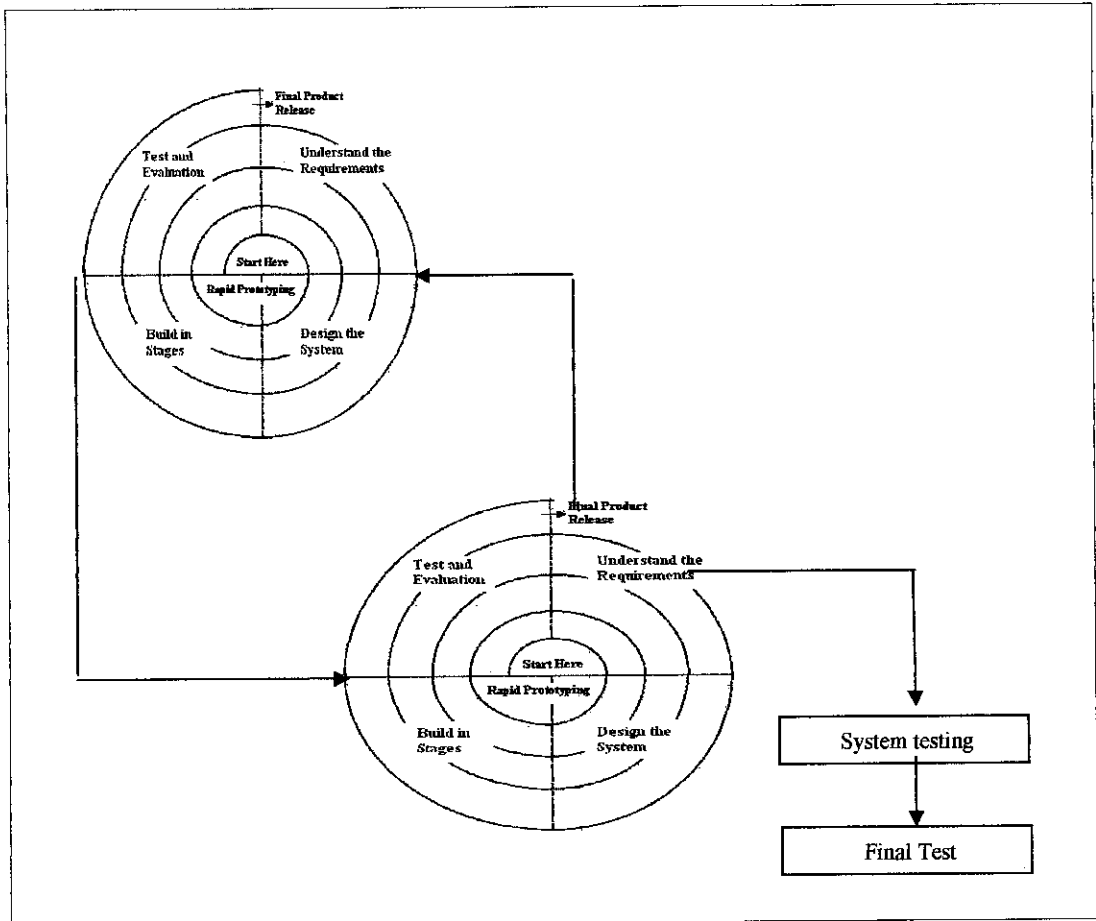


Figure 3.3 : Water-Spiral Model

In this methodology there are four basic phases involved; Understand System Concept, Design System, Built In Stage and Testing. All of these phases are repeated using the spiral model. For clear description refer to the appendix 1.

3.5 Tools

Visual Basic 6 (VB 6)

VB 6 is continuing of Microsoft's pursuit of object programming in a Rapid Application Development environment. Object oriented bring to mind the term like polymorphism, abstraction, encapsulation and inheritance. Using visual basic means primarily understanding mechanism that is used to manipulate object. An object as an a unit functionality that can be manipulated in

fundamentally the same way as objects in the real world. Object in the real world such as pencil, box, computer can be manipulated in VB 6 because it is created to be more object oriented. VB have several modes of operation that allow creation of applications. Design time is the mode used to create graphic user interface and code. Run time is the mode that actually executes the application while break mode is used to debug applications. In this project, VB 6 has been selected to be the development software which enable user to have better user interface. Web camera is use as the medium device to detect the motion which present and being detected by the system. This application is only able to be running on Microsoft Operating system because the components used in the coding have relation with the Microsoft operating system which enables the system to run well.

CHAPTER 4

RESULTS AND DISCUSSION

4 RESULTS AND DISCUSSION

Chapter 4 compiles the current findings of the project work and what will expect from the project to achieve. There have been several interesting and informative information from books and online resources that helped a lot in developing and planning the project.

4.1 Project Development

During the project development, better knowledge about VB 6 need to be considered as important thing. It is to ensure that the ability to develop the system will be easier. Discussion through the internet discussion board helps a lot in developing the project. Advices and opinions from VB experts help to design required features to the project system. This project use water-spiral model as guidance to develop the whole project system. At the moment the project has 80% completed.

4.2 Application Development

The application development of the project follows the methodology that planned before.

Phases:

- Understand System Concept
 - Define system requirement
 - Define user requirement

- Design the System
 - Software system design
 - System requirement review
 - Detailed design
 - Preliminary design review
 - Coding checking

- Built in Stage
 - Critical design review
 - Code and unit test planning

- Testing
 - Code and unit test
 - Integration and test

Above are the phases and activities that involves in the project development. The starting phase (understand system concept) consist of defining system and user requirement to identify the need of the system and user. Target user for this system is anyone that interested in motion detection analysis especially students, lecturers and researchers.

Under design phases, first is to design the software system which involves the research about motion and detection. This activity is important to get information how motion detection program can be integrate inside VB. With help and guidance with VB expert the solution for the program was found. Next to identify the system requirement that exactly important for the project. During this activity, selected requirement was reviewed from the system requirement from first phase (Define system requirement). Then detailed design is done to the system, which combines the entire requirement and the needs of the system.

Next preliminary design review (on paper draft) let the initial step in developing the system interface and coding. Since this project using the water-spiral methodology, the repeated activity is involved.

Next is the built in stage that involve critical design review and code and unit test planning. During critical design review activity, the early design is done according to the preliminary design review (on paper draft). The design then transferred to the VB environment and the coding is inserted to test whether the first stage was successful.

Then last is the testing stage which the part by part of the coding involves being tested and then the integration of the coding and the system interface is done. It may look complicated to understand the flow of the development stage because of the water-spiral combination, but the development take a lot of testing and redesign of the interface and coding to ensure that the development is successful. Stage by stage activity allows identifying the specific problem that occurs.

The latest progress done is to capture and detect any motion that present have been successfully tested. The system was successfully detected and captured motion that present. But there are some extra features is required to ensure the smoothness of the image and sensitivity of the detection. The system needs some more time in testing to ensure it is fully successfully grab it objectives.

4.3 Human Engineering

Good user interface is critical to the success of a particular system. An interface that is hard to understand will result in a high level of user errors. The project was expected to be easy to use.

4.3.1 Target user groups

The target user for this system is not limited to any group, anyone (especially students and researchers) who interested in doing research about motion and detection can use and

upgrade the system into any other application that basically involve in the same field.

4.3.2 Web camera motion detection interface

The user interface for this system has been kept on simple and easy to understand. Just give a simple as a click and be able to watch the motion that present.

4.4 System Application

Watch eyes was the name given to this application system to detect motion that presents. Figure 4.1 is the main interface of the system which consists of five main boxes available. Each box has different function relate to the motion detection process. Watch eyes system is able to detect any motion that present time to time using capturing image through the web camera connected to the system.

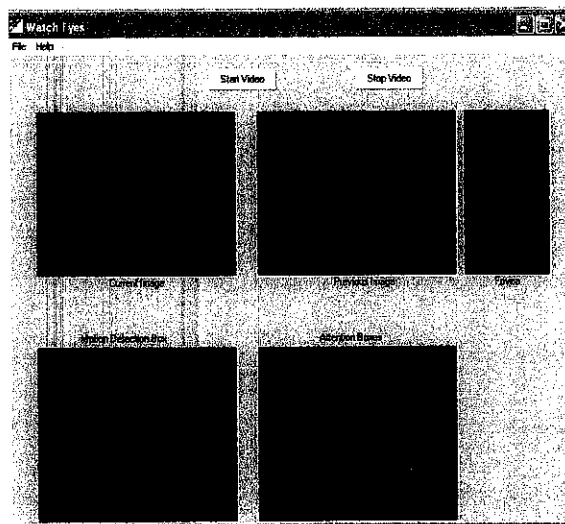


Figure 4.1 : Main Interface (Stop Mode)

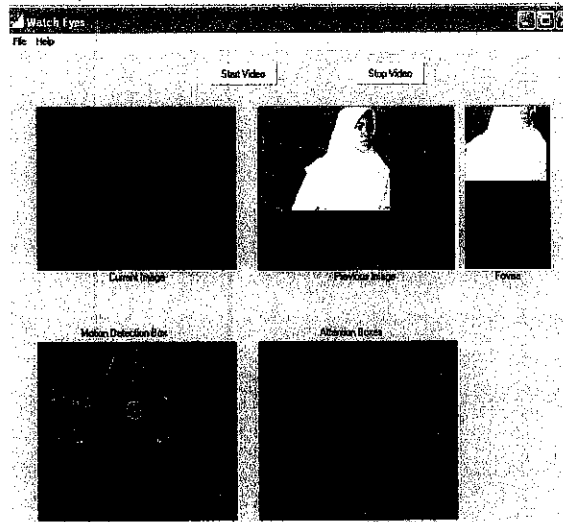


Figure 4.2 : Main Interface (Running Mode)

4.4.1 Start and Stop Button

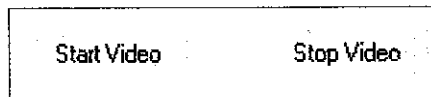


Figure 4.3 : Start and Stop Button

Two simple buttons are used in this system that enables the system to execute and done the whole process that use in the algorithm. When the start button is clicked, the system will detect the motion captured from web camera. Each box in the main interface will show the motion. Figure 4.1 shows the main interface while the system is off (stop) and figure 4.2 show while the running system.

4.4.2 Current Image and Previous Image Box

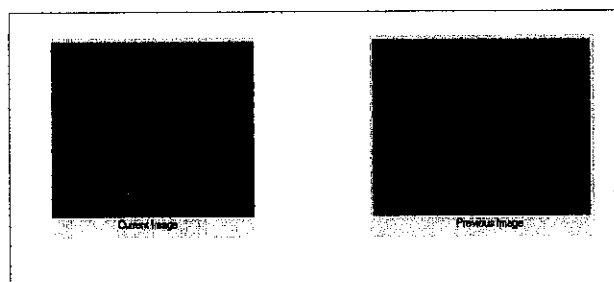


Figure 4.4 : Current Image and previous image

Current image box display the latest image that detected in the motion process and previous image box show the previous image that presents.

4.4.3 Fovea Box

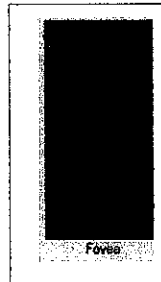


Figure 4.5 : Fovea image

Fovea box represent the focused image that detected by the system. When a huge movement part presents in the motion, system will identify the position of the movement and bring the zooming to the area. Fovea is used as one way to speed up the algorithm. It allow the scanning area to be focused and performed only at the location where the center of the area to be seemed correspond to the non-zero value (motion presents) of skin tone color at a coordinate.

This box related with the motion detection box which its select the largest moving target at any point relate to the green ball. Fovea function idea comes from the idea of the human eyes and the predatory animal that sensitive to the motion. This ability allows the function to look for vision pattern of motion characteristic of object interest. The motion detection is used to direct a smaller fovea region within the image that used for recognition of the moving object.

4.4.4 Motion Detection Box

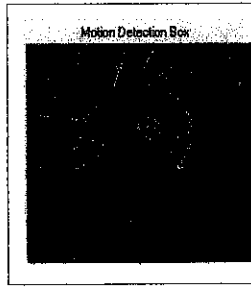


Figure 4.6 : Motion Detection image

Motion detection box relate with fovea box which is focus to the target area. This concept also used before in stereo matching algorithm which could then be used within the limit of the fovea target area. Actually it is difficult to detect motion using a web camera because of its resolution and color quality of the image is poorer than photograph quality image. Under artificial lighting, there is usually a significant amount of noise in web camera image which added to the distraction. As mentioned before the green dot in the box indicate the largest point of motion within the image. The image selected to be highlighted by the fovea is just a biggest moving part of the image but it is potentially to other criteria could be used to select from multiple attention boxes which may appear time to time.

4.5 Capturing Image From Web Camera

This system using the 32 bits VFW drivers contained within the MS Window operating system. It presents standardize programming interface to a wide variety of available video capture device. This function ensure the system able to read device connection to the system without require to install any driver or to writing any specific driver in the program.

This is the steps involves:

- 1) Creating the window

CapCreateCaptureWindow function.

32 bits number used to reference an object (window).

2) Connecting the window

hwnd function.

Handle to window function which is used to connect the window to the video driver (in this case it is use the driver installed in the web camera).

Connection made with *WM_CAP_DRIVER_CONNECT* message. It is use to bind the window with first video driver it find.

3) Retrieve capture parameter

CapSetCallbackOnFrame function.

It is a video stream to process frames during capture.

4.6 Algorithm

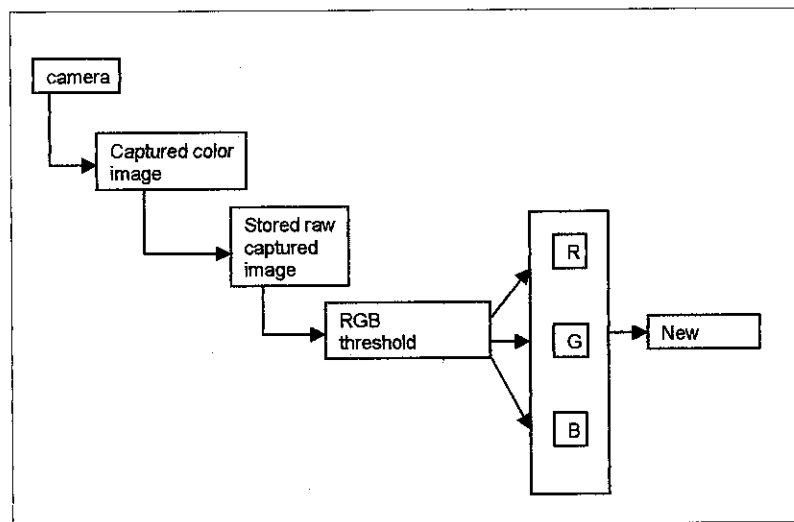


Figure 4.7: System architecture-image processing

The system starts when the button is clicked and the color image from the camera being captured. Next the raw image captured before will be stored in the memory in pixel array. Then the captured image will be pass to the RGB threshold function which then it is divided into three categories of color (red, green and blue). These three basic colors is compatible with the computer for

images. If the value of RGB is less than 0 then no motion is detected. Then the new image calculated from the separation of the intensity image is then will represent the likelihood of the skin tone color existing at a coordinate. Skin tones of different peoples are not alike. Testing has been made to three different people from Sudan, Vietnam and Malaysia (Malay). The test showed that the system is able to detect the motion together with the choices for threshold valued used. There are seemed to be a sufficient red component to the face for the algorithm to detect it as a flesh tone.

These are the step used in the algorithm:

- 1) A color image is captured from the camera and being stored in the computer in pixel array.

P_i

- 2) This array pixel containing the raw captured image which is then passed to RGB threshold function to yield three separate intensity image (red,green,blue) which is the pixel color.

$$R_a = P_i(R_{ij}) - Q_r$$

$$G_a = P_i(G_{ij}) - Q_g$$

$$B_a = P_i(B_{ij}) - Q_b$$

- **R_a, G_a and B_a** =Red, green, blue pixel intensity value at image coordinate a.
- **$P_i(R_{ij}), P_i(G_{ij})$ and $P_i(B_{ij})$** =Red, green and blue pixel intensity of raw images.
- **Q_r, Q_g and Q_b** =threshold value.

- If R_a , G_a and $B_a < 0$ then there are set to zero which mean tht there is no movement or motion detected.

3) For new image N is calculated from previous three images.

$$N_a = 2R_a - (G_a + B_a)$$

N_a = likelihood of skin tone color existing at the coordinate a (color that exist at that coordinate).

4) M image then calculated upon the skin tone likelihood and the object together with careful choice for the threshold values used within RGB. There seem to be adequate red color component to the face. It enable the algorithm to detect the color that same as flesh tone (human skin colors)

$$\text{If } N_a > 0 \text{ then } M_a = ((P_i(R_{ij}) + P_i(G_{ij}) + P_i(B_{ij}))/3$$

Threshold value are used to reduce or eliminate the bright white colors from images which can provide the shadow from light source.

5) An object having same color as human skin is not necessarily a face. To detect face within resulting image:

- Scanning is performing on various scales and at various points to detect whether face is in it or not.
- Scanning perform left to right and top to bottom. This scanning begins at the width and height to the entire image. The scanning also involves the reducing of the scan box width and height until minimum value is reached.
- To speed up the algorithm, scanning is only performing at specific location where the center area of the biggest motion detected.

6) Determine whether each sub area being scanned contain face or not done within two process.

- G image substract from the raw image P_i .
- Face template then calculated from taking the average pixel value over a number of face example image.

$$H_a = |P_a - G_a|$$

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 CONCLUSION

This project has achieved the objectives. It is able to detect motion based on the skin tone that present within the image. The development of the multimedia application such as motion detection application system will be interesting to attract people to get close to multimedia subject. Multimedia embraces many fields which each of it have different definition and use. Normally multimedia allowed the use of different media such as sound, text, graphics and images. In the project, the development also include multimedia element such as sound, images and graphics. Computer is able to determine an actual 3D area to be monitored in term of motion detection. Skin tones algorithm that use in the system is the heart of the system. It let the system works and able to detect skin tones within the image captured from the web camera. The use of the web camera is a selection to compare and test whether web camera is suitable for motion detection feature or not. During the development and system testing the result shows that the use of web camera is not very efficient to get better image quality compare to the other video device such as digital camera. But still the system can able to detect skin tones within the images because of the RGB function that presents. RGB in human eyes at the retina (fovea area) use to detect sensitivity of movement object. Computer environment also use RGB in identifying the image. Red component that available at human skin allow the system to detect whether skin tone present in the targeted image. This similarity between both human and computer give the idea to this system development.

5.2 RECOMMENDATION

Suggestion for future enhancement is to improve the system with alarm which may sound when object is detected. This is suggestion toward the monitoring security that involve motion detection. It is a nice concept that allows for motion around, over and under a specific item or area without causing a false alarm. This concept can be use in security purposes such as monitoring system. Only when the item or area of coverage is touched or entered will an alarm be announced. This is a great advancement toward providing specific coverage of items that are normally surrounded by people in motion. An example in this project is a main algorithm and technique was used in order to select motion in selected region. Since this project only focus to the motion analysis and not for security monitoring system, the alarm function mention above is not available. The expected user for the project is not limited to any group, but students and researchers who want to do any analysis with motion detection field can use part of the project and upgrade it in another purpose. Motion detection field is not limited to the detection of the unexpected motion but it is also useful in medical, sport and biochemical field that use the same concept but different functionality. In this project research, it is difficult to get information from the internet due to the slow internet connection. It will be more advantage if university can help students to resolve this problem. More can be able to carry out to for this project. Motion detection field is too broad, then more research need to be carry out to get better understand about it.

REFERENCES

- Zakhor, A and Lari, F, 1993, "Edge-Based 3-D Camera Motion Estimation with Application to Video Coding," *IEEE Trans on Image Processing*, vol. 2, pp. 48.
- Tiwari, A and Jain, N, 2002, "Video Segmentation and Video Content Analysis". Indian Institute of Technology Kanpur.
- Prabhakar, B and Damodar, V, 2001, "Automatic Detection And Matching Of Moving Objects". *CRL Technical Journal Vol.3 No.3*.
- Bigg, C, 1870 -1920, Science And The Changing Sense Of Reality.
< <http://www.mpiwg-berlin.mpg.htm>>
- Ramesh, C, Motion Detection and Estimation – Study, *Technical Report 4891/129/PC/CRL*, Bharat Electronics, Central Research Laboratory, Bangalore.
- Owusu, G, Willis, N & Jennings, N, "Motion Analysis". AI and Multimedia group, School of Computing, Leeds Metropolitan University.
- Mizianko, R, 2004, "Motion Detection Using PCA". Lecture Slide.
- Jacobson, J, and Lewis, M, 1997, " Collision Handling in Virtual Environments". *Facilitating Natural User Motion: Proceedings for the 1997 IEEE International Conference on Systems*.
<<http://usl.sis.pitt.edu/ulab/SlipperyMotion.htm>>
- Kim, J et al., 1996, "Motion Research". *Quarterly Journal of Experimental Psychology* 32, 325-333.
<<http://www.homepages.ucl.ac.uk/~smgxscd/MotionResearch.html>>
- Aurelius, M, 2001, Face Location In Unstructural Image.
<www.gutenberg.org/dirs/etext06/8durr10h.htm>.

Motion Analysis Corporation. *Motion Analysis*.

<<http://www.motionanalysis.com/applications/movement/movement.html>>

Paul, M, Murshed, M and Dooley, L,” Impact of Macroblock Classification on Low Bit Rate Video Coding Focusing on Moving Objects”, Gippsland School of Computing and Info. Tech., Monash University, Churchill Vic 3842, Australia.

Paul, R., “Local Motion Detection: Comparison Of Human Observer”. *A Dissertation for the Department of Neuroscience at the University of Pennsylvania*.

<<http://vision.psych.umn.edu/www/people/schrater/PaulSchrater.htm>>

Subramaniam, K, 1999, “Digital Signal Processing”

Postgraduate Conference Abstract.

<<http://www.ncl.ac.uk/eece/research/pgconf/pgabstract1999/subramaniam.htm>>

APPENDIX

SAMPLE OF CODING

FORM

frmMain

Option Explicit

Private Sub Form_Load()

```
Dim lpszName As String * 100
Dim lpszVer As String * 100
Dim Caps As CAPDRIVERCAPS
```

```
capGetDriverDescriptionA 0, lpszName, 100, lpszVer, 100 '
lwndC = capCreateCaptureWindowA(lpszName, WS_CAPTION Or
WS_THICKFRAME Or WS_VISIBLE Or WS_CHILD, 0, 0, 160, 120, Me.hWnd, 0)
```

```
SetWindowText lwndC, lpszName
```

```
capSetCallbackOnStatus lwndC, AddressOf MyStatusCallback
capSetCallbackOnError lwndC, AddressOf MyErrorCallback
```

```
If capDriverConnect(lwndC, 0) Then
    capDriverGetCaps lwndC, VarPtr(Caps), Len(Caps)
```

```
    If Caps.fHasDlgVideoSource = 0 Then
        mnuSource.Enabled = False
    End If
```

```
    If Caps.fHasDlgVideoFormat = 0 Then
        mnuFormat.Enabled = False
    End If
```

```
    If Caps.fHasDlgVideoDisplay = 0 Then
        mnuDisplay.Enabled = False
    End If
```

```
capPreviewScale lwndC, True
```

```
capPreviewRate lwndC, 66
```

```
capPreview lwndC, True
```

```
ResizeCaptureWindow lwndC
```

```
End If
```

End Sub

Private Sub Form_Unload(Cancel As Integer)

```
capSetCallbackOnError lwndC, vbNull
capSetCallbackOnStatus lwndC, vbNull
capSetCallbackOnYield lwndC, vbNull
capSetCallbackOnFrame lwndC, vbNull
capSetCallbackOnVideoStream lwndC, vbNull
capSetCallbackOnWaveStream lwndC, vbNull
capSetCallbackOnCapControl lwndC, vbNull
```

End Sub

Private Sub mnuAllocate_Click()

```
Dim sFile As String * 250
Dim lSize As Long

lSize = 1000000
sFile = "C:\TEMP.AVI"
capFileSetCaptureFile lwndC, sFile
capFileAlloc lwndC, lSize
```

End Sub

Private Sub mnuAlwaysVisible_Click()

```
mnuAlwaysVisible.Checked = Not (mnuAlwaysVisible.Checked)

If mnuAlwaysVisible.Checked Then
    SetWindowPos Me.hWnd, HWND_TOPMOST, 0, 0, 0, 0, SWP_NOSIZE Or
SWP_NOMOVE
Else 'MNUALWAYSVISIBLE.CHECKED = FALSE/0
    SetWindowPos Me.hWnd, HWND_NOTOPMOST, 0, 0, 0, 0, SWP_NOSIZE Or
SWP_NOMOVE
End If
```

End Sub

Private Sub mnuCompression_Click()

```
capDlgVideoCompression lwndC
```

End Sub

Private Sub mnuCopy_Click()

```
capEditCopy lwndC
```

End Sub

Private Sub mnuDisplay_Click()

```

capDlgVideoDisplay lwndC

End Sub

Private Sub mnuExit_Click()

    Unload Me

End Sub

Private Sub mnuFormat_Click()

    capDlgVideoFormat lwndC
    ResizeCaptureWindow lwndC

End Sub

Private Sub mnuPreview_Click()

    frmMain.StatusBar.SimpleText = vbNullString
    mnuPreview.Checked = Not (mnuPreview.Checked)
    capPreview lwndC, mnuPreview.Checked

End Sub

Private Sub mnuScale_Click()

    mnuScale.Checked = Not (mnuScale.Checked)
    capPreviewScale lwndC, mnuScale.Checked

    If mnuScale.Checked Then
        SetWindowLong lwndC, GWL_STYLE, WS_THICKFRAME Or WS_CAPTION
    Or WS_VISIBLE Or WS_CHILD
        Else 'MNUSCALECHECKED = FALSE/0
        SetWindowLong lwndC, GWL_STYLE, WS_BORDER Or WS_CAPTION Or
    WS_VISIBLE Or WS_CHILD
        End If

    ResizeCaptureWindow lwndC

End Sub

Private Sub mnuSelect_Click()

    frmSelect.Show vbModal, Me

End Sub

Private Sub mnuSource_Click()

    capDlgVideoSource lwndC

End Sub

Private Sub mnuStart_Click()

```



```
Dim sFileName As String
Dim CAP_PARAMS As CAPTUREPARMS

capCaptureGetSetup hwndC, VarPtr(CAP_PARAMS), Len(CAP_PARAMS)

CAP_PARAMS.dwRequestMicroSecPerFrame = (1 * (10 ^ 6)) / 30 ' 30 Frames per
second
CAP_PARAMS.fMakeUserHitOKToCapture = True
CAP_PARAMS.fCaptureAudio = False

capCaptureSetSetup hwndC, VarPtr(CAP_PARAMS), Len(CAP_PARAMS)

sFileName = "C:\myvideo.avi"

capCaptureSequence hwndC ' Start Capturing!
capFileSaveAs hwndC, sFileName ' Copy video from swap file into a real file.

End Sub
```

frmTesting

Option Explicit

Private RoddersMotion As New ClassMotion

Private Sub cmdCapture_Click()

 RodderyMotion.Vision_VFWstart picCurrentImage
 timMotion.Enabled = True

End Sub

Private Sub cmdStop_Click()

 timMotion.Enabled = False
 RodderyMotion.Vision_VFWstop

End Sub

Private Sub cmdTrack_Click()

 RodderyMotion.Tracking = Not RodderyMotion.Tracking

End Sub

Private Sub Form_Unload(Cancel As Integer)

 cmdStop_Click

End Sub

Private Sub mnuAbout_Click()

 frmAbout.Show 1

End Sub

Private Sub mnuExit_Click()

End
 cmdStop_Click

End Sub

Private Sub mnuFormat_Click()

 RodderyMotion.Vision_VFWFormatDialog

End Sub

Private Sub record_Click()

 frmAbout.Show vbModal, Me
 frmMain.Show vbModal, Me

End Sub

Private Sub timMotion_Timer()

Dim x As Single

Dim y As Single

RoddersMotion.Vision_Motion picCurrentImage, picPreviousImage, picResult
RoddersMotion.Vision_CentreOfMotion picResult, x, y, picCurrentImage, picTargets
RoddersMotion.showFovea picCurrentImage, picFovea

lblTracking.Visible = RoddersMotion.Tracking

If (RoddersMotion.Tracking) Then

lblVelocityX.Caption = RoddersMotion.velocity_x

lblVelocityY.Caption = RoddersMotion.velocity_y

End If

End Sub

MODULE

Option Explicit

Public Type RGBthingy

Value As Long

End Type

Public Type RGBpoint

Red As Byte

Green As Byte

Blue As Byte

End Type

Public Const SWP_NOZORDER As Long = &H4&

Public Const SWP_NOSENDCHANGING As Long = &H400& /* Don't send
WM_WINDOWPOSCHANGING */

Public Const HWND_BOTTOM As Long = 1&

Public Const SM_CYCAPTION As Long = 4

Public Const SM_CXBORDER As Long = 5

Public Const SM_CYBORDER As Long = 6

Public Const SM_CYMENU As Long = 15

Public Const SM_CXEDGE As Long = 45

Public Const SM_CYEDGE As Long = 46

Public Declare Function ShellAbout Lib "shell32" Alias "ShellAboutA" _

(ByVal hWnd As Long, _

ByVal szApp As String, _

ByVal szOtherStuff As String, _

ByVal hIcon As Long) As Long

Public Declare Function SetWindowTextAsLong Lib "user32" Alias "SetWindowTextA"

(ByVal hWnd As Long, ByVal LPCSTR As Long) As Long ' C BOOL

Public Declare Function GetSystemMetrics Lib "user32" (ByVal nIndex As Long) As Long

Public Declare Function SetParent Lib "user32" (ByVal hWndChild As Long, ByVal

hWndNewParent As Long) As Long

Public Declare Function SetWindowPos Lib "user32" (ByVal hWnd As Long, ByVal

hWndInsertAfter As Long, ByVal x As Long, ByVal y As Long, ByVal cx As Long, ByVal

cy As Long, ByVal wFlags As Long) As Long

Public Declare Function DestroyWindow Lib "user32" (ByVal hWnd As Long) As Long 'C
BOOL

Public Declare Function GetDiskFreeSpace Lib "kernel32" Alias "GetDiskFreeSpaceA" _

(ByVal lpRootPathName As String, _

lpSectorsPerCluster As Long, _

lpBytesPerSector As Long, _

lpNumberOfFreeClusters As Long, _

lpTotalNumberOfClusters As Long) As Long 'C BOOL

Public Declare Function GlobalAlloc Lib "kernel32" (ByVal wFlags As Long, ByVal dwBytes As Long) As Long
Public Declare Function GlobalFree Lib "kernel32" (ByVal hMem As Long) As Long
Public Declare Function GlobalLock Lib "kernel32" (ByVal hMem As Long) As Long
Public Declare Function GlobalUnlock Lib "kernel32" (ByVal hMem As Long) As Long

Public Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory" (lpvDest As Any, lpvSource As Any, ByVal cbCopy As Long)

Public Declare Function lStrCpy Lib "kernel32" Alias "lstrcpyA" (ByVal lpStringDest As Long, ByVal lpStringSrc As Long) As Long
Public Declare Sub CopyPTRtoANY Lib "kernel32.dll" Alias "RtlMoveMemory" (ByRef Dest As Any, ByVal PtrSrc As Long, ByVal length As Long)
Public Declare Sub CopyPTRtoLONG Lib "kernel32.dll" Alias "RtlMoveMemory" (ByRef LONGDest As Long, ByVal PtrSrc As Long, ByVal length As Long)

Public Declare Function BitBlt Lib "gdi32" (ByVal hDestDC As Long, ByVal x As Long, ByVal y As Long, ByVal nWidth As Long, ByVal nHeight As Long, ByVal hSrcDC As Long, ByVal xSrc As Long, ByVal ySrc As Long, ByVal dwRop As Long) As Long
Public Declare Function CreateBitmap Lib "gdi32" (ByVal nWidth As Long, ByVal nHeight As Long, ByVal nPlanes As Long, ByVal nBitCount As Long, lpBits As Any) As Long
Public Declare Function SetBkColor Lib "gdi32" (ByVal hdc As Long, ByVal crColor As Long) As Long
Public Declare Function SelectObject Lib "gdi32" (ByVal hdc As Long, ByVal hObject As Long) As Long
Public Declare Function CreateCompatibleBitmap Lib "gdi32" (ByVal hdc As Long, ByVal nWidth As Long, ByVal nHeight As Long) As Long
Public Declare Function CreateCompatibleDC Lib "gdi32" (ByVal hdc As Long) As Long
Public Declare Function DeleteDC Lib "gdi32" (ByVal hdc As Long) As Long
Public Declare Function DeleteObject Lib "gdi32" (ByVal hObject As Long) As Long

Public Const GMEM_MOVEABLE As Integer = &H2&
Public Const GMEM_SHARE As Integer = &H2000&
Public Const GMEM_ZEROINIT As Integer = &H40&

'VFW stuff

Public Const WM_USER As Integer = 1024
Public Const WM_CAP_SET_OVERLAY As Integer = WM_USER + 51
Public Const WM_CAP_SEQUENCE As Integer = WM_USER + 62
Public Const WM_CAP_SINGLE_FRAME_OPEN As Integer = WM_USER + 70
Public Const WM_CAP_SINGLE_FRAME_CLOSE As Integer = WM_USER + 71
Public Const WM_CAP_SINGLE_FRAME As Integer = WM_USER + 72
Public Const DRV_USER As Integer = &H4000
Public Const DVM_DIALOG As Integer = DRV_USER + 100

Public Declare Function capCreateCaptureWindow Lib "avicap32.dll" Alias "capCreateCaptureWindowA" (ByVal a As String, ByVal b As Long, ByVal c As Integer, ByVal d As Integer, ByVal e As Integer, ByVal f As Integer, ByVal g As Long, ByVal h As Integer) As Long

Public Declare Function SendMessageAsLong Lib "user32" Alias "SendMessageA" _
 (ByVal hWnd As Long, _
 ByVal wParam As Long, _
 ByVal lParam As Long) As Long

CLASS MODULE

Option Explicit

Private hwndc As Long
Private Const SRCCOPY = &HCC0020
Private Const SRCINVERT = &H660046

Private prev_image As StdPicture
Private Const MotionResolution As Integer = 20
Private motion(MotionResolution + 1, MotionResolution + 1, 2) As Boolean

Public NoOfAttentionBoxes As Integer
Private attentionBox(100, 4) As Integer
Private RegionArea As Integer

Private attention_x As Integer
Private attention_y As Integer

Public Tracking As Boolean
Private prev_tracking As Boolean
Private track_x As Integer
Private track_y As Integer
Public velocity_x As Single
Public velocity_y As Single

Private Const TemplateSize As Integer = 10
Private template(TemplateSize + 1, TemplateSize + 1) As Byte

Private rgbsource As RGBthingy
Private rgbdest As RGBpoint

Public Sub Vision_VFWstart(canvas As PictureBox)

 Dim temp As Long

 hwndc = capCreateCaptureWindow("Rodney Vision", WS_CHILD Or WS_VISIBLE, 0,
0, 320, 240, canvas.hWnd, 0)

 If (hwndc > 0) Then

 temp = SendMessage(hwndc, WM_CAP_DRIVER_CONNECT, 0, 0)

 temp = SendMessage(hwndc, WM_CAP_SET_PREVIEW, 1, 0)

 temp = SendMessage(hwndc, WM_CAP_SET_PREVIEWRATE, 30, 0)

 Else 'NOT (HWNDC...

 MsgBox ("Can't open capture window")

 End If

End Sub

```
Public Sub Vision_VFWFormatDialog()
```

```
    Dim temp As Long
```

```
    temp = SendMessage(hwndc, WM_CAP_DLG_VIDEOFORMAT, 0&, 0&)
```

```
End Sub
```

```
Public Sub Vision_VFWgrab(destination As PictureBox)
```

```
    Dim temp As Long
```

```
    temp = SendMessageAsLong(hwndc, WM_CAP_GRAB_FRAME, 0&, 0&)
```

```
    temp = SendMessage(hwndc, WM_CAP_EDIT_COPY, 1, 0)
```

```
    destination.Picture = Clipboard.GetData
```

```
End Sub
```

```
Public Sub Vision_CentreOfMotion(canvas As PictureBox, ByRef cx As Single, ByRef cy  
As Single, inputImage As PictureBox, targets As PictureBox)
```

```
    Dim x As Integer
```

```
    Dim y As Integer
```

```
    Dim sx As Integer
```

```
    Dim sy As Integer
```

```
    Dim p As Long
```

```
    Dim p2 As Long
```

```
    Dim tot As Double
```

```
    Dim x2 As Integer
```

```
    Dim y2 As Integer
```

```
    Dim rc As Long
```

```
    Dim surrounding As Integer
```

```
    If (Not Tracking) Then
```

```
        x2 = 0
```

```
        tot = 0
```

```
        sx = canvas.ScaleWidth / MotionResolution
```

```
        sy = canvas.ScaleHeight / MotionResolution
```

```
        For x = sx To canvas.ScaleWidth - 1 Step sx
```

```
            y2 = 0
```

```
            For y = sy To canvas.ScaleHeight - 1 Step sy
```

```
                p = canvas.Point(x, y)
```

```
                rgbsource.Value = p
```

```
                CopyMemory rgbdest, rgbsource, 3
```

```
                motion(x2, y2, 1) = 0
```

```
                If (rgbdest.Red > 10) And (rgbdest.Green > 10) And (rgbdest.Blue > 10) Then
```

```
                    motion(x2, y2, 0) = True
```

```
                Else 'NOT (RGBDEST.RED...
```

```
                    motion(x2, y2, 0) = False
```

```
                End If
```

```
                y2 = y2 + 1
```

```
            Next y
```

```
            x2 = x2 + 1
```

```
        Next x
```

```

    getAttentionBoxes attention_x, attention_y
    showAttentionBoxes inputImage, targets

    canvas.FillColor = RGB(0, 255, 0)
    canvas.FillStyle = 0
    canvas.Circle ((attention_x / MotionResolution * canvas.ScaleWidth), (attention_y /
MotionResolution * canvas.ScaleHeight)), sx

    cx = attention_x / MotionResolution
    cy = attention_y / MotionResolution

End If

End Sub

Public Sub Vision_VFWstop()

    Dim temp As Long

    temp = SendMessageAsLong(hwndc, WM_CAP_DRIVER_DISCONNECT, 0&, 0&)

End Sub

Public Sub Vision_Motion(inputImage As PictureBox, backgroundImage As PictureBox,
motionImage As PictureBox)

    Dim rc As Long
    Static firstCall As Integer
    Static t As Integer
    Dim x As Integer
    Dim y As Integer

    Vision_VFWgrab inputImage

    If (Not Tracking) Then

        If (firstCall = 0) Then
            firstCall = 1
            Set prev_image = inputImage.Picture
        End If

        t = t + 1
        If (t > 1) Then
            Set backgroundImage.Picture = prev_image
            Set prev_image = inputImage.Picture
            t = 0
        End If

        rc = BitBlt(motionImage.hdc, 0, 0, inputImage.ScaleWidth, inputImage.ScaleHeight,
backgroundImage.hdc, 0, 0, SRCCOPY)
        rc = BitBlt(motionImage.hdc, 0, 0, inputImage.ScaleWidth, inputImage.ScaleHeight,
inputImage.hdc, 0, 0, SRCINVERT)

        Else NOT (NOT...

```



```

    TrackArea inputImage
End If

    prev_tracking = Tracking

End Sub

Private Sub TrackArea(inputImage As PictureBox)

    Static w As Integer
    Static h As Integer
    Static stp_x As Integer
    Static stp_y As Integer
    Static stp_x2 As Integer
    Static stp_y2 As Integer
    Static firstCall As Integer
    Dim p As Long
    Dim x As Integer
    Dim y As Integer
    Dim xx As Integer
    Dim yy As Integer
    Dim i As Integer
    Dim ox As Integer
    Dim oy As Integer
    Dim dist As Long
    Dim minDist As Long
    Dim p1 As Integer
    Dim p2 As Integer
    Static maxDifference As Long
    Static dx As Single
    Static dy As Single
    Static ticks As Long
    Dim tx As Integer
    Dim ty As Integer

    If (firstCall = 0) Then
        w = inputImage.ScaleWidth / 15
        h = inputImage.ScaleHeight / 15
        stp_x = (w * 2) / TemplateSize
        stp_y = (h * 2) / TemplateSize
        stp_x2 = stp_x * 2
        stp_y2 = stp_y * 2
        firstCall = 1
        maxDifference = TemplateSize * TemplateSize * 30
    End If

    If (Not prev_tracking) And (Tracking) Then
        track_x = attention_x / MotionResolution * inputImage.ScaleWidth
        track_y = attention_y / MotionResolution * inputImage.ScaleHeight
        dx = 0
        dy = 0
        ticks = 0
    End If

```

```

tx = track_x
ty = track_y
minDist = maxDifference
For i = 0 To 8
  Select Case i
    Case 0
      ox = tx + stp_x2
      oy = ty - stp_y2
    Case 1
      ox = tx + stp_x2
      oy = ty
    Case 2
      ox = tx + stp_x2
      oy = ty + stp_y2
    Case 3
      ox = tx
      oy = ty + stp_y2
    Case 4
      ox = tx - stp_x2
      oy = ty + stp_y2
    Case 5
      ox = tx - stp_x2
      oy = ty
    Case 6
      ox = tx - stp_x2
      oy = ty - stp_y2
    Case 7
      ox = tx
      oy = ty - stp_y2
    Case 8
      ox = tx
      oy = ty
  End Select

  dist = 0
  x = 0
  For xx = ox - w To ox + w Step stp_x
    y = 0
    For yy = oy - h To oy + h Step stp_y
      p = inputImage.Point(xx, yy)
      rgbSource.Value = p
      CopyMemory rgbdest, rgbSource, 3
      p1 = rgbdest.Red
      p2 = template(x, y)
      dist = dist + Abs(p1 - p2)

      If (i = 8) Then
        template(x, y) = rgbdest.Red
      End If

      y = y + 1
    Next yy
    x = x + 1
  Next xx

```

```

    If (dist < minDist) Then
        minDist = dist
        tx = ox
        ty = oy
    End If

Next i

dx = dx + (track_x - tx)
dy = dy + (track_y - ty)
If (ticks > 4) Then
    velocity_x = dx / 5
    velocity_y = dy / 5
    ticks = 0
    dx = 0
    dy = 0
End If
ticks = ticks + 1

track_x = tx
track_y = ty

If (prev_tracking) And (Tracking) Then
    If (minDist <> maxDifference) Then
        attention_x = track_x / inputImage.ScaleWidth * MotionResolution
        attention_y = track_y / inputImage.ScaleHeight * MotionResolution
    Else NOT (MINDIST...
        Tracking = False
    End If
End If

End Sub

Public Sub Vision_Filter(inputImage As PictureBox, colourImage As PictureBox,
motionImage As PictureBox)

    Dim rc As Long

    Vision_VFWgrab inputImage
    rc = BitBlt(motionImage.hdc, 0, 0, inputImage.ScaleWidth, inputImage.ScaleHeight,
colourImage.hdc, 0, 0, SRCCOPY)
    rc = BitBlt(motionImage.hdc, 0, 0, inputImage.ScaleWidth, inputImage.ScaleHeight,
inputImage.hdc, 0, 0, SRCINVERT)

End Sub

Private Sub getAttentionBoxes(ByRef cx As Integer, ByRef cy As Integer)

    Dim x As Integer
    Dim y As Integer
    Dim minX As Integer
    Dim minY As Integer
    Dim maxX As Integer
    Dim maxY As Integer

```

```

Dim maxRegionArea As Integer
Dim biggest As Integer

maxRegionArea = 0
biggest = 0
NoOfAttentionBoxes = 0
For x = 0 To MotionResolution - 1
  For y = 0 To MotionResolution - 1
    If (motion(x, y, 0)) And (Not motion(x, y, 1)) Then
      RegionArea = 0
      minX = x
      minY = y
      maxX = x
      maxY = y
      fillRegion x, y, 0, minX, maxX, minY, maxY
      If (RegionArea > 5) And (NoOfAttentionBoxes < 100) Then
        If (RegionArea > maxRegionArea) Then
          maxRegionArea = RegionArea
          biggest = NoOfAttentionBoxes
        End If
        attentionBox(NoOfAttentionBoxes, 0) = minX
        attentionBox(NoOfAttentionBoxes, 1) = minY
        attentionBox(NoOfAttentionBoxes, 2) = maxX
        attentionBox(NoOfAttentionBoxes, 3) = maxY
        NoOfAttentionBoxes = NoOfAttentionBoxes + 1
      End If
    End If
  Next y
Next x

If (NoOfAttentionBoxes > 0) Then
  cx = attentionBox(biggest, 0) + ((attentionBox(biggest, 2) - attentionBox(biggest, 0)) /
2)
  cy = attentionBox(biggest, 1) + ((attentionBox(biggest, 3) - attentionBox(biggest, 1)) /
2)
End If

End Sub

Private Sub showAttentionBoxes(inputImage As PictureBox, outputImage As PictureBox)

Dim minX As Integer
Dim minY As Integer
Dim maxX As Integer
Dim maxY As Integer
Dim tx As Integer
Dim ty As Integer
Dim bx As Integer
Dim by As Integer
Dim i As Integer
Dim rc As Long
Dim sx As Integer
Dim sy As Integer
Dim c As Long

```

```

outputImage.Cls
outputImage.FillStyle = 1
outputImage.DrawWidth = 1

sx = inputImage.ScaleWidth / MotionResolution
sy = inputImage.ScaleHeight / MotionResolution
For i = 0 To NoOfAttentionBoxes - 1
    minX = attentionBox(i, 0)
    minY = attentionBox(i, 1)
    maxX = attentionBox(i, 2)
    maxY = attentionBox(i, 3)

    tx = ((minX / MotionResolution) * inputImage.ScaleWidth)
    ty = ((minY / MotionResolution) * inputImage.ScaleHeight)
    bx = ((maxX / MotionResolution) * inputImage.ScaleWidth)
    by = ((maxY / MotionResolution) * inputImage.ScaleHeight)

    c = RGB(0, 255, 0)
    outputImage.Line (tx, ty)-(bx, by), c, B
Next i

```

End Sub

Public Sub showFovea(inputImage As PictureBox, foveaImage As PictureBox)

```

Dim tx As Integer
Dim ty As Integer
Dim bx As Integer
Dim by As Integer
Dim txx As Integer
Dim tty As Integer
Dim bxx As Integer
Dim byy As Integer
Dim w As Integer
Dim h As Integer
Dim rc As Long
Dim sx As Integer
Dim sy As Integer

```

foveaImage.Cls

```

w = MotionResolution / 4
h = MotionResolution / 4
tx = attention_x - w + 1
If (tx < 0) Then
    tx = 0
End If
bx = attention_x + w + 1
If (bx > MotionResolution - 1) Then
    bx = MotionResolution - 1
End If
ty = attention_y - h
If (ty < 0) Then
    attention_y = attention_y - ty
    ty = 0

```

```

End If
by = attention_y + h
If (by > MotionResolution - 1) Then
    by = MotionResolution - 1
End If

txx = tx / MotionResolution * inputImage.ScaleWidth
tyy = ty / MotionResolution * inputImage.ScaleHeight
bxx = bx / MotionResolution * inputImage.ScaleWidth
byy = by / MotionResolution * inputImage.ScaleHeight
sx = (bx - tx) / MotionResolution * inputImage.ScaleWidth
sy = (by - ty) / MotionResolution * inputImage.ScaleHeight
rc = BitBlt(fovealImage.hdc, 0, 0, sx, sy, inputImage.hdc, txx, tyy, SRCCOPY)

```

End Sub

```

Private Sub fillRegion(px As Integer, py As Integer, depth As Integer, ByRef minX As Integer, ByRef maxX As Integer, ByRef minY As Integer, ByRef maxY As Integer)

```

```

    If (motion(px, py, 0)) And (motion(px, py, 1) = 0) Then

        motion(px, py, 1) = True
        RegionArea = RegionArea + 1

        If (px < minX) Then
            minX = px
        End If
        If (px > maxX) Then
            maxX = px
        End If
        If (py < minY) Then
            minY = py
        End If
        If (py > maxY) Then
            maxY = py
        End If

        If (depth < 100) Then

            If (py > 0) Then
                fillRegion px, py - 1, depth + 1, minX, maxX, minY, maxY
            End If

            If (px < MotionResolution - 1) And (py > 0) Then
                fillRegion px + 1, py - 1, depth + 1, minX, maxX, minY, maxY
            End If

            If (px < MotionResolution - 1) Then
                fillRegion px + 1, py, depth + 1, minX, maxX, minY, maxY
            End If

            If (px < MotionResolution - 1) And (py < MotionResolution - 2) Then
                fillRegion px + 1, py + 1, depth + 1, minX, maxX, minY, maxY
            End If

```

```

If (py < MotionResolution - 1) Then
    fillRegion px, py + 1, depth + 1, minX, maxX, minY, maxY
End If

If (px > 0) Then
    fillRegion px - 1, py, depth + 1, minX, maxX, minY, maxY
End If

If (px > 0) And (py > 0) Then
    fillRegion px - 1, py - 1, depth + 1, minX, maxX, minY, maxY
End If

If (px > 0) And (py < MotionResolution - 1) Then
    fillRegion px - 1, py + 1, depth + 1, minX, maxX, minY, maxY
End If

If (px < MotionResolution - 1) And (py < MotionResolution - 1) Then
    fillRegion px + 1, py + 1, depth + 1, minX, maxX, minY, maxY
End If

    End If
End If

End Sub

```

Option Explicit

Public Const WM_USER As Integer = &H400

Public Type POINTAPI

 x As Long

 y As Long

End Type

Public Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal hWnd As Long, ByVal wParam As Long, ByVal lParam As Integer, ByVal lParam As Long) As Long

Public Declare Function SendMessageS Lib "user32" Alias "SendMessageA" (ByVal hWnd As Long, ByVal wParam As Long, ByVal lParam As Integer, ByVal lParam As String) As Long

Public Const WM_CAP_START As Integer = WM_USER

Public Const WM_CAP_GET_CAPSTREAMPTR As Integer = WM_CAP_START + 1

Public Const WM_CAP_SET_CALLBACK_ERROR As Integer = WM_CAP_START + 2

Public Const WM_CAP_SET_CALLBACK_STATUS As Integer = WM_CAP_START + 3

Public Const WM_CAP_SET_CALLBACK_YIELD As Integer = WM_CAP_START + 4

Public Const WM_CAP_SET_CALLBACK_FRAME As Integer = WM_CAP_START + 5

Public Const WM_CAP_SET_CALLBACK_VIDEOSTREAM As Integer = WM_CAP_START + 6

Public Const WM_CAP_SET_CALLBACK_WAVESTREAM As Integer = WM_CAP_START + 7

Public Const WM_CAP_GET_USER_DATA As Integer = WM_CAP_START + 8

Public Const WM_CAP_SET_USER_DATA As Integer = WM_CAP_START + 9

Public Const WM_CAP_DRIVER_CONNECT As Integer = WM_CAP_START + 10

Public Const WM_CAP_DRIVER_DISCONNECT As Integer = WM_CAP_START + 11

Public Const WM_CAP_DRIVER_GET_NAME As Integer = WM_CAP_START + 12

Public Const WM_CAP_DRIVER_GET_VERSION As Integer = WM_CAP_START + 13

Public Const WM_CAP_DRIVER_GET_CAPS As Integer = WM_CAP_START + 14

Public Const WM_CAP_FILE_SET_CAPTURE_FILE As Integer = WM_CAP_START + 20

Public Const WM_CAP_FILE_GET_CAPTURE_FILE As Integer = WM_CAP_START + 21

Public Const WM_CAP_FILE_ALLOCATE As Integer = WM_CAP_START + 22

Public Const WM_CAP_FILE_SAVEAS As Integer = WM_CAP_START + 23

Public Const WM_CAP_FILE_SET_INFOCHUNK As Integer = WM_CAP_START + 24

Public Const WM_CAP_FILE_SAVEDIB As Integer = WM_CAP_START + 25

Public Const WM_CAP_EDIT_COPY As Integer = WM_CAP_START + 30

Public Const WM_CAP_SET_AUDIOFORMAT As Integer = WM_CAP_START + 35

Public Const WM_CAP_GET_AUDIOFORMAT As Integer = WM_CAP_START + 36

Public Const WM_CAP_DLG_VIDEOFORMAT As Integer = WM_CAP_START + 41

Public Const WM_CAP_DLG_VIDEOSOURCE As Integer = WM_CAP_START + 42

Public Const WM_CAP_DLG_VIDEODISPLAY As Integer = WM_CAP_START + 43

Public Const WM_CAP_GET_VIDEOFORMAT As Integer = WM_CAP_START + 44

Public Const WM_CAP_SET_VIDEOFORMAT As Integer = WM_CAP_START + 45

Public Const WM_CAP_DLG_VIDEOCOMPRESSION As Integer = WM_CAP_START + 46

Public Const WM_CAP_SET_PREVIEW As Integer = WM_CAP_START + 50
Public Const WM_CAP_SET_OVERLAY As Integer = WM_CAP_START + 51
Public Const WM_CAP_SET_PREVIEWRATE As Integer = WM_CAP_START + 52
Public Const WM_CAP_SET_SCALE As Integer = WM_CAP_START + 53
Public Const WM_CAP_GET_STATUS As Integer = WM_CAP_START + 54
Public Const WM_CAP_SET_SCROLL As Integer = WM_CAP_START + 55

Public Const WM_CAP_GRAB_FRAME As Integer = WM_CAP_START + 60
Public Const WM_CAP_GRAB_FRAME_NOSTOP As Integer = WM_CAP_START + 61

Public Const WM_CAP_SEQUENCE As Integer = WM_CAP_START + 62
Public Const WM_CAP_SEQUENCE_NOFILE As Integer = WM_CAP_START + 63
Public Const WM_CAP_SET_SEQUENCE_SETUP As Integer = WM_CAP_START + 64
Public Const WM_CAP_GET_SEQUENCE_SETUP As Integer = WM_CAP_START + 65
Public Const WM_CAP_SET_MCI_DEVICE As Integer = WM_CAP_START + 66
Public Const WM_CAP_GET_MCI_DEVICE As Integer = WM_CAP_START + 67
Public Const WM_CAP_STOP As Integer = WM_CAP_START + 68
Public Const WM_CAP_ABORT As Integer = WM_CAP_START + 69

Public Const WM_CAP_SINGLE_FRAME_OPEN As Integer = WM_CAP_START + 70
Public Const WM_CAP_SINGLE_FRAME_CLOSE As Integer = WM_CAP_START + 71
Public Const WM_CAP_SINGLE_FRAME As Integer = WM_CAP_START + 72

Public Const WM_CAP_PAL_OPEN As Integer = WM_CAP_START + 80
Public Const WM_CAP_PAL_SAVE As Integer = WM_CAP_START + 81
Public Const WM_CAP_PAL_PASTE As Integer = WM_CAP_START + 82
Public Const WM_CAP_PAL_AUTOCREATE As Integer = WM_CAP_START + 83
Public Const WM_CAP_PAL_MANUALCREATE As Integer = WM_CAP_START + 84

Public Const WM_CAP_SET_CALLBACK_CAPCONTROL As Integer = WM_CAP_START + 85

Public Const WM_CAP_END As Integer = WM_CAP_SET_CALLBACK_CAPCONTROL

Public Type CAPDRIVERCAPS
 wDeviceIndex As Long '
 fHasOverlay As Long
 fHasDlgVideoSource As Long '
 fHasDlgVideoFormat As Long
 fHasDlgVideoDisplay As Long
 fCaptureInitialized As Long '
 fDriverSuppliesPalettes As Long '
 hVideoIn As Long '
 hVideoOut As Long '
 hVideoExtIn As Long
 hVideoExtOut As Long '
End Type

Public Type CAPSTATUS
 uiImageWidth As Long

uiImageHeight As Long
fLiveWindow As Long
fOverlayWindow As Long
fScale As Long
ptScroll As POINTAPI
fUsingDefaultPalette As Long
fAudioHardware As Long
fCapFileExists As Long
dwCurrentVideoFrame As Long
dwCurrentVideoFramesDropped As Long
dwCurrentWaveSamples As Long
dwCurrentTimeElapsedMS As Long
hPalCurrent As Long
fCapturingNow As Long
dwReturn As Long
wNumVideoAllocated As Long
wNumAudioAllocated As Long
End Type

Public Type CAPTUREPARMS
dwRequestMicroSecPerFrame As Long
fMakeUserHitOKToCapture As Long
wPercentDropForError As Long
fYield As Long
dwIndexSize As Long
wChunkGranularity As Long
fUsingDOSMemory As Long
wNumVideoRequested As Long
fCaptureAudio As Long
wNumAudioRequested As Long
vKeyAbort As Long
fAbortLeftMouse As Long
fAbortRightMouse As Long
fLimitEnabled As Long
wTimeLimit As Long
fMCIControl As Long
fStepMCIDevice As Long
dwMCICStartTime As Long
dwMCICStopTime As Long
fStepCaptureAt2x As Long
wStepCaptureAverageFrames As Long
dwAudioBufferSize As Long
fDisableWriteCache As Long
End Type

Public Type CAPINFOCHUNK
fccInfoID As Long
lpData As Long
cbData As Long
End Type

Public Type VIDEOHDR
lpData As Long
dwBufferLength As Long
dwBytesUsed As Long

```

    dwTimeCaptured As Long
    dwUser As Long
    dwFlags As Long
    dwReserved(3) As Long
End Type

'// The two functions exported by AVICap
Public Declare Function capCreateCaptureWindowA Lib
    ByVal lpszWindowName As String,
    ByVal dwStyle As Long,
    ByVal x As Long, ByVal y As Long, ByVal nWidth
As Long, ByVal nHeight As Integer, _
    ByVal hWndParent As Long, ByVal nID As Long)
As Long
Public Declare Function capGetDriverDescriptionA Lib
    ByVal wDriver As Integer,
    ByVal lpszName As String,
    ByVal cbName As Long,
    ByVal lpszVer As String,
    ByVal cbVer As Long) As Boolean

Public Const IDS_CAP_BEGIN As Integer = 300
Public Const IDS_CAP_END As Integer = 301

Public Const IDS_CAP_INFO As Integer = 401
Public Const IDS_CAP_OUTOFMEM As Integer = 402
Public Const IDS_CAP_FILEEXISTS As Integer = 403
Public Const IDS_CAP_ERRORPALOPEN As Integer = 404
Public Const IDS_CAP_ERRORPALSAVE As Integer = 405
Public Const IDS_CAP_ERRORDIBSAVE As Integer = 406
Public Const IDS_CAP_DEFAVIEXT As Integer = 407
Public Const IDS_CAP_DEFPALTEXT As Integer = 408
Public Const IDS_CAP_CANTOPEN As Integer = 409
Public Const IDS_CAP_SEQ_MSGSTART As Integer = 410
Public Const IDS_CAP_SEQ_MSGSTOP As Integer =
Public Const IDS_CAP_VIDEDITERR As Integer = 412
Public Const IDS_CAP_READONLYFILE As Integer =
Public Const IDS_CAP_WRITEERROR As Integer =
Public Const IDS_CAP_NODISKSPACE As Integer = 415
Public Const IDS_CAP_SETFILESIZE As Integer = 416
Public Const IDS_CAP_SAVEASPERCENT As Integer = 417
Public Const IDS_CAP_DRIVER_ERROR As Integer = 418
Public Const IDS_CAP_WAVE_OPEN_ERROR As Integer = 419
Public Const IDS_CAP_WAVE_ALLOC_ERROR As Integer = 420
Public Const IDS_CAP_WAVE_PREPARE_ERROR As Integer = 421
Public Const IDS_CAP_WAVE_ADD_ERROR As Integer = Public Const
IDS_CAP_WAVE_SIZE_ERROR As Integer = 423
Public Const IDS_CAP_VIDEO_OPEN_ERROR As Integer = 424
Public Const IDS_CAP_VIDEO_ALLOC_ERROR As Integer = 425

Public Function capDriverGetName(ByVal hWnd As Long, ByVal szName As Long, ByVal
wSize As Integer) As Boolean
Const YOURCONSTANTMESSAGE As Integer = 0

```

```
capDriverGetName = SendMessage(hwnd, YOURCONSTANTMESSAGE, wSize, szName)
```

```
End Function
```

```
Public Function capDriverGetVersion(ByVal hwnd As Long, ByVal szVer As Long, ByVal wSize As Integer) As Boolean
```

```
capDriverGetVersion = SendMessage(hwnd, WM_CAP_DRIVER_GET_VERSION, wSize, szVer)
```

```
End Function
```

```
Public Function capDriverGetCaps(ByVal hwnd As Long, ByVal s As Long, ByVal wSize As Integer) As Boolean
```

```
capDriverGetCaps = SendMessage(hwnd, WM_CAP_DRIVER_GET_CAPS, wSize, s)
```

```
End Function
```

```
Public Function capFileSetCaptureFile(ByVal hwnd As Long, szName As String) As Boolean
```

```
capFileSetCaptureFile = SendMessageS(hwnd, WM_CAP_FILE_SET_CAPTURE_FILE, 0, szName)
```

```
End Function
```

```
Public Function capFileGetCaptureFile(ByVal hwnd As Long, ByVal szName As Long, wSize As String) As Boolean
```

```
capFileGetCaptureFile = SendMessageS(hwnd, WM_CAP_FILE_SET_CAPTURE_FILE, wSize, szName)
```

```
End Function
```

```
Public Function capFileAlloc(ByVal hwnd As Long, ByVal dwSize As Long) As Boolean
```

```
capFileAlloc = SendMessage(hwnd, WM_CAP_FILE_ALLOCATE, 0, dwSize)
```

```
End Function
```

```
Public Function capFileSaveAs(ByVal hwnd As Long, szName As String) As Boolean
```

```
capFileSaveAs = SendMessageS(hwnd, WM_CAP_FILE_SAVEAS, 0, szName)
```

```
End Function
```

```
Public Function capFileSetInfoChunk(ByVal hwnd As Long, ByVal lpInfoChunk As Long) As Boolean
```

```
capFileSetInfoChunk = SendMessage(hwnd, WM_CAP_FILE_SET_INFOCHUNK, 0, lpInfoChunk)
```

```
End Function
```

```

Public Function capFileSaveDIB(ByVal hWnd As Long, ByVal szName As Long) As
Boolean
    capFileSaveDIB = SendMessage(hWnd, WM_CAP_FILE_SAVEDIB, 0, szName)
End Function

Public Function capEditCopy(ByVal hWnd As Long) As Boolean
    capEditCopy = SendMessage(hWnd, WM_CAP_EDIT_COPY, 0, 0)
End Function

Public Function capSetAudioFormat(ByVal hWnd As Long, ByVal s As Long, ByVal wSize
As Integer) As Boolean
    capSetAudioFormat = SendMessage(hWnd, WM_CAP_SET_AUDIOFORMAT, wSize, s)
End Function

Public Function capGetAudioFormat(ByVal hWnd As Long, ByVal s As Long, ByVal wSize
As Integer) As Long
    capGetAudioFormat = SendMessage(hWnd, WM_CAP_GET_AUDIOFORMAT, wSize,
s)
End Function

Public Function capGetAudioFormatSize(ByVal hWnd As Long) As Long
    capGetAudioFormatSize = SendMessage(hWnd, WM_CAP_GET_AUDIOFORMAT, 0,
0)
End Function

Public Function capDlgVideoFormat(ByVal hWnd As Long) As Boolean
    capDlgVideoFormat = SendMessage(hWnd, WM_CAP_DLG_VIDEOFORMAT, 0, 0)
End Function

Public Function capDlgVideoSource(ByVal hWnd As Long) As Boolean
    capDlgVideoSource = SendMessage(hWnd, WM_CAP_DLG_VIDEOSOURCE, 0, 0)
End Function

Public Function capDlgVideoDisplay(ByVal hWnd As Long) As Boolean
    capDlgVideoDisplay = SendMessage(hWnd, WM_CAP_DLG_VIDEODISPLAY, 0, 0)
End Function

Public Function capDlgVideoCompression(ByVal hWnd As Long) As Boolean

```

```
capDlgVideoCompression = SendMessage(hwnd,  
WM_CAP_DLG_VIDEOCOMPRESSION, 0, 0)
```

End Function

```
Public Function capGetVideoFormat(ByVal hwnd As Long, ByVal s As Long, ByVal wSize  
As Integer) As Long
```

```
capGetVideoFormat = SendMessage(hwnd, WM_CAP_GET_VIDEOFORMAT, wSize,  
s)
```

End Function

```
Public Function capGetVideoFormatSize(ByVal hwnd As Long) As Long
```

```
capGetVideoFormatSize = SendMessage(hwnd, WM_CAP_GET_VIDEOFORMAT, 0,  
0)
```

End Function

```
Public Function capSetVideoFormat(ByVal hwnd As Long, ByVal s As Long, ByVal wSize  
As Integer) As Boolean
```

```
capSetVideoFormat = SendMessage(hwnd, WM_CAP_SET_VIDEOFORMAT, wSize, s)
```

End Function

```
Public Function capPreview(ByVal hwnd As Long, ByVal f As Boolean) As Boolean
```

```
capPreview = SendMessage(hwnd, WM_CAP_SET_PREVIEW, f, 0)
```

End Function

```
Public Function capPreviewRate(ByVal hwnd As Long, ByVal wMS As Integer) As  
Boolean
```

```
capPreviewRate = SendMessage(hwnd, WM_CAP_SET_PREVIEWRATE, wMS, 0)
```

End Function

```
Public Function capOverlay(ByVal hwnd As Long, ByVal f As Boolean) As Boolean
```

```
capOverlay = SendMessage(hwnd, WM_CAP_SET_OVERLAY, f, 0)
```

End Function

```
Public Function capPreviewScale(ByVal hwnd As Long, ByVal f As Boolean) As Boolean
```

```
capPreviewScale = SendMessage(hwnd, WM_CAP_SET_SCALE, f, 0)
```

End Function

```
Public Function capGetStatus(ByVal hwnd As Long, ByVal s As Long, ByVal wSize As  
Integer) As Boolean
```

```

    capGetStatus = SendMessage(lwnd, WM_CAP_GET_STATUS, wSize, s)

End Function

Public Function capSetScrollPos(ByVal lwnd As Long, ByVal lpP As Long) As Boolean

    capSetScrollPos = SendMessage(lwnd, WM_CAP_SET_SCROLL, 0, lpP)

End Function

Public Function capGrabFrame(ByVal lwnd As Long) As Boolean

    capGrabFrame = SendMessage(lwnd, WM_CAP_GRAB_FRAME, 0, 0)

End Function

Public Function capGrabFrameNoStop(ByVal lwnd As Long) As Boolean

    capGrabFrameNoStop = SendMessage(lwnd, WM_CAP_GRAB_FRAME_NOSTOP, 0,
0)

End Function

Public Function capCaptureSequence(ByVal lwnd As Long) As Boolean

    capCaptureSequence = SendMessage(lwnd, WM_CAP_SEQUENCE, 0, 0)

End Function

Public Function capCaptureSequenceNoFile(ByVal lwnd As Long) As Boolean

    capCaptureSequenceNoFile = SendMessage(lwnd, WM_CAP_SEQUENCE_NOFILE, 0,
0)

End Function

Public Function capCaptureStop(ByVal lwnd As Long) As Boolean

    capCaptureStop = SendMessage(lwnd, WM_CAP_STOP, 0, 0)

End Function

Public Function capCaptureAbort(ByVal lwnd As Long) As Boolean

    capCaptureAbort = SendMessage(lwnd, WM_CAP_ABORT, 0, 0)

End Function

Public Function capCaptureSingleFrameOpen(ByVal lwnd As Long) As Boolean

    capCaptureSingleFrameOpen = SendMessage(lwnd,
WM_CAP_SINGLE_FRAME_OPEN, 0, 0)

End Function

```

```

Public Function capCaptureSingleFrameClose(ByVal hwnd As Long) As Boolean

    capCaptureSingleFrameClose = SendMessage(hwnd,
WM_CAP_SINGLE_FRAME_CLOSE, 0, 0)

End Function

Public Function capCaptureSingleFrame(ByVal hwnd As Long) As Boolean

    capCaptureSingleFrame = SendMessage(hwnd, WM_CAP_SINGLE_FRAME, 0, 0)

End Function

Public Function capCaptureGetSetup(ByVal hwnd As Long, ByVal s As Long, ByVal wSize
As Integer) As Boolean

    capCaptureGetSetup = SendMessage(hwnd, WM_CAP_GET_SEQUENCE_SETUP,
wSize, s)

End Function

Public Function capCaptureSetSetup(ByVal hwnd As Long, ByVal s As Long, ByVal wSize
As Integer) As Boolean

    capCaptureSetSetup = SendMessage(hwnd, WM_CAP_SET_SEQUENCE_SETUP,
wSize, s)

End Function

Public Function capSetMCIDeviceName(ByVal hwnd As Long, ByVal szName As Long)
As Boolean

    capSetMCIDeviceName = SendMessage(hwnd, WM_CAP_SET_MCI_DEVICE, 0,
szName)

End Function

Public Function capGetMCIDeviceName(ByVal hwnd As Long, ByVal szName As Long,
ByVal wSize As Integer) As Boolean

    capGetMCIDeviceName = SendMessage(hwnd, WM_CAP_GET_MCI_DEVICE, wSize,
szName)

End Function

Public Function capPaletteOpen(ByVal hwnd As Long, ByVal szName As Long) As
Boolean

    capPaletteOpen = SendMessage(hwnd, WM_CAP_PAL_OPEN, 0, szName)

End Function

Public Function capPaletteSave(ByVal hwnd As Long, ByVal szName As Long) As
Boolean

```



```
capPaletteSave = SendMessage(lwnd, WM_CAP_PAL_SAVE, 0, szName)
```

```
End Function
```

```
Public Function capPalettePaste(ByVal lwnd As Long) As Boolean
```

```
capPalettePaste = SendMessage(lwnd, WM_CAP_PAL_PASTE, 0, 0)
```

```
End Function
```

```
Public Function capPaletteAuto(ByVal lwnd As Long, ByVal iFrames As Integer, ByVal  
iColor As Long) As Boolean
```

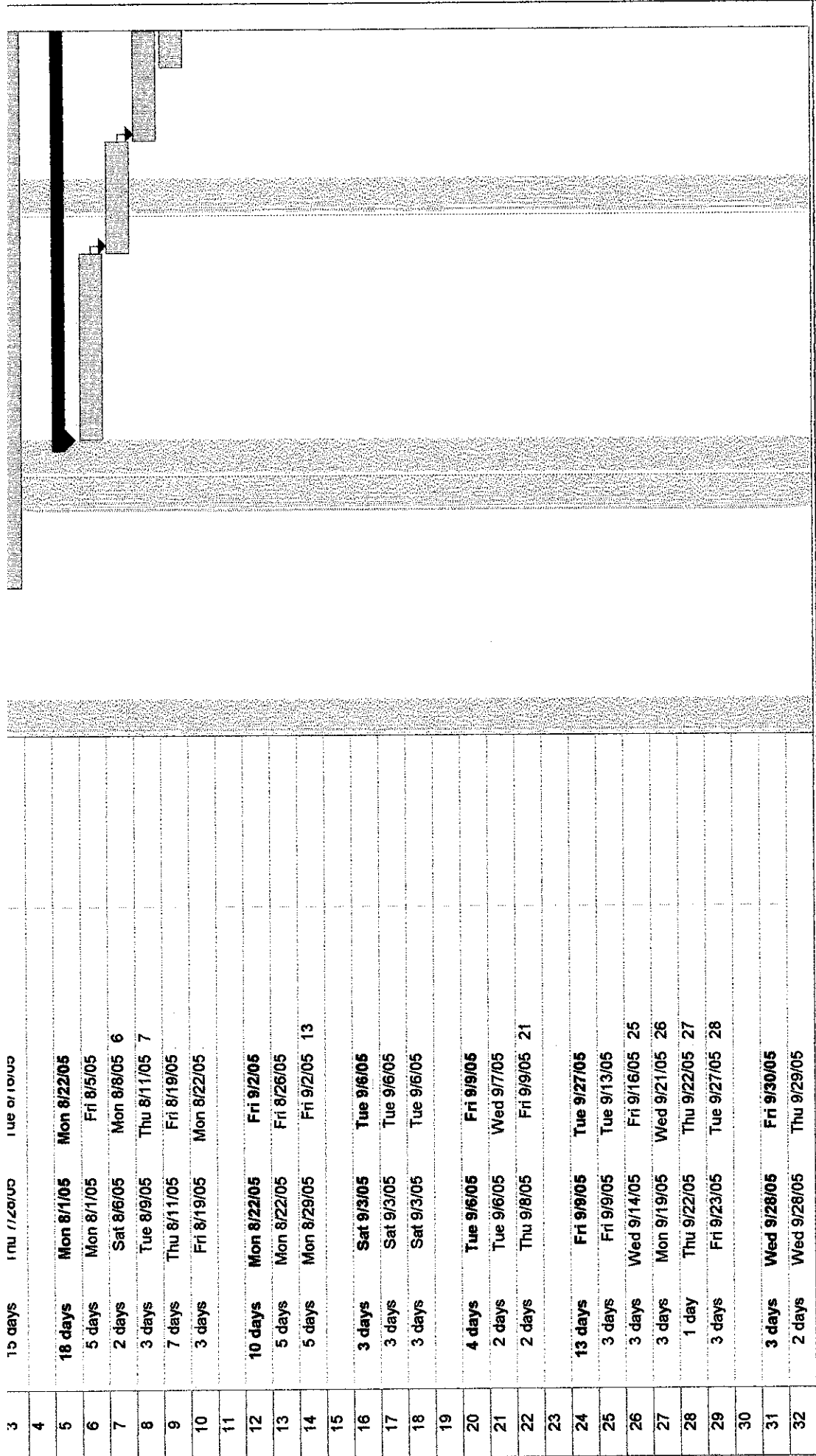
```
capPaletteAuto = SendMessage(lwnd, WM_CAP_PAL_AUTOCREATE, iFrames,  
iColor)
```

```
End Function
```

```
Public Function capPaletteManual(ByVal lwnd As Long, ByVal fGrab As Boolean, ByVal  
iColors As Long) As Boolean
```

```
capPaletteManual = SendMessage(lwnd, WM_CAP_PAL_MANUALCREATE, fGrab,  
iColors)
```

```
End Function
```



External Tasks

External Milestone

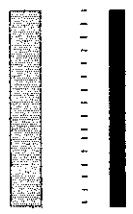
Deadline



Milestone

Summary

Project Summary



Task

Split

Progress

Project: fyp
Date: Tue 12/13/05

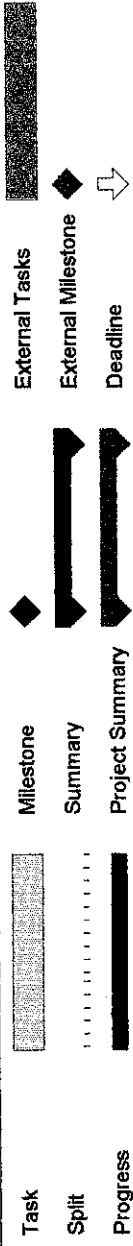
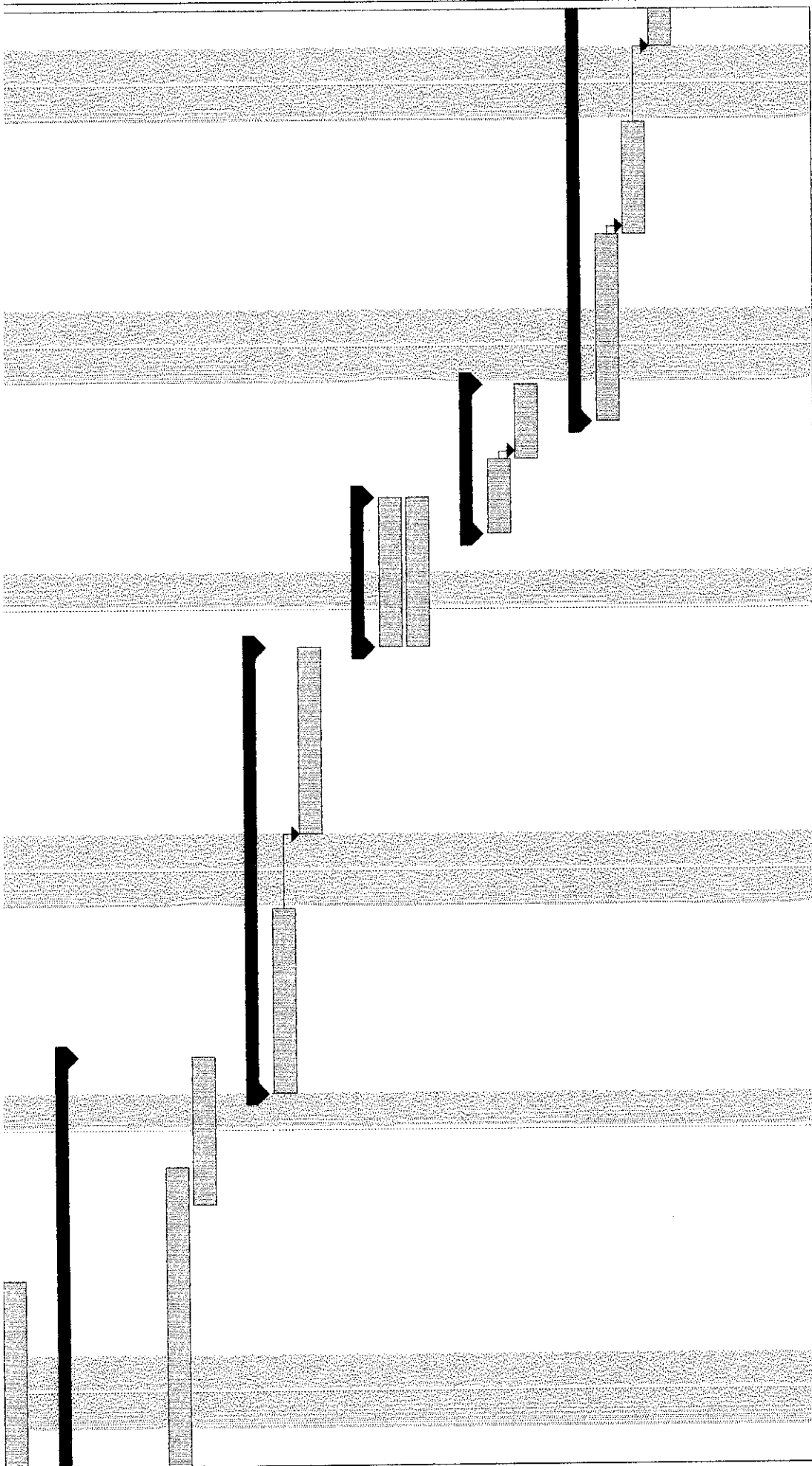
35	9 days	Mon 10/3/05	Mon 10/13/05	
36	3 days	Mon 10/3/05	Wed 10/5/05	
37	6 days	Thu 10/6/05	Thu 10/13/05	36
38				
39				
40	6 days	Mon 10/17/05	Mon 10/24/05	
41	6 days	Mon 10/17/05	Mon 10/24/05	

External Tasks
External Milestone
Deadline

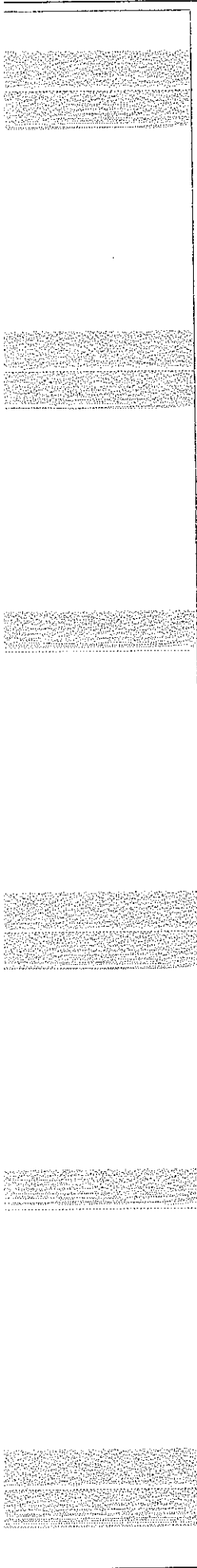
Milestone
Summary
Project Summary

Task
Split
Progress

Project: fyp
Date: Tue 12/13/05



Project: fyp
Date: Tue 12/13/05



External Tasks



External Milestone



Deadline



Milestone



Summary



Project Summary



Task



Split



Progress

Project: fyp
Date: Tue 12/13/05



External Tasks



External Milestone



Deadline



Milestone



Summary



Project Summary



Task

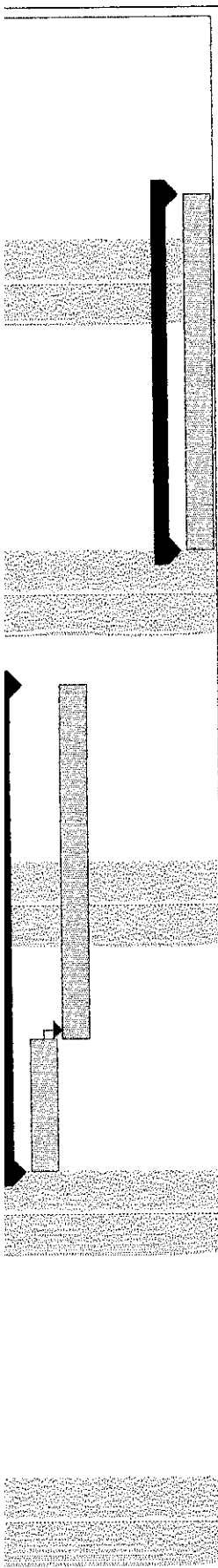


Split



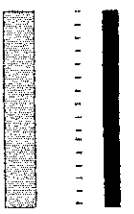
Progress

Project: fyp
Date: Tue 12/13/05

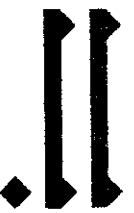


Project: fyp
Date: Tue 12/13/05

Task
Split
Progress



Milestone
Summary
Project Summary



External Tasks
External Milestone
Deadline

