Chapter One

Introduction

1.1 Introduction

Human–computer interaction (HCI) is the study of how humans interact with computer systems. Many disciplines contribute to HCI, including computer science, psychology, economics, engineering, and graphic design.

A basic goal of HCI is to improve the interactions between users and computers by making computers more usable and receptive to the user's needs, specifically it is concerned with:

- Methods for implementing interfaces.
- Techniques for evaluating and comparing interfaces.
- Developing new interfaces and interaction techniques.

Computer systems are combination of hardware and software components that receive input from a user, and communicate output to support his or her task. Although the user interface is simply the part of the computer system that enables interaction and serves as a bridge between users and the system so poorly interface designed can lead to many unexpected problems.

1.2 Problem Statement

Using the computer mouse for a long time results in damaging the arm muscle thus, producing pain in the human body. In addition to that, people who lost their hand due to any accidents and substitute it with an artificial arm cannot use the computer mouse in a proper way because of their delay in controlling fingers.

1.3 Proposed Solution

Design a program to add a vision capability to the computer operation system (windows), so as to enable the control of the computer mouse movement in the pc screen according to the position of the color finger-tip.

1.4 Methodology

The methodology for achieving the proposed solution can be categorized into three main parts; preparing the camera for capturing snapshot, extracting the position of colored finger-tip where segmentation is used to identify the color object and the center position of the object is calculated. Finally performing the user required mouse events depending on the finger-tip color. Where the red object (finger-tip) move the mouse and the blue object perform left click and the green object perform right click.

1.5 Thesis Outline

This thesis consists of five chapters as follow;

Chapter two presents the literature review and the principles used in Human Computer Interaction.

Chapter three describes the details of the system design.

Chapter four presents the simulation, and the results obtained from testing the system.

Chapter five is a conclusion and recommendations for future work.

Chapter Two

Human Computer Interaction

(HCI)

2.1 Introduction

The term Human–Computer Interaction (HCI) is the study of how humans interact with computer systems. Many disciplines contribute to HCI, including computer science, psychology, ergonomics, engineering, and graphic design. It has been in widespread use since the advent of the IBM computer for personal use in the mid-1980s. However, the groundwork for the field of HCI certainly started earlier, at the onset of the Industrial Revolution. Tasks became automated and power-assisted, primarily to save labor, but also motivated by the need to overcome some limitations in human abilities and to perform tasks at a reduced cost. This triggered an interest in studying the interaction between humans and machines in order to make the interaction between them more effective. To enable humans to effectively interact with devices to perform tasks and to support activities, systems need to be designed to be useful and to be usable. HCI is a broad term that covers all aspects of the way in which people interact with computers. In their daily lives, people are coming into contact with an increasing number of computer-based technologies. Some of these computer systems, such as personal computers are used directly. The main components of HCI are humans, computers and interaction.

Humans can be single or multiple users, with diverse physical and mental abilities, interacting cooperatively or competitively. Computers are not just PCs but also a range of embedded computing devices and a range of device sizes such as dust, tabs, pads and boards.

Interaction may be directed via a command or by manipulating virtual objects (windows, desktop) but it can also involve more natural interaction such as speech interaction, gestures.

2.2 Related Work

Much of the research in the field of Human-Computer Interaction takes an interest in many areas

In [1] the authors designed a computer interface, where they present a new approach for controlling mouse movement using a real-time camera. Their method used a camera and computer vision technology, such as image segmentation and gesture recognition, to control mouse tasks (left and right clicking, double-clicking, and scrolling). The results showed that their method can perform everything a current mouse device can do.

In [2] the authors presented a new approach for a 3D mouse for interacting with virtual objects appearing on a computer display. The 3D mouse consists of a body configured to be moveable on a surface, representing the plane of 3D graphics, and has a position sensor for producing the main control signals.

In [3] the authors implemented interfaces by means of software tools where they provide a graphical user interface to old programs without changing the existing application code. They did this by providing an in-memory buffer that pretends to be the screen of an old character terminal, leading to an easily programming interface.

In [4] the authors implemented a software tools for dynamic data visualization. These tools, emphasize the display of dynamically changing data on a computer, and are used as front ends for process control, system monitoring and data analysis.

In [5] the authors introduced an evaluation methodology for text editor; the basis of their approach was classification of editing tasks and evaluation along several dimensions, including time to perform tasks, error cost, learning time, and functionality.

2.3 HCI Design Methodologies

A number of diverse methodologies outlining techniques for human–computer interaction design have emerged since the rise of the field in the 1980s. Most design methodologies stem from a model for how users, designers, and technical systems interact. Early methodologies, for example, treated users' cognitive processes as predictable and quantifiable and encouraged design practitioners to look to cognitive science results in areas such as memory and attention when designing user interfaces. Modern models tend to focus on a constant feedback and conversation between users, designers, and engineers and push for technical systems to be wrapped around the types of experiences users want to have, rather than wrapping user experience around a completed system.

Activity theory used in HCI to define and study the context in which human interactions with computers, it provides a framework for describing the structure, development, and context of computer supported activity .there are several types of HCI design such as user-centered design and Value sensitive design.

User-centered design: user-centered (UCD) is a modern, widely practiced design philosophy rooted in the idea that users must take center-stage in the design of any computer system. Users, designers and technical practitioners work together to articulate the wants, needs and limitations of the user and create a system that addresses these elements. Often, user-centered design projects are informed by ethnographic studies of the environments in which users will be interacting with the system. This practice is similar but not identical to participatory design, which emphasizes the possibility for end-users to contribute actively through shared design sessions and workshops.

Principles of user interface design: these are seven principles of user interface design that may be considered at any time during the design of a user interface in any order: tolerance, simplicity, visibility, affordance, consistency, structure and feedback.

Value sensitive design: Value Sensitive Design (VSD) is a method for building technology that account for the values of the people who use the technology directly, as well as those who the technology affects, either directly or indirectly. VSD uses an iterative design process that involves three types of investigations: conceptual, empirical and technical. Conceptual investigations aim at understanding and articulating the various stakeholders of the technology, as well as their values and any values conflicts that might arise for these stakeholders through the use of the technology. Empirical investigations are qualitative or quantitative design research studies used to inform the designers' understanding of the users' values, needs, and practices. Technical investigations can involve either analysis of how people use related technologies, or the design of systems to support values identified in the conceptual and empirical investigations.

2.4 HCI in Ubiquitous Computing

In ubiquitous computing Explicit HCI refers to the processes and the models for design and the operational interface for some products. For many users, the User Interface (UI) part of the system is the product. HCI is considered from the perspective of the explicitness (eHCI) versus the implicitness (iHCI) of the interaction. In which the focus on the human is having a model of the system rather than the system having a model of the individual user. HCI can be classified into explicit HCI(eHCI) and implicit HCI(iHCI).

2.4.1 Explicit HCI

This technique put the user at the centre of the interactive systems, so that the control of the system responds to and is driven externally by the user, rather than the system being driven internally.

Poorly designed user interface (UI) can lead to both higher training costs and higher usage costs and of course leads to lower product sales. The reasons for higher training costs include users spending time working out what is happening, trying out inappropriate computer services and impaired task quality. Users may feel that a particular machine forces them to do tasks in ways they prefer not to (no control). Users may have to re-learn how to perform tasks: starting work again lowers productivity. Poor UIs can lead to higher error rates not acceptable in safety-critical systems that require some human interaction. The motivation for HCI is to be effective in three ways;

- Useful: accomplish a user task that the user requires to be done.
- Usable: do the task easily, naturally, safely (without danger of error).
- Used: enrich the user experience by making it attractive, engaging.

The success of a product depends largely on both the user's experience with how usable a product is and how useful it is in terms of the value it brings to them. This combination has been neatly summarized as Heckel's law and Heckel's inverse law [6]. Heckel's law states that the quality of the user interface of an appliance is relatively unimportant in determining its adoption by users if the perceived value of the appliance is high. Heckel's inverse law states the importance of the user interface design in the adoption of an appliance is inversely proportional to the perceived value of the appliance. What these laws express is that although the

usability of the UI is important, the overriding concern is the usefulness of the device itself. If a difficult user interface acts as an inhibitor to the uptake of an appliance, then the appliance probably does not have enough perceived value to be useful.

2.4.1.1 Complexity of Ubiquitous Explicit HCI

Explicit HCI is complex for Ubiquitous Computing scenarios even if it is well designed for individual devices because there is a need to use tasks as part of activities that require access to services across multiple devices.

The amount of explicit HCI used can overwhelm users as more and more computer devices require inputs, even if individually, computers are designed to be usable. Individual devices may be Human–Computer Interaction simple to use but sometimes multiple devices may need to be used concurrently to support multiple concurrent user activities such as receiving a phone call while performing some other activity. Some tasks may require multiple individual devices to be configured and their individual behavior to be orchestrated, e.g., to record live audio–video content while playing already recorded content.

2.4.2 Implicit HCI

Implicit interaction is based on the assumption that the computer has a certain understanding of users' behavior in a given situation: it is a user-aware type of context-aware system. This knowledge of users' behaviors in given situations is then considered an additional input to the computer while doing a task [7]. For example, consider a person entering a very dark room to retrieve an object, he or she will explicitly switch on the lights. A key part of the usability of the interaction design is to position the switch to be conveniently reachable and possibly to

illuminate the switch so that it can be activated in the dark. Alternatively, human interaction may be implicitly modeled, rather than requiring humans to switch on a light when someone enters the room, the environment is smart. It contains devices which monitor people entering the room and switches on the light for authorized people automatically. Another example is shown in figure-(2.1) where the system activates the scream for presentation when the user left the meeting table toward the screen.

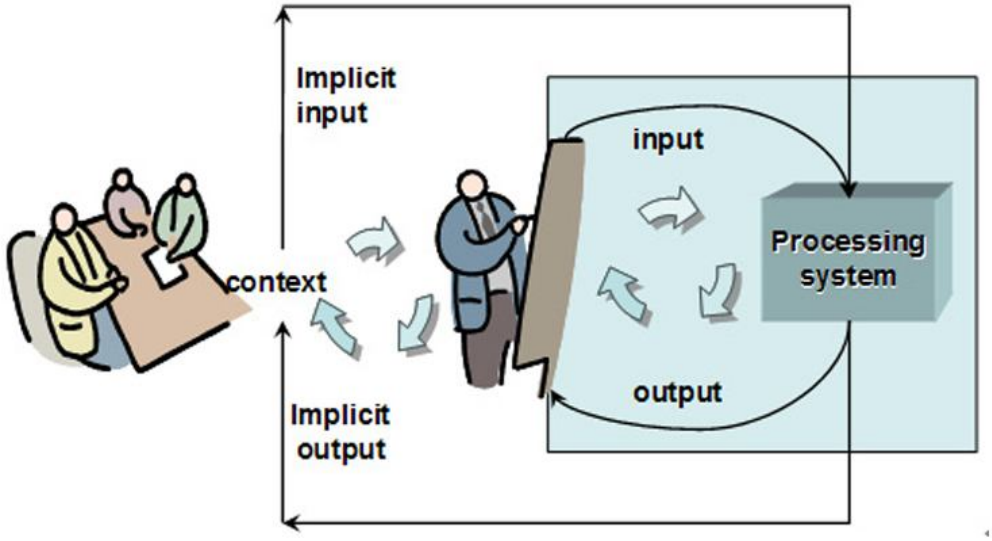


Figure (2.1): Implicit HCI Example

Implicit interaction can allow some of the interaction to be hidden from users and hence for the device to become invisible. Implicit interaction is also introduced when designing systems which can interact with in a more natural way. For example, if a hand gestures is used, such as a clap, to control a device to switch it on and off, the clap may also be used for other reasons such as to express an emotion. Unless the context of the gesture is also defined and shared, clapping cannot be unambiguously used to imply that it is a command to switch a device on or off.

2.4.2.1 Complexity of Ubiquitous Implicit HCI

There are some obvious challenges in supporting implicit interaction. It can sometimes be complex to accurately and reliably to determine the user context because of: the non-determinism of the subject, the user has not clearly decided what to do, or because of non-determinism in the subject's environment. Implicit interaction, in contrast to explicit interaction, reduces or even removes explicit user control. The user context determination may however invade and distract users' attention in order to directly interact with them or the determination may be inaccurate because it can only indirectly model users, e.g., via observing user interaction and inferring user behavior. Systems may also require time in order to learn and build an accurate model of a user.

Partial implicit HCI systems can be very useful in practice if designed appropriately. For example, they can be used to anticipate actions, to priorities choices and to lessen the overload on and the interaction by users.

2.5 User Interfaces and Interaction for ICT Devices

The most commonly used networked ICT device is the personal computer in its various forms such as desktop and laptop, hand-held mobile devices. Each of these can be designed to perform a common set of functions. Some methods of the user interfaces of these devices are discussed here.

2.5.1 Personal Computer Interface

Windows systems were demonstrated as early as 1965. From the early 1980s, computers started to become more widely used by non-specialist users. The keyboard and visual command interface which allowed text to be entered and displayed one line at a time dominated computer interfaces. Users needed to be

able to remember the command name and the syntax of any parameters used to qualify the command such as data files for the command to act on.

From the early 1990s, the Windows interface which supports direct manipulation of visual objects prevailed. Users can interact with computers by typing text commands but for many tasks direct interaction can be used by activating and moving active window sub-areas called icons and menus, using mouse clicks and mouse drag-and-drop respectively. Such an interface is commonly called WIMPS (Windows, Icons, Menu and Pointer device) and direct manipulation. It is the dominant interface for desktop and laptop computers at this time. The mouse as a pointer device was first demonstrated in 1965 as a replacement for the light-pen. The WIMPS interface is associated with a desktop metaphor. The two main advantages of the WIMPS UI over the command UI are that the order of multiple commands can be much more ad hoc and the use of direct manipulation means users do not need to remember command names. WIMPS UI is not an improvement for visually impaired users; it consumes screen space which is more critical in lower resolution displays; the meaning of visual representations may be unclear or misleading to specific users, also mouse pointer control and input require a good hand–eye coordination that can be slow.

2.5.2 Mobile Hand-Held Device Interfaces

Early mobile devices did not work effectively on PC-style WIMPS interaction because the display area is smaller. It is impractical to have several windows open at a time. It can be difficult to locate windows and icons. The design of user interfaces for mobile devices impose several challenges on interaction design. Screen real estate is very small due to limited physical size of the devices stressing the design of graphical interfaces and forcing designers to explore the use of new

means of output. Especially the use of sound is being carefully investigated. Correspondingly, interaction with mobile device interfaces is limited due to handheld operation forcing designers to explore new means of input such as gestures, speech, environmental sensors and context awareness. Now a day Screen navigation using fingers on a touch-pad are bigger and can use a PC-style WIMPS interaction.

2.6 Hidden User Interfaces

The WIMPS-style interface which dominates PCs is considered by many computer scientists to be obtrusive in the sense that it requires users to consciously think about how to operate a mouse pointer interface and which keys to press to use the computer. The computer itself is localized and users must go to its location to use it. In contrast, systems which can be situated where our activities are based and which directly make use of natural human sensory input offer a less obtrusive computer interface, figure (2.2) shows an example of hidden user interfaces and the way human-machine interface translating the aspects of a knowledge base into modalities of perception for a human operator. In the following sections, several of these more natural interfaces are considered

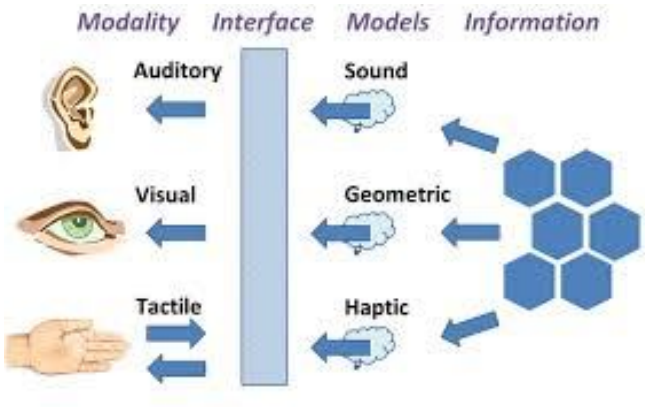


Figure (2.2): Hidden User Interfaces Example

2.6.1 Auditory Interfaces

Auditory interfaces have the benefit of supporting hands-free input and output which can be used as an additional interaction modality when the visual senses are already being consciously used. Auditory interfaces can support natural language text input as a replacement for keyboard text input. There are two basic auditory interfaces: speech-based and non-speech based. Auditory interfaces can provide an interface between the user and a computer-based application that permits spoken interaction with applications in a relatively natural manner compared to WIMPS interfaces.

2.6.2 Visual Interfaces

The modality of interaction refers to a mode of human interaction using one of the human senses and to the type of computer input. The categories of human senses are sight, touch, hearing, smell, and taste. ICT systems have modalities that are equivalent to some of the human senses such as cameras (sight), input devices such as touch screens, keypads and pointer devices (touch), microphones (hearing) and the use of various chemical sensors and analyzers (smell and taste). In [8] Jaimes and Sebe (2005) classify modalities into two types: the human senses and other ICT device modalities such as mouse and keyboard. They defined several types of multi-modal interfaces. Perceptual interfaces seek to leverage sensing (input) and rendering (output) technologies in order to provide interactions not feasible with standard interfaces and common I/O devices such as the keyboard, the mouse and the monitor. Attentive interfaces or iHCI are context-aware interfaces that rely on a person's attention as the primary input. The goal of attentive interfaces is to use gathered information to estimate the best way to communicate with the user.

2.6.3 Tactile Interfaces

Tactile interface are base on gesture, but there are expressive, meaningful body motions involving physical movements of the fingers, hands, arms, head, face, or body, with the intent of conveying meaningful information about interacting with the environment. There are three main types of body gestures: hand and arm gestures, head and face gesture and full body movement Gestures can be sensed using wearable devices such as gloves or body suits; by attaching sensors such as various magnetic field trackers, accelerometers and gyroscopes to the surface of the body; by using cameras and computer vision techniques. Figure-(2.3) shows an example of gesture interface for a human arm movement

Figure (2.3): Human Arm Gesture Interface

Gesture recognition has wide-ranging UbiCom applications such as: developing aids for those with impaired hearing enabling very young children to interact with computers; designing techniques for passive identification such as gait analysis; recognizing sign language used by deaf people; monitoring people's emotional states or stress levels (as part of affective computing, classifying body behavior

Such as lie detection; navigating and/or manipulating virtual environments and to enhance interaction with ICT devices such as games consoles.

The first basic gesture interfaces were the pen-based gesture interfaces based upon light-pens in the mid-1960s, allowing pointers to be moved on the screen, synchronized to the movement of the pen. This can also emulate regular hand-written characters and can be used as a substitute for keyboard-based text input.

2.7 Combining Input and Output User Interfaces

In the UIs discussed so far, input devices are separated from output devices. For example, a pointer input device such as a mouse only outputs a single position (x, y) event at any given time together with one, two, or three button presses. The state of the input is available as a visual cue only. One of the earliest combinations of user input and an output device is the touch-screen where the user can touch the display in order to activate a selection on the screen [8].

Touch-screen is display where the position of contact with the screen, generally with pointed physical objects such as pens or with fingers, can be detected. An event can then be generated for an associated visual object at that position and an associated action triggered. A touch-screen can behave as a soft control and display panel that is reprogrammable and which can be customized to suit a range of applications and users. Touch-screens can operate using a variety of electromagnetic principles such as resistive, capacitive, surface acoustic waves, etc. Resistive touch-screens are composed of two thin metallic electrically conductive and resistive layers separated by a thin space. When some object touches the screen, the layers connect at the contact point and using basic electricity principles such as Kirchoff's Laws, the position can be deduced from

the position-dependent current flow [8]. Touch-screens today are used routinely in many applications and many devices such as point of sales terminals, mobile devices and games consoles and vehicle navigation systems. They can also support particular movements of single fingers to indicate a next screen command and support multiple finger gestures such as moving two fingers resting on an object away from the object indicating zooming out, magnifying or stretching the object.

Chapter Three

Finger Tracking System

3.1 Introduction

For a long time research on human-computer interaction (HCI) has been restricted to techniques based on the use of monitor, keyboard and mouse. Recently this paradigm has changed. Techniques such as vision, sound, speech recognition, projective displays and location aware devices allow for a much richer, multi-modal interaction between man and machine.

Finger-tracking is the usage of bare hand to operate a computer in order to make human-computer interaction much faster and easier, fingertip finding deals with extraction of information from hand features and positions. They use the position and direction of the fingers in order to get the required segmented region of interest.

Finger tracking system focuses on user-data interaction, where the user interacts with virtual data, by handling through the fingers the volumetric of a 3D object that want to represent. This system was born based on the human-computer interaction problem. The objective is to allow the communication between them and the use of gestures and hand movements to be more intuitive, Finger tracking systems have been created. These systems track in real time the position in 3D and 2D of the orientation of the fingers of each marker and use the intuitive hand movements and gestures to interact.

The system is able to track the finger movement without building the 3D model of the hand. Coordinates and movement of the finger in a live video feed can be taken to become the coordinates and movement of the mouse pointer for human-computer interaction purpose.

3.2 General Block Diagram of Finger Tracking System

Finger-tip tracking idea is based on object detection principles, and the object is holding on the figure-tip. A block diagram of the finger tracking system is shown in figure3.1.The system require a webcam attached to the computer so as to add a vision capability, then extract the object and identifies its position, after that the mouse is moved to the object position, and so on the mouse movement will be according to position of the figure-tip.

Figure (3.1): General Block Diagram for System Design

3.2.1 Image Acquisition

The first stage of any vision system is the image acquisition stage, and it can be gray scale or colored, after the image has been obtained, various methods of processing can be applied to the image to perform many different vision tasks required. However, if the image has not been acquired satisfactorily then the intended tasks may not be achievable, even with the aid of some form of image enhancement, the webcam of the computer is used as a sensor, and it captures the real time video at a fixed frame rate and resolution which is determined by the hardware of the camera. The frame rate and resolution can be changed in the system if required [9].

3.2.1.1 Gray Scale Image Acquisition

Image acquisition principles transform illumination energy reflected by the scene into digital image. The incoming energy is transformed into a voltage by the sensor material then the digital quantity is obtained from each sensor by digitizing its response as illustrated in figure (3.2)

The scenario of image acquisition is as follow

1. Illumination source reflected from a scene element.
2. Imaging system collects the incoming energy and focuses it onto an image plane using sensor array.
3. Response of each sensor proportional to the integral of the light energy projected.
4. Sensor output ,the analog signal is digitized (Quantized, Encoded)
5. Image readout one row at a time, each row transferred in parallel to a serial output register[10]

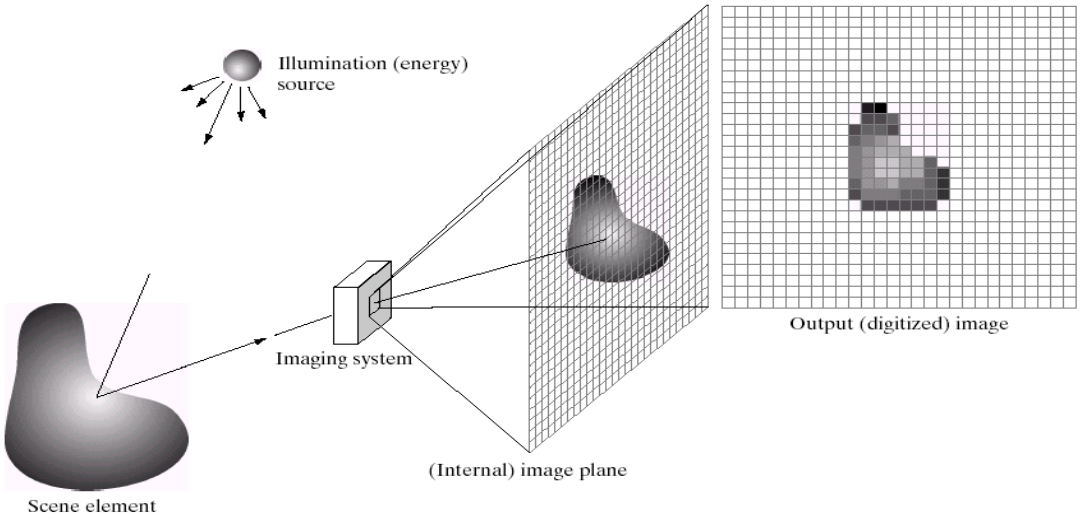
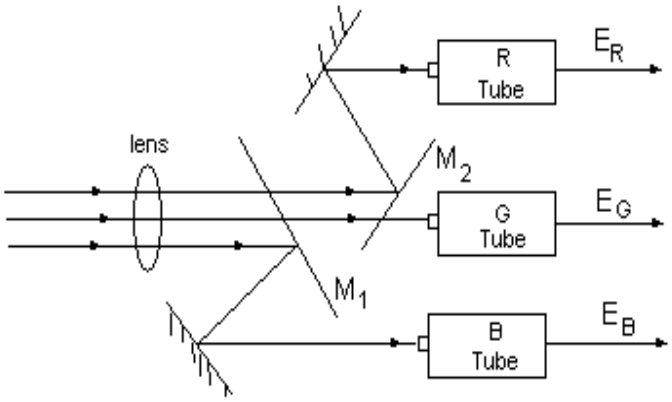


Figure (3.2): Transform of Illumination Energy into Digital Image.

3.2.1.2 Color Image Acquisition

The source is three separate monochrome cameras with a common lens system and diachronic optics to split the light from scene into 3-component red, blue and green as shown in figure (3.3)



M1= dichotic mirror reflects blue and transmits red and green

M2= dichotic mirror reflects red and transmits blue and green

Figure (3.3): Principles of Chrome Camera

3.2.2 Representation of Digital Image

Digital image is a finite collection of discrete samples (pixels) of any observable object, each pixel having its own discrete value in a finite range.

Digital image consists of N x M pixels, each is represented by k bits. A pixel can thus have 2k different values, typically illustrated using different shades of gray scale image is given in Figure (3.4). In practical applications, the pixel values are considered as integers varying from 0 (black pixel) to 2k-1 (white pixel). The images are obtained through a digitization process, in which the object is covered by a two-dimensional sampling grid. The main parameters of the digitization are

- Image resolution: the number of samples (pixels) in the grid.
- Pixel accuracy: number of bits used per sample,[10].

3.2.2.1 Gray Scale Image

An image is an array, or a matrix, of square pixels (picture elements) arranged in columns and rows. In an 8-bit grey-scale image each picture element assigned an intensity that ranges from 0 to 255. An example of gay image is shown in figure (3.4).

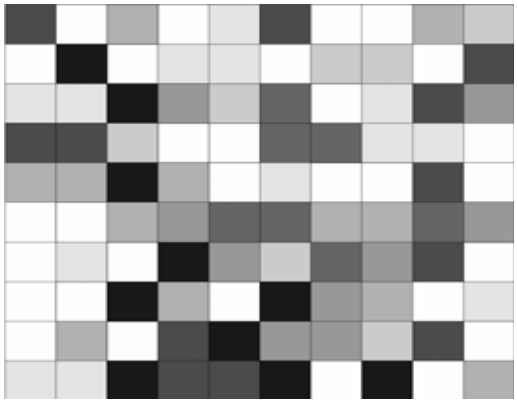


Figure (3.4): Each Pixel has A value from 0 (Black) to 255 (White).

An image of size: M*N pixel can be represented as a matrix as shown in figure (3.5) where each (x, y) position contain the value of the gray level [10].

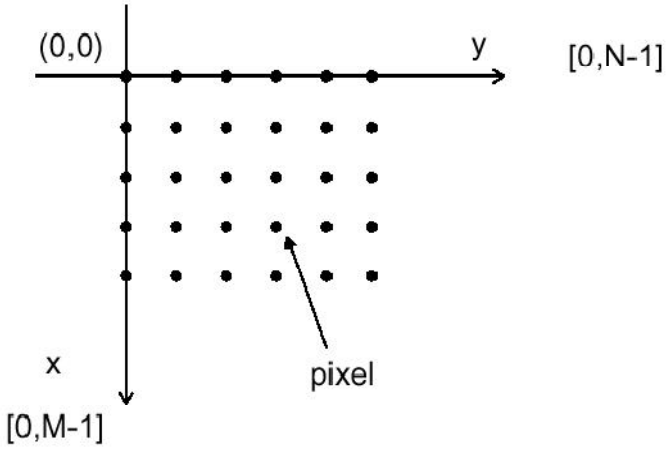


Figure (3.5): Matrix of A gray Scale Image

3.2.2.2 Color Image

According to the theory of the human eye, all colors are seen as variable combinations of the three so-called primary colors red (R), green (G), and blue (B). The primary colors can be added to produce the secondary colors magenta (red + blue), cyan (green + blue), and yellow (red + green), mixing all the three primary colors results in white. There exist several useful color models: RGB, YUV, and HIS [9].

1. RGB Color Model

The RGB color space can be considered as a three-dimensional unit cube, in which each axis represents one of the primary colors. The image is stored in an RGB format as three matrices, and each pixel has a three color component as shown in figure (3.6).

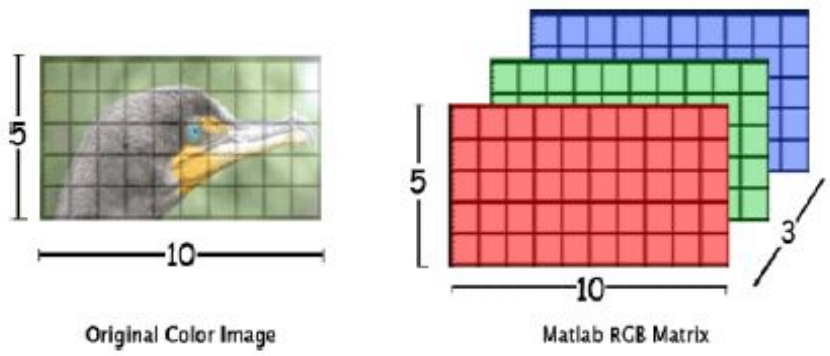


Figure (3.6): Representation of RGB Image

- (x,y,1) red pixel component
- (x,y,2) green pixel component
- (x,y,3) blue pixel component

Extracting a red component can be obtained by taking the image red pixels values in a new matrix R

2. YUV Color Model

The basic idea in the YUV color model is to separate the color information apart from the brightness information. The components of YUV are:

$$Y = 0.3 \cdot R + 0.59 \cdot G + 0.11 \cdot B \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (1)$$

$$U = B - Y \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2)$$

$$V = R - Y \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (3)$$

Y represents the luminance of the image, while **U, V** consists of the color information (chrominance). The luminance component can be considered as a gray-scale version of the RGB image. The advantages of YUV compared to RGB are:

- The correlations between the color components are reduced.
- The brightness information is separated from the color information
- Most of the information is collected to the Y component, while the information content in the U and V is less.

3. HSI Color Model

The HSI model consists of hue (H), saturation (S), and intensity (I). Intensity corresponds to the luminance component (Y) of the YUV models. Hue is an attribute associated with the dominant color as perceived by an observer. Saturation refers to relative purity of the amount of white light mixed with hue [9].

3.3 Information Extraction

The required information to be extracted is the position of the finger-tip. The main functions in information extracting are segmentation and position identification.

3.3.1 Segmentation

The process of breaking an image up into various regions is called segmentation. It is partitioning an image into a set of disjoint regions. Accurate image segmentation is a fundamental importance in computer vision and image processing**.** In another word Image segmentation refers to the decomposition of a scene into its component. It is a key step in image analysis.

Image Segmentation methods looked for objects that either have some measure of homogeneity within themselves or have some measure of contrast with the objects on their border. The homogeneity and contrast measures can include features such as gray level, color and texture. There are many methods used for segmentation such as region growing, clustering, thresholding, and adaptive thresholding [9].

3.3.1.1 Region Growing

This technique is a bottom-up procedure that starts with a set of seed pixels. The aim is to grow a uniform connected region from each seed. A pixel is added to a region if and only if

- It has not been assigned to any other region.
-It is a neighbor of that region.
-The new region created by addition of the pixel is still uniform.

A complete segmentation of an image must satisfy a number of criteria:

- All pixels must be assigned to regions.
- Each pixel must belong to a single region only.
- Each region must be a connected set of pixels.
- Each region must be uniform.
- Any merged pair of adjacent regions must be non-uniform.

3.3.1.2 Clustering

In this technique of image segmentation methods, individual elements are placed into groups; these groups are based on some measure of similarity within the group. The major difference between these techniques and region growing techniques is that the spatial domain may be considered as the primary domain for clustering. Some of these other domains include color spaces, histogram spaces, or complex feature spaces.

The simplest clustering method is to divide the space of interest into regions by selecting the center or median along each direction and splitting it here; this can be done iteratively until the space is divided into specific number of regions needed.

3.3.1.3 Thresholding

Thresholding was used to segment an image by setting all pixels whose intensity values are above a threshold level to a foreground value and all the remaining pixels to a background value. So the image will be changed to binary image white objects and black background.

3.3.1.4 Adaptive Thresholding

Adaptive thresholding changes the threshold dynamically over the image. This more sophisticated version of thresholding can accommodate changing lighting conditions in the image, *e.g.* those occurring as a result of a strong illumination gradient or shadows. The drawback of this method is that it is computational expensive and, therefore, is not appropriate for real-time applications [9].

3.3.2 Object Position Identification

Segmentation converts the image to a set of foreground pixels to define the object and background. It is important to threshold them so as to yield a set of binary

image so as to extract the object position by labeling the foreground binary pixels representing the object. Labeling the foreground object depend on the connected point principles.

3.3.2.1 Connected Point Principles

One simple relationship between pixels is connectivity in which pixels are next to each others. There are two methods for defining connectivity of pixels

4-connected neighbors which are the pixels share an edge of a pixel as shown in figure (3.7A), and 8-connected neighbors that share the edge and corners of a pixel as shown in figure (3.7B).

A B

Figure (3.7): Methods for Defining Connectivity

3.3.2.2 Labeling Algorithm

The objective of connected component labeling is to determine all the connected set of components in an image and assign a distinct numerical value label to each pixel in the same connected component. There are two approaches for connecting component Labeling; Recursive and sequential [10].

Recursive

a binary image scanned from left to right, an unlabeled object pixel is assigned a label, say *X* and each of its neighboring object pixels is assigned the same label till there is no more unlabelled object pixel in the image.

Sequential

An alternate strategy of connected component labeling can be defined in two passes. In the first pass, each object pixel is assigned a label according to the following criteria

• If the pixel has no connected neighbors with the same value that have already been labeled, create a new unique label and assign it to that pixel.

• If the pixel has exactly one label among its connected neighbor with the same value that has already been labeled, give it that label.

• If the pixel has two or more connected neighbors with the same value but different labels, choose one of the labels and remember that these labels are equivalent.

In the second pass, the equivalent labels are merged to create unique labels for each connected component in the image. If *X* and *Y* are two equivalent labels in E , then reassign Y by X when X < Y,the center point of a binary image can be calculated as follow

First calculating the object area,

$$A = \sum_{Y_{min}}^{Y_{max}} \sum_{X_{min}}^{X_{max}} I(x,y) \dots\dots\dots\dots\dots\dots\dots\dots\dots (4)$$

Then finding the position of the area center

- The x axes given as

$$\bar{x} = \frac{\sum_{Y_{min}}^{Y_{max}} \sum_{X_{min}}^{X_{max}} x * I(x,y)}{A} \dots\dots\dots\dots\dots\dots\dots\dots\dots (5)$$

- The y axes given as

$$\bar{y} = \frac{\sum_{X_{min}}^{X_{max}} \sum_{Y_{min}}^{Y_{max}} y * I(x,y)}{A} \dots\dots\dots\dots\dots\dots\dots\dots\dots (6)$$

After labeling the foreground object and assuming all pixels are square having the same area the values of all area pixels equals to one I(x,y) =1,[9].

3.4 Mouse Processing

A mouse is a pointing device that detects two-dimensional motion relative to a surface. This motion is typically translated into the motion of a pointer on a display, which allows controlling of an interface. Physically, a mouse consists of an object held in one's hand, with one or more buttons.

Mouse clicking means stopping movement while the cursor is within the bounds of an area then pressing and release a bottom to select files, programs or icon. The mouse has two bottoms, the right and left one. The function of left bottom click is to select an icon, and the function of the right bottom click is to give a menu contains all the available actions that can be done.

Chapter Four

Simulation and Results

4.1 Introduction

The program objective is to add a vision capability to the computer operation system (windows), which help the people who lost their hand due to any accidents and substitute it with an artificial arm to use the computer mouse in a proper way because of their delay in controlling fingers.

The solution basic idea is to control the movement of the computer mouse and its function in the pc screen using finger-tip. The methodology used in implementing the program is categorized into three main parts; preparing the camera for capturing snapshot, extracting the position of finger-tip and finally perform the user required mouse events.

4.2 System Setup

The finger-tip system is implemented in Matlab environment, when the system is started it detects the camera attached to the computer and use it to capture a real time video. The user hand will be placed in front of a camera as shown in figure (4.1). Colored dough is held on the finger-tip to be used as a pointer for controlling the computer mouse movement actions. The colored dough are used; red, green, and blue, Each color will be placed on a different finger. The red color is used for computer mouse movement, the blue color is used for mouse left click, and the green color is used for mouse right click.

The system control computer mouse cursor movement and click events using a camera based color detection technique. Real time video has been captured using a Web-Camera, and the colored finger-tip provides information to the system. The system processing techniques involve an image subtraction algorithm to detect

colors; once the colors are detected the system performs various operations to track the cursor and performs control actions.

Figure (4.1): Capturing the Video

4.3 System Flow Chart

The flow chart in figure (4.2) shows a simplified operation of the program.

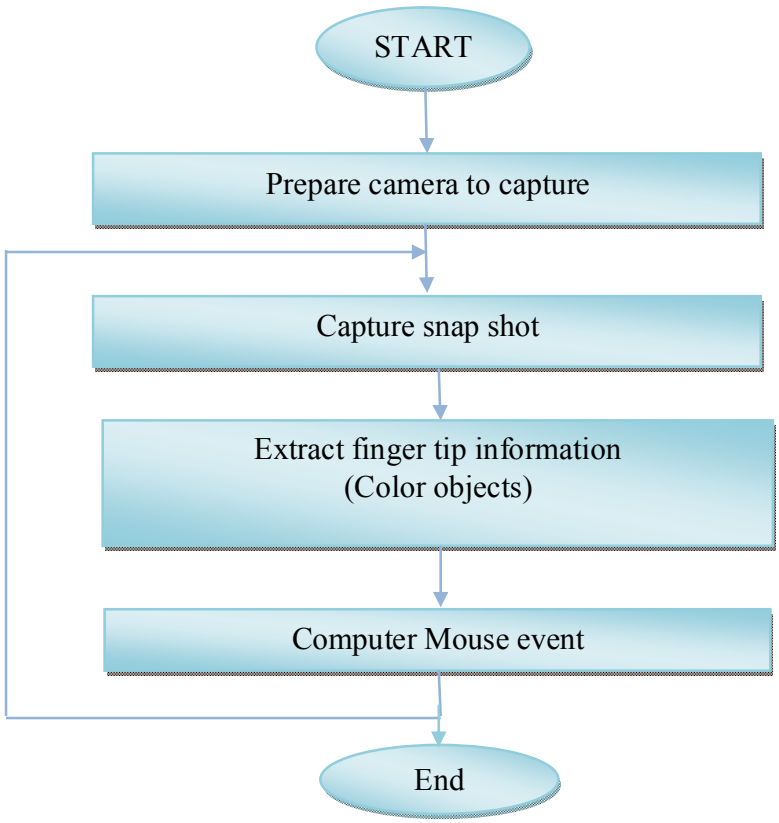


Figure (4.2): Program Operation

4.3.1 Preparing the Camera for Capturing Snapshot

The vision capability is added to the operating systems using a built in camera or an external one. The camera captures a scene of the front of the computer and stores a copy to be processed by the system. The steps in preparing the camera is shown in figure (4.3)

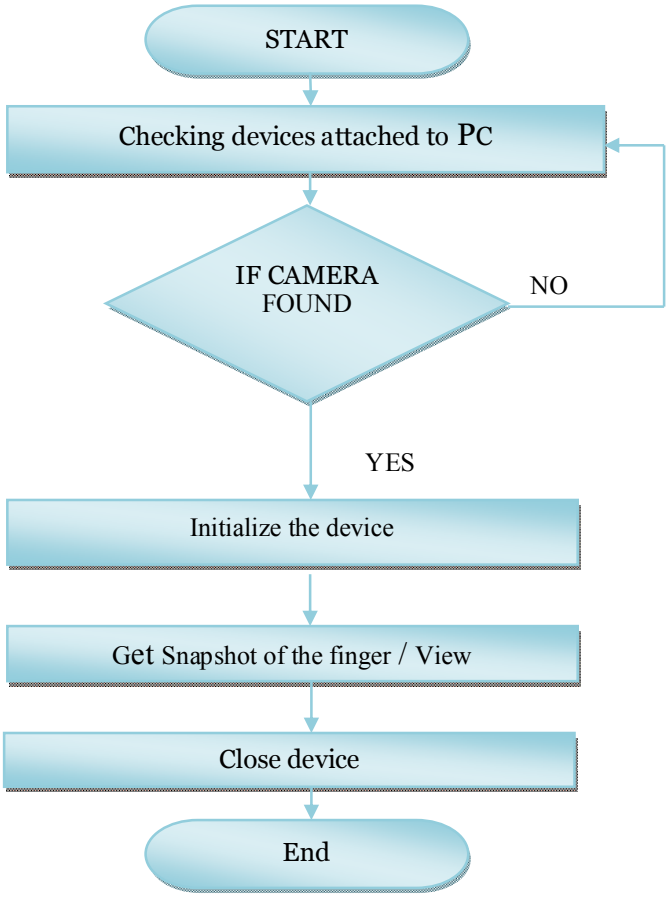


Figure (4.3): The Image Acquisition Steps

1. Checking Devices Attached to PC

Now-a-days most of the cameras are available with USB interface. Once the camera's driver is installed, the computer detects the device whenever it is connected, and Windows operating system automatically detects the device.

MATLAB has a built-in adaptor for accessing these devices and it gives each adapter a name and an identity, these information are used by the system to access the device.

2. Initialize the Device

For the camera to work in Matlab environment a connection must be initiated with Matlab, this is done by using a Matlab built in function with the adapter name and identity.

3. Get Snapshot of the Finger / View

The camera is then initiated to get snapshot image. The way a snapshot is taken must be specified by defining the number of frames taken per trigger, and the number of triggered time. For this project one frame per trigger is satisfied, and because of continuity of the program the number of trigger time must be set to infinity.

4.3.2 Extracting Figure-tip Information

In Matlab environment the image stored in an RGB format as three matrices, and each pixel in the image contains three colors values from the corresponding pixels in each matrices. Extracting a red component can be obtained by taken the image red pixels values in a separated matrix called **RedImage** using one of Matlab built in functions. The same function is used to extract the green component **GreenImage** and the blue component **BlueImage.** After the color components are

saved as different matrices from the RGB image, the Intensity is extracted from them to have a new image that contains only the actual color on different saturations. A median filter is then used to filter out the noise for each color matrix

The color object is selected to have a high saturated value. Thresholding is the method used for segmentation,a proper saturation value is chosen that makes the object area as foreground, and the other color saturations as background. After thresholding the result is enhanced by removing all the areas other than the object.

4.3.3 Computer Mouse Events

Matlab environment does not support computer mouse events so Java classes are import to system program so as to perform the mouse events. The events are performed according to color object detection principles. The flow chart in figure (4.4) shows the steps used in performing the mouse events.

1. Move the Cursor According to Finger-tip Position

Matlab build in function is used to find the center of the red object area. The return value of the function is the X and Y axes, where the mouse cursor must be moved.

2. Mouse Click Events

The red object is responsible of mouse movement. For performing a left click, a blue object must appear in front of the camera snapshot view. Right click is performed when a green object appears in front of the camera snapshot view. Matlab environment does not interact with mouse driver, so Java class functions were used to perform the mouse left click bottom when a blue object is detected. If the mouse spend more time in the same position a double click event occur. The mouse right click is performed when a green object detected.

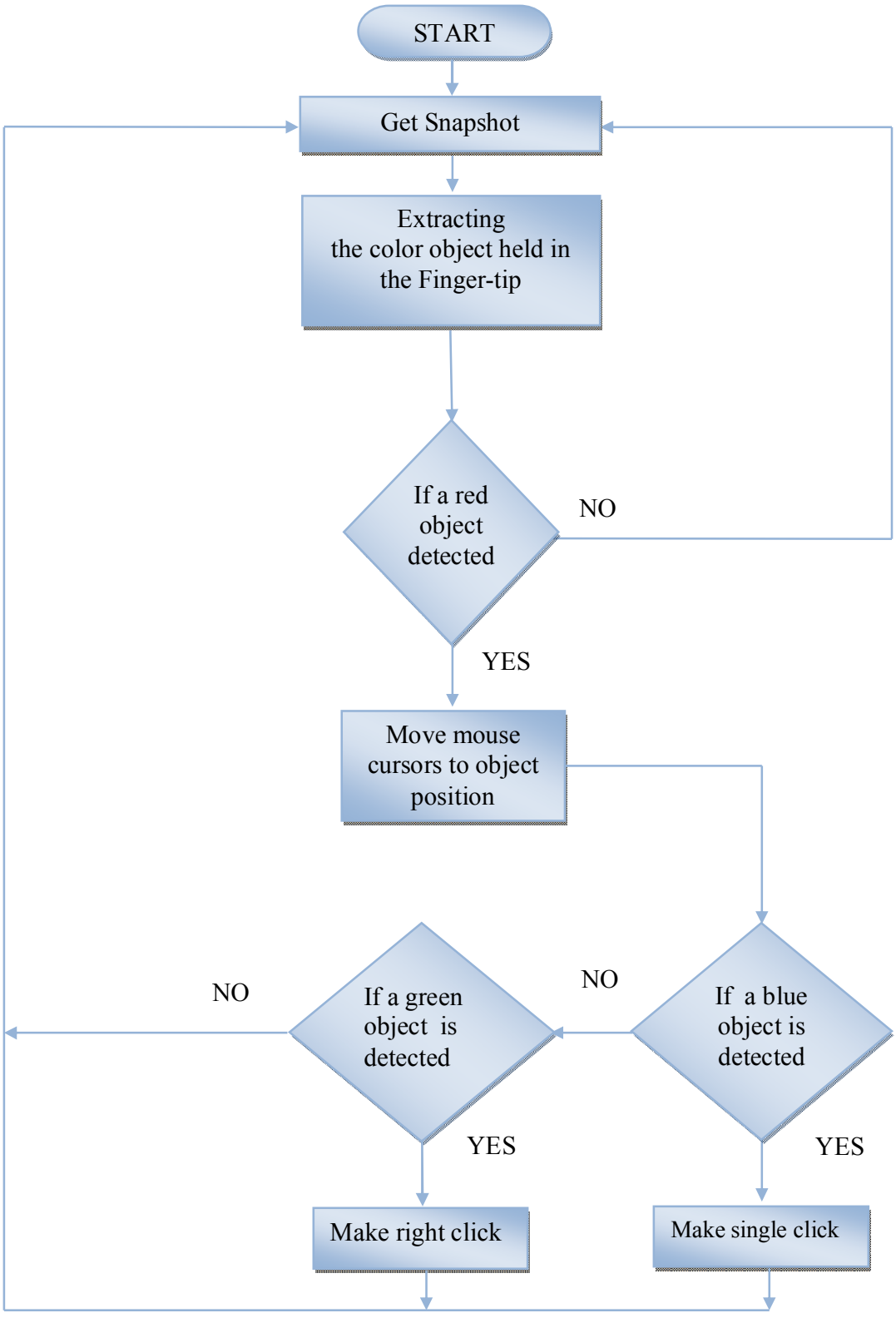


Figure (4.4): The Steps Used in Performing the Mouse Events.

4.4 Results and Discussion

The system setup used for the results taken is as follow;

- Laptop computer with a built in webcam
- black background was used, and the color dough definitions are
  - Red object (230 R+38 B+38G)……(7)
  - Blue object (83 R+253 B+74G)……(8)
  - Green object (21R+11B+213G)……(9)

4.4.1 Extraction of color objects from snapshot image

Figure (4.5) shows a snapshot view in front of the computer. The image is stored as an RGB image. It is the target of processing for color objects detection.

Figure (4.5): RGB Snapshot View

The intensity are extracted from the snapshot view by converting the RGB image to gray scale and the result is shown in figure (4.6).

Figure (4.6): Gray Scale Snapshot View

A separated color images (red, green, blue) are obtained from the RGB snapshot and saved in a separate matrix. The result is shown in figure (4.7), where sub-figures a, b, and c shows the blue, green, and red image respectively.

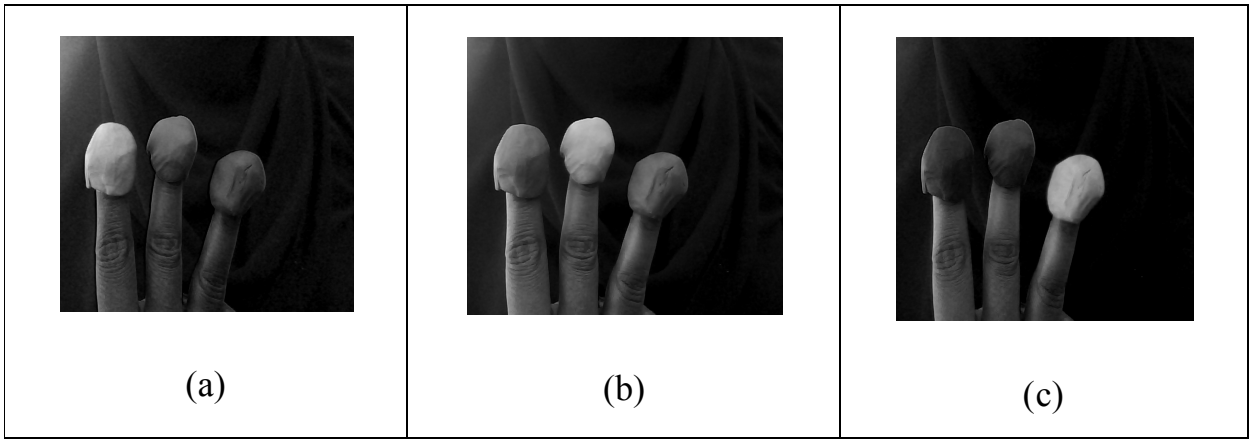


Figure (4.7): The Blue, Green, and Red Component of the Snapshot View

The Intensity is then extracted from the color component images to have a new image that contains only the pure color with different saturation. The result is shown in figure (4.8). The sub-figures a, b, and c shows the pure blue, green, and red images respectively.

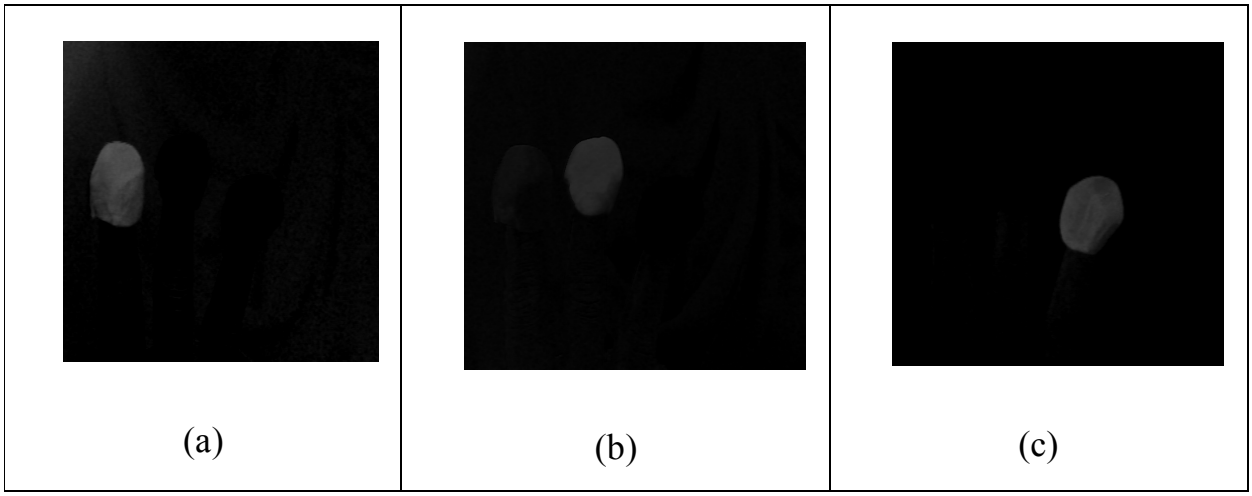


Figure (4.8): The Color Component Intensity Subtracted Images

From figure (4.8) it is observed that the color objects are not very clear because the image contains different saturation of the color object. To clarify the object thresholding segmentation is used with thresholding value equals 0.11 for red, 0.1 for blue and 0.1 for green. The results are shown in figure (4.9), the sub-figures a, b, and c shows the segmented blue, green, and red objects respectively.

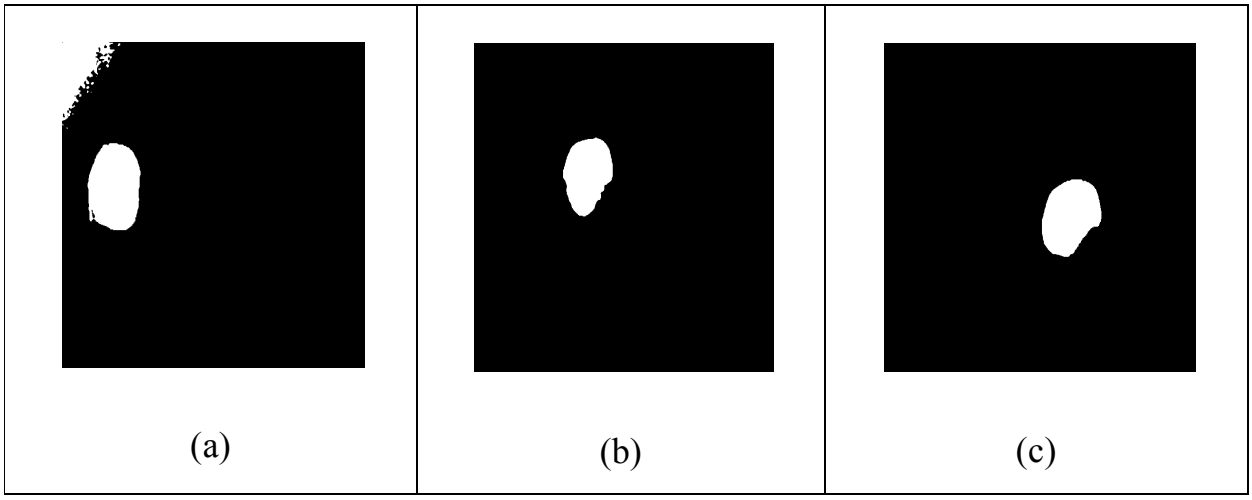


Figure (4.9): The Segmented Color Objects

From the result shown in figure (4.9) it can be observed that the thresholding process replaces all pixels in the image with value greater than threshold level with the value 1 (white) and replaces all other pixels with the value 0 (black).

Chapter Five

Conclusion and recommendation

5.1 Conclusion

The wireless computer mouse based on finger-tip tracking aims to add a vision capability to the computer operation system, so as to be able to control the operation of the computer mouse in a pc screen using finger-tip. To achieve this objective color object detection principles was used.

Using the computer mouse for a long time results in damaging the arm muscle thus, producing pain in the human body. In addition to that, people who lost their hand due to any accidents and substitute it with an artificial arm cannot use the computer mouse in a proper way because of their delay in controlling fingers.

In this project MATLAB was used to design a program to add a vision capability to the computer operation system (windows), so as to control the operation of the computer mouse in a pc screen using finger-tip.

The program was tested and it works well, and performs computer mouse actions properly. The results show that when taking a high resolution captured image from the front-view of the pc, the program processing time increase and this slow down the tracking process. And when taking a low resolution captured image it will be smaller that the computer screen resolution and the affect appear in the smoothing of the mouse movement and it will be difficult to locate the mouse position for icons clicking. The performance increases when working in part of the screen

5.2 Recommendation

This thesis discussed a wireless computer mouse based on finger-tip tracking under operation system (windows). For future work the following recommendations are a hereby made

- In this thesis, the system setup was prepared manually. A better result can be obtained if the system is self adaptive and more intelligent.
- For future work increase the function that can be performed by the computer mouse so as to be used as a touch screen if it is integrated with a hand gesture algorithm.

References

[1] Hojoon Park, 2000," A Method for Controlling Mouse Movement using a Real-Time Camera", ACM.

[2] Liang Chen, 2005," 3D Mouse for Interacting with Virtual Objects", ISCAS.

[3] Easel, 1991,"Programming Tools", INFO WORLD MA.

[4] BA Myers, 1994, "State of Art in User Interface Software Tools", Ablex.

[5] Borenstein,N,S,1985,"The Evaluation of Text Editor", ACM.

[6]Derrett, Heckel, 2004," Conclusions from the User Interface Design of a Music Appliance", IEEE International Conference.

[7]Schmidt, 2000, "Implicit Human–Computer Interaction through Context", Personal Technologies.

[8] Stefan Poslad, 2009."Ubiquitous Computing", WILEY.

[9] Rafael C. Gonzalez ,Richard E. Woods,2001," Digital image processing", Prentice Hall.

[10]Pasi F.,2003,"Lecture Note." Digital image processing",Universitiy of Eastern Finland.

Appendix

Code:

```
%%% manual camera setting
% set the camera in a flip position right left
%%%%%video capture setup
vid=videoinput('winvideo',1);
triggerconfig(vid,'manual');
set(vid,'FramesPerTrigger',1);
set(vid,'TriggerRepeat',inf);
start(vid);
%%%% java library
import java.awt.Robot;
import java.awt.event.*;
mouse = Robot;
StopCount=0;
%%%% starting program loop
while(StopCount<30)
   trigger(vid);
   data=getdata(vid,1);
%%%% YCbCr converted to RGB
   RGB = ycbcr2rgb(data);
%%%% extracting red, green,blue component
```

```
    dataR=RGB(:,:,1);
    dataG=RGB(:,:,2);
    dataB=RGB(:,:,3);
    RGBG=rgb2gray(RGB);
    dataSubR=imsubtract(dataR,RGBG);
    dataSubB=imsubtract(dataB,RGBG);
    dataSubG=imsubtract(dataG,RGBG);
%%%%threshold and filtering
    dataBinR=im2bw(dataSubR,.11);
    dataBinR= medfilt2(dataBinR,[3 3]);
%%%% extracting xand y position and moving mouse
        dataFR=bwareaopen(dataBinR,100);
        [LR numR]= bwlabel(dataFR,8);
        if numR>=1
          stats = regionprops(LR,'Centroid');
          cent = stats(1).Centroid;
          mouse.mouseMove(cent(1),cent(2));
         else
          StopCount=StopCount+1;
        end
%%%% mouse left click action (extracting blue object)
```

```
    dataBinB=im2bw(dataSubB,.1);
    dataBinB= medfilt2(dataBinB,[3 3]);
    dataFB=bwareaopen(dataBinB,100);
       [LB numB]= bwlabel(dataFB,8);
     if numB==1
           mouse.mousePress(InputEvent.BUTTON1_MASK);
           mouse.mouseRelease(InputEvent.BUTTON1_MASK);
     end
%%%% mouse right click action (extracting green object)
     dataBinG=im2bw(dataSubG,.1);
     dataBinG= medfilt2(dataBinG,[3 3]);
     dataFG=bwareaopen(dataBinG,100);
     [LG numG]= bwlabel(dataFG,8);
    if numG==1
      mouse.mousePress(InputEvent.BUTTON3_MASK);
      mouse.mouseRelease(InputEvent.BUTTON3_MASK);
    end
%figure,imshow(RGB);figure,imshow(RGBG);
%figure,imshow(dataR);figure,imshow(dataSubR);figure,imshow(dataBinR);figur
e,imshow(dataFR);
%figure,imshow(dataB);figure,imshow(dataSubB);figure,imshow(dataBinB);figur
e,imshow(dataFB);
```

```
%figure,imshow(dataG);figure,imshow(dataSubG);figure,imshow(dataBinG);figur
e,imshow(dataFG);
flushdata(vid);
end
delete(vid);
clear vid;
```