ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

NEUMANN JÁNOS

FACULTY OF INFORMATICS

THESIS

OE-NIK	Student's name:	**SIBOUH, MOHAMMED**
2022	Student's registration number:	**T/008881/FI12904/N**

Óbuda University
John von Neumann Faculty of Informatics
Institute for Cyber-physical Systems

DIPLOMA THESIS
TOPIC DESCRIPTION

Student name: **Mohammed Sibouh**
Registration number: T/008881/FI12904/N
Neptun's code: [redacted]

Title of diploma thesis:

How to improve human-computer interaction
Az ember és a számítógép interakciójának javítása

Internal supervisor: Péter Fésüs
External supervisor

Deadline: 15th of December, 2022.

Topic description:

Design and implement a system of **Hand gesture recognition** with the following attributes and stages:

- The skin segmentation stage should provide to segment the skin foreground object.
- The motion detection stage should use to detect the moving object.
- A color distribution model should use to execute hand tracking.
- Non-skin-colored objects or objects without long-time movements should be eliminated to avoid false alarms.
- Arm to palm intercept should be removed and a convex hull algorithm have to be used to complete hand gesture recognition.

The diploma thesis should cover:

- the precise description and detailed analysis of the problem to be solved,
- possible implementation methods and solutions,
- the detailed system plan and its justification,
- presentation of the technologies used in the implementation,
- a description of the implementation process, work phases and experiences,
- test methods, test plans and test results,
- the application and further development possibilities of the completed system,
- The source code of the developed system and the prepared documentation on a CD or DVD attachment.

ÓBUDAI EGYETEM Neumann János Informatikai Kar 1. Budapest

Ph.

.....
Dr. Rita Fleiner
head of institute

Term of limitation: **15th of December, 2024.**

The diploma thesis is adequate and can be submitted:

Péter Fésüs

.....
external supervisor

.....
internal supervisor

STUDENT'S DECLARATION

I the undersigned student hereby declare that this thesis is the result of my own work; I have disclosed the references and tools used in an identifiable manner.

The results indicated in my thesis completed may be used by the university and the institution announcing the task for their own purposes free of charge.

Dated: Budapest, 24.04. 2022

SIBOUH Mohammed
student's signature

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Faculty of Informatics

CONSULTATION LOG

Thesis II (2021/2022)

Student's name:	Neptun code:	Specialty:
SIBOUH Mohammed	[redacted]	**Computer Science**

Phone number: Mailing address (e.g. domicile):

[redacted]

Degree project / thesis[1] title in Hungarian:

How to improve human-computer interaction

Degree project / thesis[2] title in English:

Az ember és a számítógép interakciójának javítása

Institutional supervisor: External supervisor:

Fésüs Péter

Please write the data in printed capital letters!

Occ.	Date	Content	Signature
1.	08/02/2022	General project context	
2.	10/02/2022	Existing methodologies and approaches	
3.	22/02/2022	Analysis of gesture and recognition	
4.	24/02/2022	Hand gesture detection and recognition	

is required to be countersigned by either supervisor on a total of 4 occasions of consultation.

The student has complied with the requirements of the subject titled Thesis 1 / Thesis 2 / Thesis 3 / Thesis 4, and is allowed to proceed for reporting / defense.

Dated: Budapest, 30/04/2022

Péter Fésüs

institutional supervisor

CONSULTATION LOG

Thesis III (2021/2022)

Student's name:	Neptun code:	Specialty:
SIBOUH Mohammed	[redacted]	**Computer Science**

Phone number: Mailing address (e.g. domicile):

[redacted]

Degree project / thesis[1] title in Hungarian:

How to improve human-computer interaction

Degree project / thesis[2] title in English:

Az ember és a számítógép interakciójának javítása

Institutional supervisor: External supervisor:

Fésüs Péter

Please write the data in printed capital letters!

Occ.	Date	Content	Signature
1.	08/03/2022	Identifying the selected methodology	
2.	10/03/2022	Proposed hand gesture recognition system	
3.	22/03/2022	Describing the implementation design	
4.	24/03/2022	Review I	

is required to be countersigned by either supervisor on a total of 4 occasions of consultation.

The student has complied with the requirements of the subject titled Thesis 1 / Thesis 2 / Thesis 3 / Thesis 4, and is allowed to proceed for reporting / defense.

Dated: Budapest, 30/04/2022

Péter Fésüs

institutional supervisor

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Faculty of Informatics

CONSULTATION LOG

Thesis IV (2021/2022)

Student's name:	Neptun code:	Specialty:
SIBOUH Mohammed	[redacted]	**Computer Science**

Phone number: Mailing address (e.g. domicile):

[redacted]

Degree project / thesis[1] title in Hungarian:

How to imporve human-computer interaction

Degree project / thesis[2] title in English:

Az ember és a számítógép interakciójának javítása

Institutional supervisor: External supervisor:

Fésüs Péter

Please write the data in printed capital letters!

Occ.	Date	Content	Signature
1.	05/04/2022	Started the practical part	Fésüs Péter
2.	07/04/2022	Installation of the necessary software and packages	Fésüs Péter
3.	19/04/2022	Collecting the results and final system simulation	Fésüs Péter
4.	21/04/2022	Review II and Final review and presentation	Fésüs Péter

is required to be countersigned by either supervisor on a total of 4 occasions of consultation.

The student has complied with the requirements of the subject titled Thesis 1 / Thesis 2 / Thesis 3 / Thesis 4, and is allowed to proceed for reporting / defense.

Dated: Budapest, 30/04/2022

Fésüs Péter

Péter Fésüs

institutional supervisor

Table of Contents

1. LIST OF ABBREVIATIONS

ASL	**A**merican **S**ign **L**anguage
ACM	**A**ssociation for **C**omputing **M**achinery
CHI	**C**omputer-**H**uman **I**nteraction
CNN	**C**onvolutional **N**eural **N**etworks
DP	**D**ynamic **P**rogramming
FACs	**F**acial **A**ction **C**oding **S**ystem
FFT	**F**ast **F**ourier **T**ransform
FCM	**F**uzzy **C**-**M**eans Algorithm
GUI	**G**raphic **U**ser **I**nterface
GMM	**G**eometrical **M**athematical **M**odel
HSV	**H**erpes **S**implex **V**irus
HCI	**H**uman-**C**omputer **I**nteraction
HMI	**H**uman-**M**achine **I**nteraction
HMMs	**H**idden **M**arkov **M**odels
IREX	**I**nteractive **Re**habilitation and **E**xercise System
KFM	**K**ohohen **F**eature **M**ap
LVQ	**L**earning **V**ector **Q**uantization
MMI	**M**an-**M**achine **I**nteraction
MIT	**M**assachusetts **I**nstitute of **T**echnology
MLP	**M**ulti**l**ayer **P**erceptron
PDAs	**P**ersonal **D**igital **A**ssistants
ROI	**R**egion **o**f **I**nterest
UCD	**U**ser-**C**entered **D**esign
VUI	**V**oice **U**ser **I**nterface
VSD	**V**alue **S**ensitive **D**esign

2. DEDICATION

We dedicate this modest work to all who near and far have me provided moral and physical support for the achievement of this work:

To our parents for their support during all of us study and who have not ceased to lavish us with their loves. To our brothers and sisters will find here the expression of our respect and love.

To all our department colleagues and in particular those of the MSc Computer Science Specialist To everyone we know and with whom we exchange feelings of Friendship of Love and Respect.

3. ACKNOWLEDGEMENTS

I'd like to express my inner feelings, to everyone who has helped, directly or indirectly, with this modest project.

First of all, I'd like to express my sincere gratitude to the professor and my thesis research supervisor, **Mr. Péter Fésüs**:

- **For** his complete trust in me throughout the thesis research.
- **For** his wise and far-sighted guidance that led me to well this modest work.
- **For** its guidance and relevant remarks.
- **For** his encouragement and ability to communicate and explain and share information.

Finally, our warmest thanks will also extend to our faculty and our university **"John von Neumann Faculty of Informatics | Óbuda University"** and to all our **professors** and **staff** who supported us during these two years.

4. ABSTRACT

The primary purpose of Human-Computer Interaction is to increase user-computer interaction by making computers more attentive to user demands. Today, the human-computer interface with a personal computer is not restricted to keyboard and mouse interaction. Human interaction is mediated by various sensory modes such as gesture, speech, facial and body expressions. Natural interaction with the system is becoming increasingly important in many fields of Human-Computer Interaction.

A gesture-controlled mouse is a device that allows us to move the mouse by directing it with our wrist and turning the direction of the mouse via it. We can move and alter the direction of our mouse with the aid of an accelerometer, and we can also change the speed of the mouse with the use of an accelerometer. This revolutionary technology allows us to operate any gadget by moving our wrists. After improving it, we may construct various gadgets that are highly beneficial to humans and can make a difference in our lives. It may be employed in various settings and will be a part of the future. The gesture will be vital, as our body motions will be crucial for operations.

Hand gesture recognition is a system that can recognize hand gestures in real-time video. Hand gestures are classified according to their subject matter. One of the challenging jobs in this study is the design of hand gesture recognition, which involves two significant problems. The first and important step is to identify the existing of a hand. Another issue is creating a sign that can only be utilized one hand at a time. This study focuses on how a system detects, recognizes, and interprets hand gesture recognition using computer vision with problematic parameters such as posture, orientation, position, and scale variability.

To achieve that, the following report describes a methodology to develop a design of hand gesture recognition system for human-computer interaction using Python and the cross-platform image processing module **OpenCV**, and **implementing the mouse actions** using **Python-specific library PyAutoGUI.**

This **Report** contains a **theoretical part** that present the whole theme of the **design** and the existing **methodologies and approaches**, and the **practical part** that describes the different steps to **implement** the suggested design.

5. CHAPTER I – GENERAL INTRODUCTION

Human-computer interaction (HCI) is the study of computer architecture and application., focusing on the interfaces between people (users) and computers. HCI researchers study how humans interact with computers and improve techniques that help humans to interact with computers in novel ways.

Human-computer interaction is a research field that combines computer science, behavioral sciences, design, media studies, and several other disciplines. Even though the authors used the term for the first time in 1980, and the first known use was in 1975, Stuart K. Card, Allen Newell, and Thomas P. Moran popularized it in their seminal book The Psychology of Human-Computer Interaction, published in 1983. The term implies that, in contrast to other tools with limited uses (such as a hammer, which helps drive nails but not much else), a computer has many uses, which occur as an **open-ended-dialog** between the user and the computer.

Humans-interact with computers in various ways, and the interface that exists between humans and the computers they use is critical to facilitating this interaction. Desktop applications, web browsers, handheld computers, and computer kiosks all use today's popular graphical user interfaces (GUI). Voice user interfaces (VUI) are used in speech recognition and synthesis systems and emerging multi-modal. Gestalt User Interfaces (GUI) enables humans to interact with embodied character agents in ways that other interface paradigms cannot.

Today, human-computer interaction with a personal computer is not limited to keyboard and mouse interaction. Human interaction is mediated by several sensory modalities such as gesture, word, face, and bodily emotions. Natural interaction with the system is becoming increasingly crucial in many aspects of Human-Computer Interaction.

The human-computer interaction field has grown in interaction quality and branching in its history. Instead of designing traditional interfaces, various research branches have focused on concepts such as multimodality rather than unimodality, intelligent adaptive interfaces rather than commands/action-based ones, and finally, active rather than passive interfaces.

Human-computer interaction is also defined by the Association for Computing Machinery (ACM) as **" a discipline concerned with the design and evaluation, and implementation of interactive computing systems for the human use, as well as the study of major phenomena surrounding them "[1]**. Securing user satisfaction is an important aspect of HCI (or simply End-User Computing Satisfaction). "Because human-computer interaction studies the interaction of a human and a machine, it draws on supporting knowledge from both the machine and the human side." Operating systems in computer graphics, techniques, programming languages, and development environments are relevant on the machine side. Social sciences, graphic and industrial design disciplines, communication theory, linguistics, cognitive psychology, social psychology, and human factors such as computer user satisfaction are human factors.

Engineering and design methods, of course, are essential." People from various backgrounds contribute to the success of HCI due to its multidisciplinary nature. HCI is also known as human-machine interaction (HMI), man-machine interaction (MMI), and computer-human interaction (CHI). Human-machine interfaces that are poorly designed can cause several unexpected issues. The Three Mile Island nuclear meltdown accident is a classic example of this, with investigations concluding that the design of the human-machine interface is at least partly responsible for the disaster. Similarly, accidents in aviation have occurred as a result of manufacturers' decisions to use non-standard flight instrument or throttle quadrant layouts: even though the new designs were proposed to be superior in basic human-machine interaction, pilots had already ingrained the "standard" layout, and thus the conceptually good idea had unintended consequences**. CMU's**

Human-Computer Interaction Institute, **Georgia Tech's GVU Center**, and the University of Maryland's Human-Computer Interaction Lab are top academic research institutions.[2]

6. CHAPTER II – GENERAL PROJECT CONTEXT

6.1. Problem statement

Because of continuous innovation and breakthroughs in computer software and hardware technologies in recent decades, social life and information technology have a close relationship in the twenty-first century. In the future, consumer electronics product interfaces (e.g., smartphones, games, and infotainment systems) will have more and more functions and be more complex. The question of how to create a user-friendly human-machine interface

(**Human Machine Interaction/Interface, HMI**) for each consumer electronics product has become critical. The traditional electronic input devices, such as the mouse, keyboard, and joystick, remain the most common means of interaction. However, this does not imply that these devices are the most natural and convenient input devices for most users. Gestures have been used as a means of communication and interaction between people since ancient times. Before the invention of language, people could easily express their ideas through gestures.[6]

Gestures are still used naturally by many people today, and they are important and natural ways of interaction for deaf people. In recent years, gesture control has emerged as a new development trend for many human-based electronics products, including computers, televisions, and games.[6]

6.2. The main goal of the project

The use of a physical controller for human-computer interaction, such as a mouse or keyboard, hampers the natural interface since it creates a strong barrier between the user and the computer. However, the goal of a **gesture recognition-based Human-Computer Interaction** control system is to **solve** the existing **problems** of **lower precision and poor real-time ability** in gesture recognition algorithms.[3]

"Hand Gesture Recognition based on Camera" is based on the image processing concept. In the last year, there has been a lot of research on gesture recognition using Kinect sensors on HD cameras, but cameras and Kinect sensors are more expensive. This **project aims** to reduce the cost and **improve** the **robustness** of the proposed system by using a **simple web camera**.

With the advancement of ubiquitous computing, current user interaction approaches such as keyboard, mouse, and pen are no longer adequate. Because of these devices' limitations, the available command set is also limited. **Hands** can be used **directly** as an input device to provide natural interaction.[4]

6.3. Design of Human-Computer Interaction

6.3.1. Principles

The user interacts directly with hardware for human input and output, such as displays, via a graphical user interface, for example. Using the provided input and output (I/O) hardware, the user interacts with the computer via this software interface.

Hardware and Software must be matched so that user input is processed quickly enough and computer output latency is not disruptive to the workflow.

When assessing an existing user interface or building a new user interface, keep the following experimental design concepts in mind:

- **Early emphasis on the user(s) and task(s):** Determine how many users are required to complete the task(s) and who the suitable users should be; someone who has never used the interface and will not use it in the future is most likely not a genuine user. Define the task(s) that the users will complete and how frequently the task(s) must be completed.
- **Empirical measurement:** Early on, test the interface with real users who interact with it daily. Keep in mind that the results may vary depending on the user's performance level and may not be an exact representation of regular human-computer interaction. Determine quantitative usability specifics such as the number of users doing the activity(s), the time required to complete the tasks, and the number of mistakes committed throughout the tasks.
- **Iterative design:** After deciding on the users, tasks, and empirical measures to include, proceed with the iterative design processes listed below:

1. Design the user interface
2. Test
3. Analyze results
4. Repeat

6.3.2. Methodologies & Approaches

Since the field's inception in the 1980s, many **approaches** detailing strategies for **human-computer interface design** have evolved. The majority of design approaches are based on a model of how users, designers, and technical systems interact.

Hand gesture recognition has been accomplished using both **non-vision and vision-based techniques**. The detection of finger movement using a pair of wired gloves is an example of a non-vision-based technique. In general, vision-based techniques are more natural since they do not involve hand devices. The literature divides hand gestures into two types: static and dynamic gestures. Static hand gestures are those in which the location and orientation of the hand in space do not vary over time. Dynamic gestures occur when there are changes within a specific time frame. Waving the hand is an example of a dynamic hand gesture, whereas combining the thumb and forefinger to create the "Ok" symbol is an example of a static hand gesture.[8]

Early approaches, for example, portrayed users' cognitive processes as predictable and quantifiable, encouraging designers to refer to cognitive science results in areas like memory and attention when creating user interfaces. Modern models emphasize ongoing input and discussion between users, designers, and engineers. They advocate for technical systems to be wrapped around the experiences consumers want rather than the user experience being wrapped around a completed system.[2]

Activity theory: is a type of HCI theory used to describe and investigate the context in which humans engage with computers. Activity theory provides a framework for reasoning about activities in various situations, analytical tools in the form of checklists of items to examine for researchers, and influences interface design from an activity-centric perspective.[2]

User-centered design (UCD): is a current, widely applied design concept based on the premise that people must be at the core of any computer system's design. Users, designers, and technical practitioners collaborate to identify the user's goals, requirements, and limits and develop a solution that meets these factors. Ethnographic studies of the contexts in which users will interact with the system are frequently used to influence user-centered design initiatives. This method is comparable to but not the same as participatory design, which stresses the ability of end-users to engage through collaborative design sessions and workshops.[2]

Principles of user interface design: simplicity, Tolerance, visibility, affordance, consistency, structure, and feedback are seven user interface design concepts that may be addressed at any point throughout the design of a user interface in any order.[2]

Value Sensitive Design (VSD): is a technique of developing technology that considers the values of individuals who use the technology directly and those who are affected by the technology, either directly or indirectly. VSD employs an iterative design approach that includes three forms of research: conceptual, empirical, and technological. Conceptual investigations seek to comprehend and articulate the many stakeholders of the technology and their values and any value conflicts that may occur for these stakeholders due to their usage of the technology.[2]

6.3.3. Human-Computer-Interface

The human-computer interface is the point of contact between the human user and the computer. **The interaction loop** is described as the flow of information between a human and a computer. **The interaction loop** contains multiple components, including:

Visual Based: Visual-based human-computer interaction is perhaps the most common field of HCI study.[2]

Audio-Based Interaction: Another critical area of HCI systems is audio-based interaction between a computer and a person. This section deals with data obtained from various audio signals.[2]

Task environment: The circumstances and objectives imposed on the user.[2]

Interface regions: non-overlapping interface areas involve human and computer operations unrelated to their contact. Meanwhile, the overlapping sectors are only concerned with the procedures that govern their interaction.[2]

Input flow: The flow of information that starts in the task environment when the user has a task that necessitates the usage of a computer.[2]

Output: The flow of information that starts in the machine environment is output.

Feedback loops: Loops via the interface that assess, regulate, and validate processes as they flow from the person to the computer and back.[2]

Fit: The fit between the computer design, the user, and the job to optimize the human resources required to complete the activity.[2]

7. CHAPTER III – ANALYSIS OF GESTURE DETECTION & RECOGNITION

7.1. Hand Gesture Detection and Recognition

7.1.1. Hand Detection

Hand **detection** is an important pre-processing step in many computer vision applications involving **human hands**, such as hand posture estimation, hand gesture **identification**, human activity **analysis**, etc. However, because of the complicated appearance diversities of dexterous human hands (e.g., diverse hand forms, skin colors, illuminations, orientations, scales, etc.) in color photographs, correctly distinguishing several hands from cluttered sceneries remains a difficult issue.[9]

Traditional hand detection methods primarily utilize **low-level image features** such as **skin color** and **shape for hand region detection**. Nowadays, convolutional neural networks (CNN) based detection approaches are more robust and accurate due to the deep discriminative features learned. However, compared to familiar objects, human hands are highly articulated, appearing in various orientations, scales, shapes, skin colors, and sometimes partial occlusions. Therefore, reliably detecting multiple human hands from unconstrained cluttering scenes remains a challenging problem.

The following variables make the hand detection task challenging to solve:

Image plane and posture variations:

The **rotation**, **translation**, and **scaling** of the **camera** stance or the hand itself cause the hands in the image to fluctuate. The rotation can occur both within and outside of the plane.

Skin Color and Other Structure Components:

The look of a hand is greatly influenced by skin color, size, and the presence or absence of extra characteristics such as hairs on the hand.

Background and lighting conditions:

The features of the light source, as illustrated in the figure, influence the look of the hand. The backdrop, which defines the profile of the hand, is also significant and should not be overlooked.[10]

Figure 1: Background and lighting challenges

7.1.2. Hand recognition

Hand detection and recognition have been essential topics in computer vision and image processing during the last 30 years. Significant progress was made in these sectors, and several techniques have

been presented. The usual method of a fully automated hand gesture recognition system, on the other hand, is depicted in the figure below:

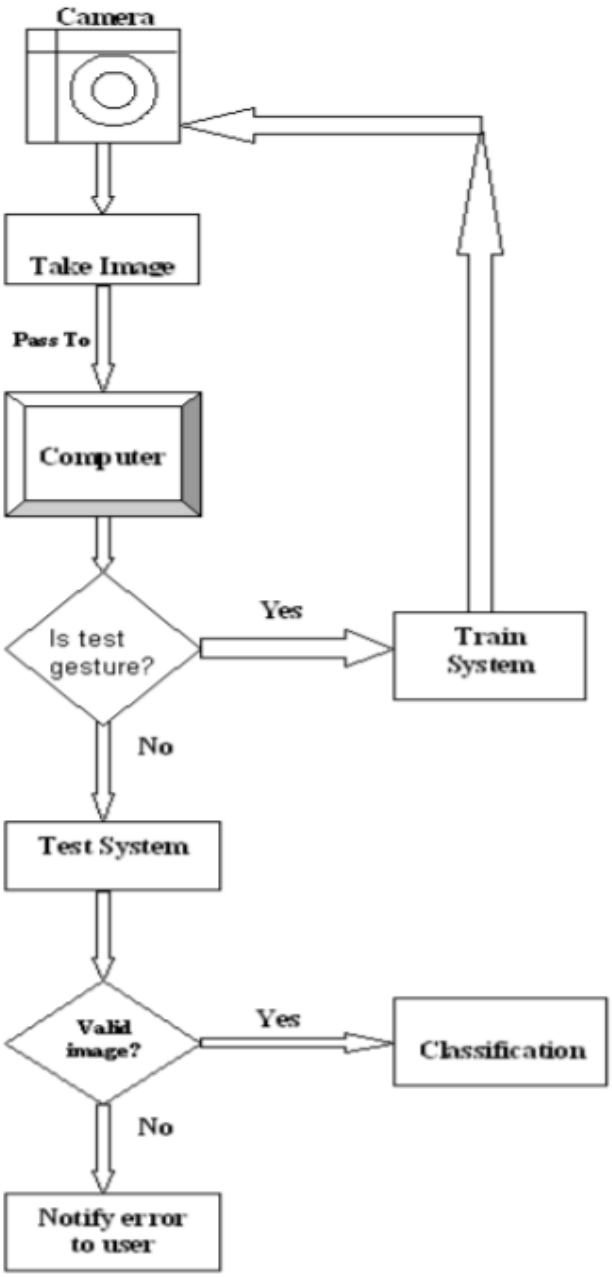


Figure 2: Hand Gesture Recognition Flow Chart

7.1.3. Motivational aspects

Biometric technologies use physical and behavioral characteristics such as fingerprints, expression, face, hand gestures, and movement. These characteristics are subsequently analyzed by sophisticated machines for detection and recognition and hence employed for security. Unlike standard security methods such as passwords and security cards, which can be easily lost, duplicated, or stolen, biometric traits are unique to people and cannot be replaced or altered.

Among the biometric sector, hand gesture recognition is gaining more and more attention because of their demand regarding security for law enforcement agencies and private sectors such as surveillance systems.

In a video conferencing system, it is necessary to automatically operate the camera so that the current speaker is constantly in focus. One easy solution is to steer the camera using sound or primary indications like motion and skin tone.

Quick and efficient hand gesture detection algorithms are necessary to construct completely automated systems that interpret the information contained in photos.[10]

7.1.4. Literature Review

There **are three types of hand gesture recognition research**. **First, a "Glove-based Analysis"** attaches a sensor to a glove, either mechanical or optical, to convert finger flexing into electrical impulses for hand posture evaluation, as well as an extra sensor for hand position. This sensor, commonly acoustic or magnetic, is connected to the glove. Some apps use a look-up table software toolset to identify hand position.

The second approach is "Vision-based Analysis," which assumes that humans obtain information from their environment, and this is perhaps the most challenging strategy to apply satisfactorily. So far, several alternative implementations have been tested. One option is to use a 3-D representation of the human hand. Several cameras are coupled to this model to establish matching hand photos, palm position, and joint angles to conduct hand gesture categorization. Lee and Kunii created a hand gesture analysis system with 27 degrees of freedom based on a three-dimensional hand skeleton model. The integrated five fundamental limitations based on human hand kinematics to simplify the model parameter space search. Specially labeled gloves were used to facilitate model matching.

The third implementation, "Analysis of Drawing Gesture," uses a stylus as an input device. The examination of drawings leads to the identification of written text. Mechanical sensing work has been employed at a high level for hand gesture identification indirect and virtual environment manipulation. Mechanically detecting hand position presents several issues, including electromagnetic noise, dependability, and accuracy. Visual sensing gesture interaction can be practical, yet it is the most challenging challenge for robots.[10]

7.1.5. Lighting

A careful choice of lighting greatly simplifies distinguishing skin pixels from background pixels. According to Ray Lockton, if the lighting is consistent across the camera's view, the effects of self-shadowing can be minimized.[10]

7.1.6. Camera Orientations and Distance

It is critical to be mindful of camera direction to select a background easily**.** Pointing the camera at a wall or the floor is a better and more successful technique. Because the camera was pointing downwards, the lighting was standard; the intensity of the light would be more robust, and the shadowing effects would be lesser. The camera's distance from the hand should be such that it mostly captures the sweeping gesture. There is no influence on the system's accuracy of whether the image is zoomed; the concept is to extensively cover the entire hand region.[10]

7.1.7. Background Selection

Another critical consideration is to increase distinction, which means that **the background color** should be as different as possible from the skin color. The floor color utilized that is most recommended was black, and this color was chosen because it provided the least amount of self-shadowing compared to other background colors.[10]

The following are the many recognition techniques investigated:

7.1.8. Pen-Based Gesture Recognition

Recognizing gestures from two-dimensional input devices like a pen or **mouse** has long been investigated. Light-pen gestures, for example, were employed in the early Sketchpad system in 1963. Since the 1970s, some commercial systems have employed pen motions. There are instances of gesture recognition in document editing for air traffic control and design jobs like editing splines.

Recently, systems like the Ogi Quick Set system have proved the potential of using pen-based **gesture detection** in conjunction with speech recognition to operate a virtual world. Quick Set understands 68 pen gestures, including map symbols, editing gestures, route indications, area indicators, and taps.

Personal Digital Assistants (PDAs) have been commercially available for several years, beginning with the Apple Newton and, more recently, the 3Com Palm Pilot and various Windows CE devices. Long and Rowe survey the problems and benefits of these gestural interfaces and provide insight for interface designers.

Although pen-based gesture detection appears to be promising for many HCI applications, it is dependent on the availability and proximity to a flat surface or screen. Techniques that allow users to walk about and engage in more natural ways are more in virtual worlds.[10]

7.1.9. Tracker-Based Gesture Recognition

Numerous commercially available tracking systems can be used for gesture recognition, primarily tracking eye gaze, hand gestures, and the overall body and its position. Each sensor in virtual environment interaction has its own set of strengths and drawbacks. Because gestural interface eye gaze can be helpful, I will concentrate on gesture-based input from tracking the hand and the body here.[10]

7.1.10. Data Gloves

People utilize their hands for various actions, such as communication and manipulation.

With approximately 29 degrees of freedom, hands and wrists are highly elegant, expressive, and convenient. The hand might be a control device with sophisticated input in various application domains, allowing real-time control with numerous degrees of freedom for complex tasks. Sturman examined task features and needs, hand action capabilities, and device capabilities, as well as critical difficulties in designing whole-hand input approaches.[11]

Sturman proposed a taxonomy of whole-hand input that divides input techniques into two categories: classes of continuous or discrete hand actions and interpretation of hand actions that can be direct, mapped, or symbolic.[11]

A given interaction task may be examined to determine which style is best suited to the task. Mulder gave an overview of hand gestures in human-computer interaction, including hand movement categorization, common hand gestures, and interface design.[12]

There is various commercially available equipment for measuring the location of the hand configuration and calculating the degree of precision, accuracy, and completeness. These gadgets include exoskeletons and "Data gloves," instrumented gloves mounted on the hand and figure. Data gloves have several advantages, including direct measurement of hand and finger parameters, data provision, high sampling frequency, ease of use, line of sight, low-cost version, and data translation independence.[13]

However, along with the benefits of data gloves, there are a few drawbacks, such as difficulties in calibration, a loss in the range of motion and comfort, noise in low-cost systems, and the high cost of precise systems. Furthermore, the user is required to wear burdensome equipment.

Many projects have incorporated hand input from data gloves for **"point, reach, and grab"** activities or more advanced gestural interfaces.

Latoschik and **Wachsmuth** present a multi-agent architecture for recognizing pointing movements in a multimedia application.[13]

Väänänen and **Böhm** created a neural network system that recognizes static movements and allows the user to teach the machine new gestures interactively. Böhm et al. expand their approach to dynamic gestures and data reduction using a Kohohen Feature Map (KFM).

Glove GRASP is a C/C++ class library developed by the University of Washington's HIT Lab that allows software developers to add gesture recognition capabilities to **SGI systems**, including user-dependent training and one- or two-handed gesture recognition. General Reality sells a commercial version of this technology.[13]

7.1.11. Body Suits

People may recognize patterns such as movements, activities, identities, and other body elements through the process of minuscule strategically placed dots on the human body. One technique to posture and human movement identification is to optically measure 3D position such as markers connected to the body and then reconstruct the temporally variable articulate structure of the body. This articulated sensing via position and joint angles is accomplished using electromechanical sensors. However, other systems require a little ball or dot to be put on the user's clothes. I favor the general term "bodysuits" for capturing bodily movements.

Bodysuits, like data gloves, offer advantages and cons. It offers reliable data at a high sample rate, but it is time-consuming and costly. Calibration is not easy. Several cameras are utilized by an optical system, which is often an offline data process, and the lack of cables and ropes are key drawbacks.

7.1.12. Head And Face Gestures

People use a variety of facial expressions. **FACS** is a technique devised by Ekman and Friesen for measuring facial movement and coding expression; this description serves as the foundation for many facial expression analysis systems.[15]

Zelinsky and **Heinzmann** created a real-time system for recognizing head and facial motions. They employed feature template tracking in a Kalman filter framework to distinguish thirteen head/face gestures.[15]

Essa and Pentland produced reliable estimations of facial motion by combining optical flow information with a physical muscle model of the face. This technique was also utilized to build Spatio-temporal motion-energy templates of the entire face for each expression, subsequently used for expression recognition.[13]

7.1.13. Hand And Arm Gestures

These two bodily parts (**Hand** and Arm) receive the most significant attention from individuals who research gestures; in fact, many references solely consider these two for gesture detection. The vast majority of automated recognition systems are designed to recognize deictic gestures

(pointing), iconic gestures (single signs), and sign languages (with a limited vocabulary and syntax). Some are parts of bimodal systems that are merged with voice recognition. Some provide accurate hand and arm configurations, while others create coarse motion.[7]

Stark and Kohler created the ZYKLOP **system to recognize hand** stances and **gestures** in **real-time**. The hand posture is characterized by segmenting the hand from the backdrop and extracting information such as shape moments and fingertip locations. The hand postures and their motion trajectory are used to perform temporal gesture recognition. The gesture catalog comprises a small number of hand positions, and a sequence of them forms a gesture.[13]

Maggioni and Kämmerer, on the other hand, described the Gesture Computer, which detected both hand motions and head movements. There were many interests in developing gadgets that can automatically translate different sign languages to help the deaf community. Starner, who utilized HMMs to detect a restricted vocabulary of ASL words, was the first to employ computer vision without needing the user to wear anything special. **Hand** and arm **gesture recognition** has been used in entertainment applications.[13]

Freeman created a real-time method for recognizing hand positions based on picture moments and orientation histograms, which he then used to interactive video games. Cutler and Turk presented a system that classifies data based on optical flow to allow youngsters to connect with lifelike individuals and perform virtual instruments.[13]

7.1.14. Body Gestures

Tracking whole-body movements, identifying body gestures, and recognizing human activities are all covered in this section. Action can be described over a considerably longer period than a gesture; for example, two persons meeting in an open place, pausing to converse, and then continuing on their journey may be deemed recognized activity. Bobick offered a taxonomy of motion comprehension: Movement – the atomic constituents of motion, Activity is a series of motions or static configurations. At the same time, Action is a high-level description of what is going on in context.[7]

Several organizations have employed the MIT Media Lab's Pfinder technology for body tracking and gesture detection. It creates a two-dimensional depiction of the body by employing statistical color and shape models. The body model serves as a powerful interface for applications such as video games, interpretative dance, navigation, and interaction with virtual characters.[13]

Lucente coupled Pfinder with speech recognition in Visualization Space, an interactive environment that allows users to handle virtual items and explore virtual environments.

Paradiso and Sparacino utilized Pfinder to create an interactive performance environment in which a dancer may generate music and images using their body motions - for example, hand and body gestures can cause rhythmic and melodic changes in the music.[13]

Human **motion analysis systems** in virtual environments might be valuable in medical rehabilitation and sports training. A system that has the same functionalities like the one developed by **Boyd** and **Little** to detect human gaits, for example, may be used to assess rehabilitation progress.[13]

Davis and Bobick employed a view-based technique to record and recognize human activity using "temporal templates," A single picture template preserves the most recent history of motion. This method was employed in the Kids Room system, an interactive, immersive, storytelling environment for children.[13]

There were a lot of interest in video surveillance and monitoring human activities in recent years. The W4 system, created at the University of Maryland, follows people and recognizes behavior patterns.[13]

7.2. Summary of the segmentation

Building an effective hand gesture detection system is critical for easy interaction between humans and machines. In summary, we conducted a comparative evaluation of multiple gesture recognition systems, focusing on the segmentation, feature detection, and extraction phases, which are critical for gesture modelling and analysis. Figure 4 depicts the identification rates of various recognition methods based on their presence in the paper.[17]

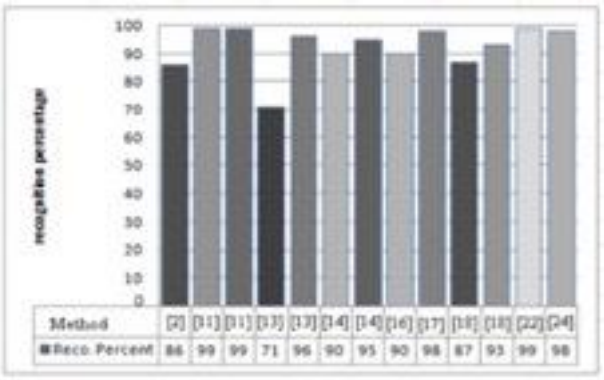


Figure 3: Recognition rates of gesture recognition systems

Using examples, the table summarizes the examination of the segmentation and features vector representation phases in hand gesture recognition systems. Figure 5 shows the three primary processes for a hand gesture recognition system: segmentation, feature representation, and recognition, as well as the methodologies employed in each phase. Finally, the table summarizes the benefits and drawbacks of various hand motion recognition systems.[17]

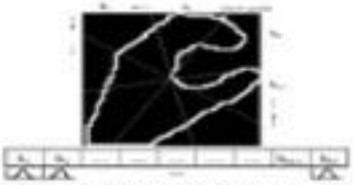
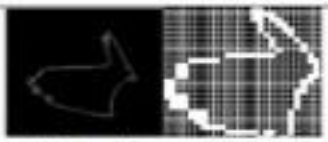
Method	Segmentation	Features Vector Representation
[2]	 A: Input image. B: Segmented image.	 A: Noiseless normalized hand. B: Image partitioned into 12 blocks feature vector.
[5]	 A: Boundary representation of the hand. B: Grid division. C: Normalized the grid.	 Feature vector histograms represented by dividing the image into number of regions N in a radial form.
[12]	 A: Input image. B: Segmented image.	 A: RC Angle. B: TC Angle. C: distance from the palm center.
[19]	-	 The input gesture represented by a histogram.
[23]	 A: Input image. B: Segmented image.	 A: Edged image. B: Features brightness division.
[23]	 A: Input image. B: Segmented image.	 Features vector formed by five distances from palm to all fingers and four angles between those distances.

Figure 4: Examples of segmentation and feature vectors representation of various systems.

[17] COMPARATIVE STUDY OF HAND GESTURE RECOGNITION SYSTEM.

https://www.slideshare.net/cscpconf/comparative-study-of-hand-gesture-recognition-system

Method	Segmentation	Features Vector Representation	Classifier
[2]	HSV threshold	13 parameters as a feature vector; the first parameter represents the ratio aspect of the bounding hand box and 12 parameters are the mean values of brightness pixels in the image	Fuzzy C-Means (FCM) algorithm
[5]	thresholding	Three features vector: Boundary Chord's size FFT, Boundary Chord's size, Boundary Chord's size histogram	MLP Neural Network / Dynamic Programming (DP) matching
[12]	YCbCr color space	Three angles from hand shape; RC Angle, TC Angle, Distance from the palm center.	Gaussian distribution
[13]	Threshold	13 data item for postures/ 16 data item for gestures	Back propagation network / Elman recurrent network
[16]	GMM and YCbCr space	Orientation quantization	HMM
[17]	thresholding	Template for posture/ hand location for gesture	structural analysis for postures / HMM for gestures.
[18]	Thresholding	Divide the posture image into Blocks of different odd sizes ((1x1), 3x3)….(23x23))	Euclidean Distance Metric.
[19]	thresholding	Representing hand gesture as an orientation histogram	Euclidean Distance Metric.
[22]	Segmented manually	Three different features group; Invariant Moments, K-curvature Geometric, Shape Descriptors.	Multilayer Perceptron Artificial Neural Network.
[23]	HSV and thresholding method	Nine numerical features; five distances from palm to all fingers and four angles between these distances	Learning Vector Quantization (LVQ).

Figure 5:Summary of segmentation, features vector representation, and recognition of hand gesture recognition systems

[17] COMPARATIVE STUDY OF HAND GESTURE RECOGNITION SYSTEM.

https://www.slideshare.net/cscpconf/comparative-study-of-hand-gesture-recognition-system

[13]	(1) Simple and active, and successfully can recognize a word and alphabet. (2) Automatic sampling, and augmented filtering data improved the system performance.	(1) Required long time for Learning; several hours for learning 42 characters, and four days to learn ten words. (2) Large difference in recognition rate for both registered and unregistered people. 98 % for registered and 77 % for unregistered people.
[16]	(1) Recognized both isolated and meaningful gestures for Arabic numbers.	(1) Recognition limited to numbers only.
[17]	(1)The system successfully recognized static and dynamic gestures. (2) Could be applied on a mobile robot control.	(1) The system doesn't reflect the dynamic gesture characteristics.
[18]	(1) Hand Saturation algorithm kept the hand object as a complete area.	(1)The edge method less effective since some important features was lost.

Method	**Advantages**	**Disadvantages**
[2]	(1) Speed and sufficient reliable for recognition system. (2) Good performance system with complex background.	(1) Irrelevant object might overlap with the hand. (2) Wrong object extraction appeared if the objects larger than the hand. (3) Performance recognition algorithm decreases when the distance greater than 1.5 meters between the user and the camera is
[5]	(1)The radial form division and boundary histogram for each extracted region, overcome the chain shifting problem, and variant rotation problem.	(1)The proposed method is susceptible to errors, especially in shapes like square and circular.
[12]	(1)Exact shape of the hand obtained led to good feature extraction. (2) Fast and powerful results from the proposed algorithm.	(1) System limitations restrict the applications such as; gestures are made with the right hand only, the arm must be vertical, the palm is facing the camera, background is plain and uniform.

[19]	(1) Simple, fast, and easy to implement. (2) Can be applied on real system and play games.	(1)Similar gestures have different orientation histograms. (2) Different gestures have similar orientation histograms. (3) Any objects dominate the image will be represented as orientation histograms.
[22]	(1) Three different groups of features are examined to decide the good performance group.	(1) The database samples, variation in scale, translation and rotation led to a misclassification in the.
[23]	(1) Used wool glove not costly	(1) The feature vector with highest score evaluated first by the classifier then the system selects that features hypothesis.

Figure 6: Advantages and disadvantages of hand gesture recognition systems.

[17] COMPARATIVE STUDY OF HAND GESTURE RECOGNITION SYSTEM.

https://www.slideshare.net/cscpconf/comparative-study-of-hand-gesture-recognition-system

8. CHAPTER IV – SELECTED METHODOLOGY

8.1. Introduction

this chapter will be about describing the selected methodology of the mentioned project, the reason for the selection, what are the requirements, And what are the expected results.

Data processing rates have recently grown considerably due to digitalization in every industry, with computers has evolved to the point where they can aid people in complicated jobs. However, input technologies appear to be a key bottleneck in accomplishing some activities by under-utilizing available resources and limiting utilization due to cost constraints. Compared to the optical mouse, which has a restricted range of connecting cable length and requires a surface to operate on, the wireless mouse isn't much use other than allowing for a desktop with fewer wires attached.

8.2. Application

Gesture recognition technology is rapidly gaining commercial acceptance as it improves and matures.[18] These are built on embedded vision algorithms that use cameras to recognize and interpret finger, hand, and body motions. However, gesture recognition as a user interface approach has many uses outside of consumer electronics. It is beneficial in our project since we can use our motions to control the mouse cursor. For example, gesture is viewed as a convenience-driven add-on function for managing the back hatch and sliding side doors in the car industry. Cameras already fitted in the vehicle's back for reversing and side mirrors for blind spot warning can be used for these extra capabilities.[19]

Gesture interfaces can also be effective in rehabilitation settings. IREX, for example, from Gestures, takes patients via interactive workouts that target specific body areas. There are additional non-traditional health-related applications for gesture recognition.[19] It is also valuable in 3-D modelling and may be used as part of sign language recognition.

8.3. Selected method _Hand Gesture

8.3.1. Gesture Controlled Technique

Typically, gestures are either **static** (requiring less processing complexity) or **dynamic** (requiring or requiring more computational complexity while being suited for a real-time setting). As we all know, various additional approaches for gesture recognition have been developed. However, we are employing hand gestures to move the pointer rather than the typical optical mouse in this case.

The input visual gesture pictures are identified as meaningful gestures based on gesture modeling and analysis during the recognition phase.

The categorization accuracy is determined by the proper selection of gesture parameters of features in the recognition process.

Edge detection and contour operators, for example, are not suited for gesture recognition since they may result in misclassification. Neural networks have been widely used in the extraction of hand shapes and the recognition of hand gestures. The Hidden Markov Model (HMM) demonstrated Excel's participation as a recognition tool. Clustering techniques played a role in the recognition process, with the fuzzy c-mean clustering algorithm being used to regulate robot mobility.[16]

8.3.2. Proposed Hand Gesture Recognition System

The overall system consists of the back end and the front end. The back-end system consists of three modules: A camera module, a Detection module, and an Interface module, as shown in Fig. 8, They are summarized as follows:

8.3.3. Description of the method

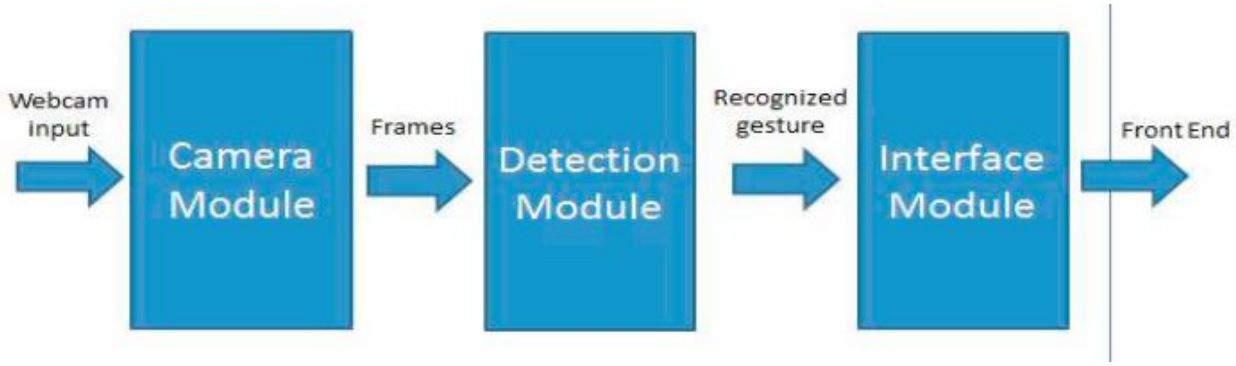


Figure 7: Back-end architecture

8.4. Camera module

This module is in charge of connecting and capturing input from various image detectors and sending this picture to the detection module for processing in the form of frames. Data gloves, hand belts, and cameras are prominent techniques of recording input. The proposed system makes advantage of the built-in camera, which is cost-effective for identifying both dynamic and static motions. The system is set up to accept input from an internal or USB camera, however this would need considerable human effort. The acquired picture frames are in the form of a video.[21]

8.5. Detection module

This module is in charge of image processing. The image produced from the camera module is subjected to several image processing techniques such as color conversion, noise reduction, and thresholding before being contour extracted. If the picture has flaws, convexity flaws are discovered, and the gesture is identified as a result. If no faults are found, the picture is classed using the **Haar cascade** to **detect** the **gesture**.

The detection module recognizes dynamic gestures as follows: If Microsoft PowerPoint is loaded with a slideshow enabled and the webcam detects palm movement for five continuous frames, the dynamic gesture swipe is identified. [21]

8.6. Interfacing module

This module is in charge of mapping the observed hand motions to their corresponding actions. These activities are then sent to the relevant application. Three windows make up the front end. The first window displays the video input from the camera together with the name of the gesture recognized. The contours detected in the input photographs are displayed in the second pane. The smooth thresholded version of the image is displayed in the third window. The benefit of including a threshold and contour window as part of the Graphical User.

The interface's purpose is to make the user aware of any backdrop inconsistencies that may impair the system's input so that they may adjust their laptop or desktop web camera to prevent them. This would lead to improved performance. [21]

8.7. Suggested method

As shown in Fig. 8, we suggest a marker-less gesture identification system that employs a very appropriate method.

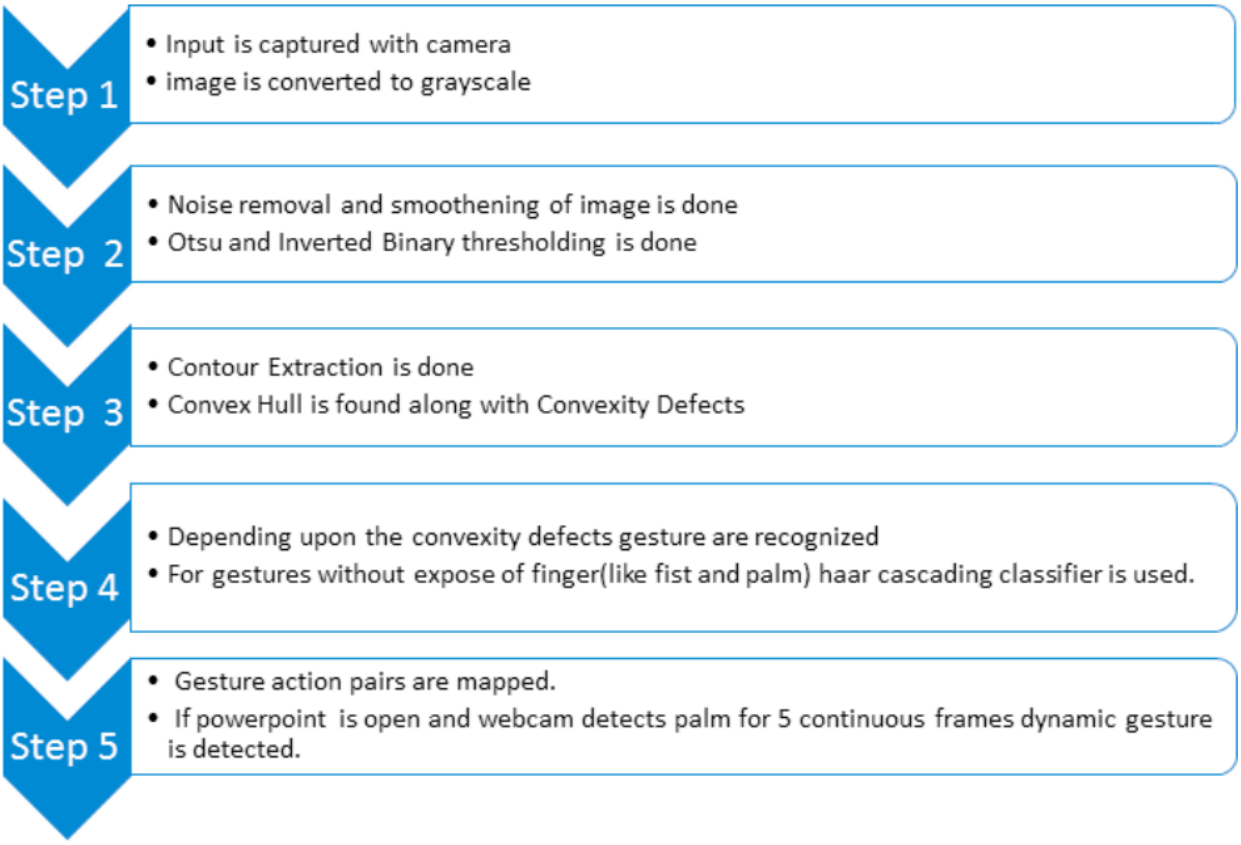


Figure 8: Proposed method for our gesture recognition system

8.7.1. Noise removal and Image smoothening

The RGB color space input picture is reduced to 300 By 300 pixels in size. The picture is then transformed to grayscale.

Noise in pictures is described as a random change of brightness or color information that is typically created during the webcam image collecting process. This noise is an unappealing picture that must be erased. A Gaussian filter is used to do this. The convolution of the Gaussian kernel with each point in the input array performs Gaussian filtering, and these are joined together to form the output arrays.[21]

The 2D Gaussian kernel can be represented mathematically, as shown in Eqn. 1.

$$G_0(x,y) = G_0 = (x,y) = A_e \frac{-(x-u_x)^2}{2\sigma_x^2} + \frac{-(y-u_y)^2}{2\sigma y^2} \qquad (1)$$

8.7.2. Thresholding

Then there's thresholding, which is a simple segmentation approach. The thresholding is used to convert a grayscale image to a binary image. Each pixel intensity value (I) is compared to the threshold value using the thresholding technique (T). If I>T, the pixel is replaced with a white one, and if IT, it is replaced with a black one. In our research, we employed a threshold value (T) of 127 to classify the pixel intensities in the grayscale picture. If any pixel in the picture meets the threshold

value, the pixel value is set to 255.Inverted Binary and Inverted Binary are the two methods of thresholding used. Otsu's Thresholding and Thresholding. The colors are inverted in Inverted Binary Thresholding, resulting in a white picture on a black backdrop. Eqn. 2 shows how to express this thresholding procedure.[21]

$$Dest(x,y) = \begin{cases} 0 \; if \; src\;(x,y) > T \\ max\,Val(255) otherwise \end{cases} \qquad (2)$$

So, if the pixel intensity **src(x, y)** is greater than the threshold value T, then the new intensity of the pixel is initialized to '0'. Otherwise, the pixels are set to **maxVal**.

Nobuyuki Otsu has given us Otsu's method. Clustering-based image thresholding is achieved from this method. Otsu binarization automatically calculates a threshold value from an image histogram for a bimodal image, which is an image whose histogram has two peaks.

In Otsu's method, we find the threshold that can minimizes the intra-class variance (the variance within the class), defined as a weighted sum of variances of the two classes as seen in Eqn. 3

Weights W_0 and W_1 represent the probability of the two classes separated by a threshold t, whereas weights σ_0^2 and σ_1^2 represent the variances.

The L histograms are used to calculate the class probability (0, 1) (t). This is seen in Eq 4.

$$\sigma_\omega^2(t) = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\,\sigma_1^2(t) \qquad (3)$$

$$\omega_0(t) = \sum_{i=0}^{t-1} \mathrm{p(i)} \;;\; \omega_1(t) = \sum_{i=0}^{L-1} \mathrm{p(i)} \qquad (4)$$

Otsu shows that minimizing the intra-class variance and maximizing inter-class variance generates the same results in Eqn. 5.

$$\sigma_b^2(t) = \sigma^2 - \sigma_\omega^2(t) = \omega_0(\mu_0 - \mu_T)^2 + \omega_1(\mu_1 - \mu_T)^2$$
$$= \omega_0(t)\omega_1(t)[(\mu_0(t) - \mu_1(t)]^2 \qquad (5)$$

This is expressed in terms of ω for probabilities and **μ**. While the class means $\mu_{0,1,T}$ (t) can be expressed as shown in Eqn.6.

$$\mu_0(t) = \sum_{i=0}^{t-1} i\,\frac{P(i)}{\omega_0}$$

$$\mu_1(t) = \sum_{i=t}^{L-1} i\,\frac{P(i)}{\omega_1} \qquad (6)$$

$$\mu_T = \sum_{i=t}^{L-1} i\;P(i)$$

The following relations in Eqn. 7 can be easily verified.

$$\omega_0\mu_0 + \omega_1\mu_1 = \mu_T$$
$$\omega_{0+}\,\omega_1 = 1 \tag{7}$$

The class probabilities and means can be computed iteratively, and this can provide a practical algorithm. Before finding contours, the threshold has been applied to the binary image to achieve higher accuracy.[21]

8.7.3. Implemented Method

To handle this problem of hand gesture recognition, we have developed code by using Python programming language along with the OpenCV library.

The fundamental concept behind this algorithm is to process each frame of a **live camera** stream. We created the method **process image(),** to which each frame would be sent for **analysis**.

The algorithms, however, will be executed on a predetermined ROI (region of interest), as we do not want to evaluate the entire frame picture with a lot of duplicate information.

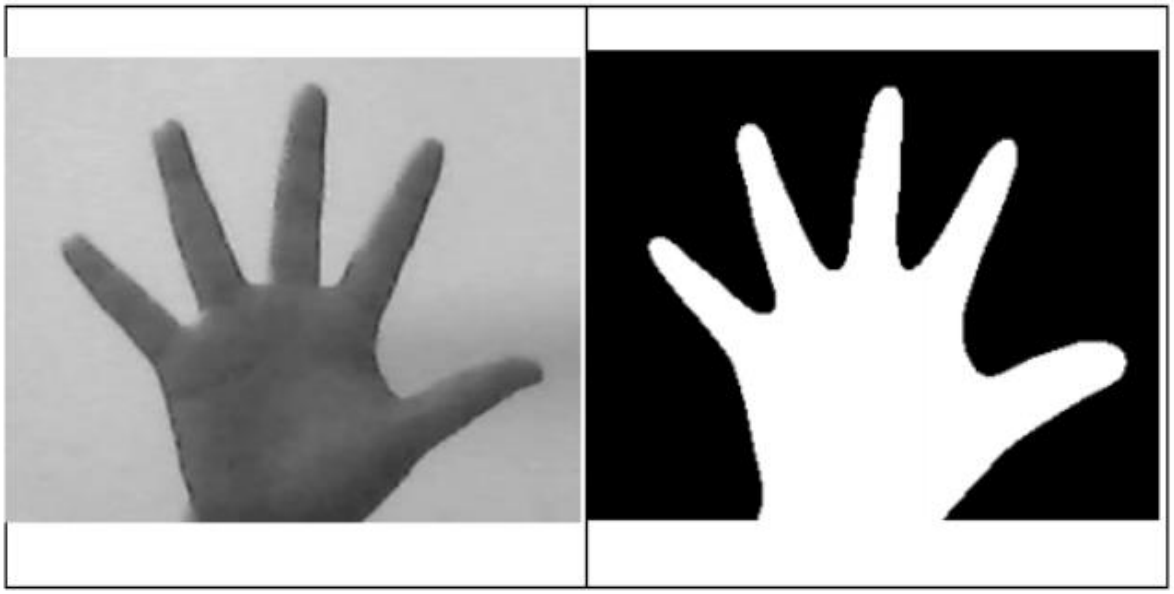


Figure 9: Comparison between gray scale image and thresholded image

8.7.4. Region of Interest (ROI)

The Territory of Interest, or ROI, is a region that will be treated as the region of sought area while disregarding the surrounding region, known as the backdrop. The detected region must be positioned in the same location as the region of interest to detect the availability of hand recognition. To put it another way, when the classifier follows the desired object, in this example, the bounding rectangle around the hand contour, the area of hand must coincide with the region of interest. The system may determine if the region selected, in this example, the rectangle surrounding the hand contour, overlaps with another active region, in this case, the hand itself, by applying the conditional approach of the overlapping region.[21]

By applying this theoretical aspect of ROI to simulation and hardware implementation, the process of detecting the appearance of a hand will be significantly simplified in comparison to the other

way described in [21]. Figure 2 depicts the ROI overlap criterion used to identify overlapping ROIs. These conditions are required to ensure that the software can recognize hand gestures.

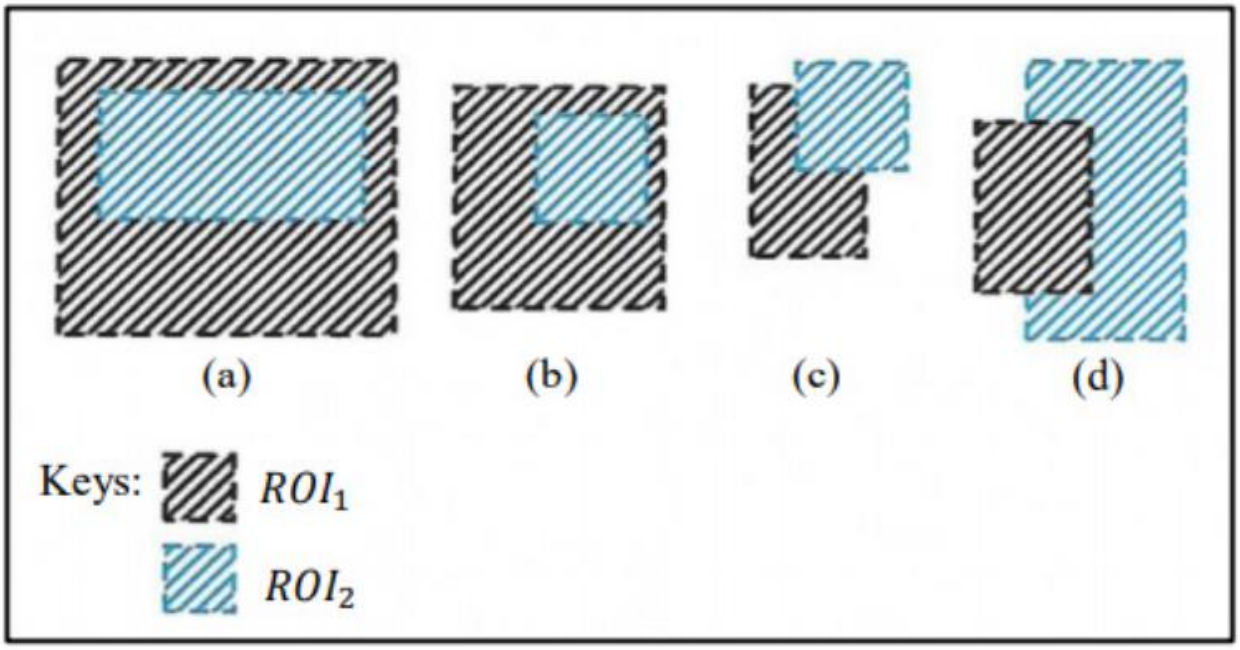


Figure 10: Overlapping Region of Interest (ROI)

https://www.researchgate.net/publication/349497837_Hand_gesture_recognition_on_python_and_opencv

The Region of Interest (ROI) is utilized in this project to view the hand motion since the ROI implementation is suited for detecting the overlapping two regions. Furthermore, the system will identify just two outputs for each hand gesture recognition. As seen in fig 12, this is possible:

Region of Interest (ROI)	Region implemented for
ROI_1	An active region, in this case the hand itself.
ROI_2	The rectangle sub window on screen. This region is static or not movable outside the frame of the realtime video and it highlights the area that involves.

Figure 11 : Application of Region of Interest (ROI).

The space consumption between the area between the convex hull and the contour of the hand was used to determine the detection of hand gestures. The convex hull is used to detect the edge of a user's hand by constructing a description of the edges in the most miniature convex set. The OpenCV library files, appropriate for any image processing approach, are utilized in this project.[21]

Based on the flowchart in Figure 13, it is possible to generate a Python file for both simulation and hardware prototyping. The principle of Region of Interest is used as the primary program action to identify the appearance and motion of the hand (ROI). For programs to monitor the rectangle surrounding the hand contour, the program must determine if the designated region of interest, in this example, the rectangle, will overlap with hand detection.[21]

Uncertainty is viewed as revealing the sort of gestures such as numbers or sign and the area inside the zone of interest. If there is no overlap (outside of the region of interest), the program can be thought of as **"show your hand in a box."**[21]

A collection of coordinates will be recognized to display the area of bounding rectangle around the hand contour to plan the region of interest for this project. There is a boundary rectangle area for this project, and therefore there is a zone of interest to be implemented with each gesture.[21]

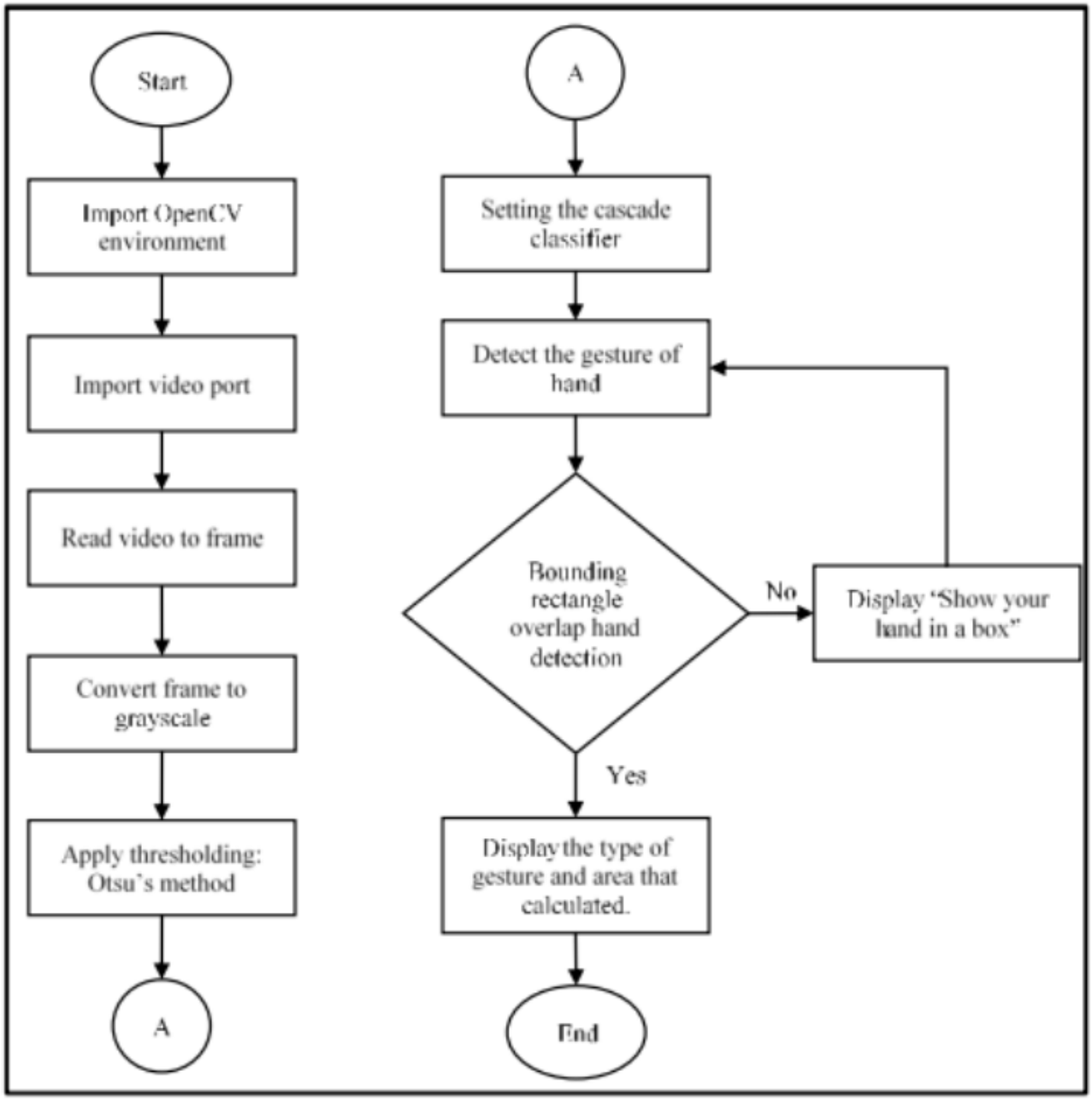


Figure 12 : Flowchart of Python file for hand gesture recognition

https://www.researchgate.net/publication/349497837_Hand_gesture_recognition_on_python_and_opencv

After generating the Python main file, the simulation begins, where the detection of hands is evaluated, identifying the type of motions displayed based on the specified region of interest. The OpenCV environment must be linked and executed through the command window before the Python program can be run.

The same processes were then followed for the hardware prototype. USB video web was used for this prototype. To convert the camera from the laptop web camera to the USB Video web camera, the source code in the Python core must be adjusted in the video capture section. During the simulation, the techniques for recognizing and analyzing hand gestures utilizing the USB video web camera stay the same.

8.8. Conclusion of the theoretical part

As a consequence of the project's findings, it is clear that building hand gesture recognition using Python and OpenCV can be accomplished by employing the theories of hand segmentation and the hand detection system.

To summarize, this system has accomplished several goals in this project:

(1) establish a complete system for detecting, recognizing, and interpreting hand gesture recognition through computer vision using Python and OpenCV, and (2) create the numbers and sign languages of hand gestures shown in the system that will meet the project's name.

This approach will be recommended in the future to execute more gestures that will allow each user with varying skin color and palm-size to accomplish more functions simply. The present system performs movements with both hands and a specified location in ROI. As a result, the planned upgrade of the approach may involve employing both of the user's hands to conduct distinct signals with computer activities. Furthermore, background removal methods can be utilized to improve speed. In the future, the Graphical User Interface (GUI) will be implemented so that people understand how to transform a gesture from its meaning to a sign or number and vice versa.

9. CHAPTER V – PRACTICAL PART

9.1. Introduction

It is a mouse simulation system executes all the mouse's operations in response to the hand movements and gestures. Simply put, a camera records the video, and we may move the cursor and execute left-click, right-click, drag, select, and scroll up and down based on the hand motions. Only three fingers identified with various colors are used in the prescribed movements.

9.2. Technical Overview

It is simply a software that does image processing, gets relevant data, and applies it to the computer's mouse interface based on specified conceptions.

Python3.7 is used to write the code, and it uses the cross-platform image processing package OpenCV and implements mouse operations with the Python-specific tool PyAutoGUI. The webcam video recordings are analyzed, and the three colored fingers are removed. Their centers are estimated using the method of moments, and their respective locations determine the action to be taken.

9.3. Step 1: Install Anaconda & Packages

Anaconda software is a free open-source data science tool that focuses on the distribution of the R and Python programming languages for use in data science and machine learning projects. Anaconda's goal is to make data management and deployment easier.

For data scientists, Anaconda is a strong data science platform. Anaconda's package manager is conda, which keeps track of package versions.[23]

9.3.1. Visit Anaconda.com/downloads

We Go to the following link: Anaconda.com/downloads.[22]

The Anaconda Downloads Page will look something like this:

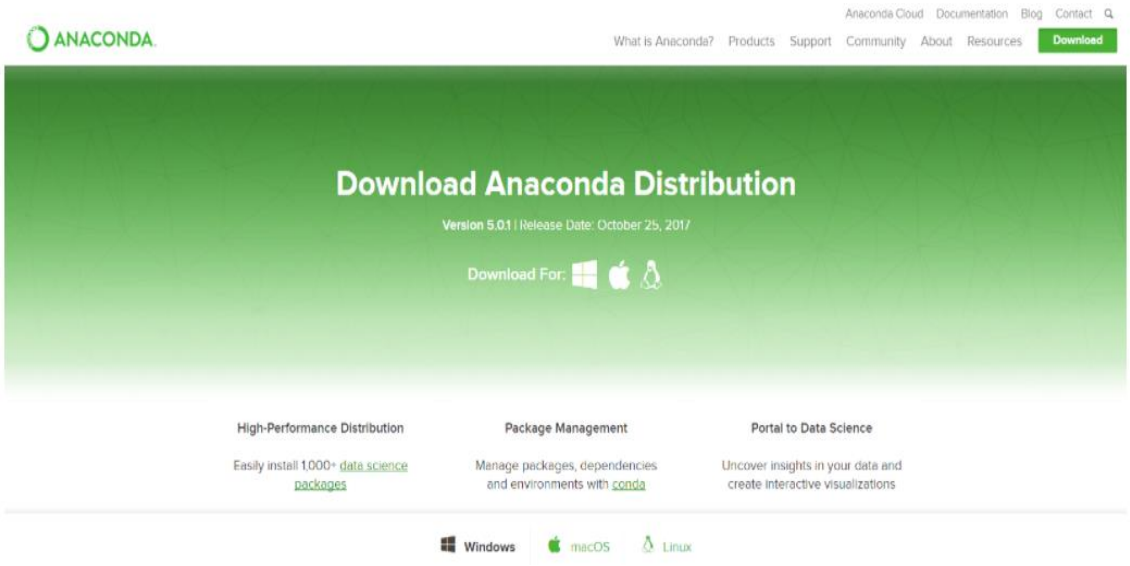


Figure 13: Anaconda download platform.

https://problemsolvingwithpython.com/01-Orientation/01.03-Installing-Anaconda-on-Windows/

9.3.2. Selecting Windows

We select **Windows** where the three operating systems are listed.

Figure 14 : Different OS that support anaconda installation.

https://problemsolvingwithpython.com/01-Orientation/01.03-Installing-Anaconda-on-Windows/

9.3.3. Download

We download the last version of Python 3 release. Python 3.6 was the most current version released. Python 2.7 is considered legacy Python. Choose Python 3.6 as the version. If there is an unclarity that the machine is running a 64-bit or 32-bit version of Windows, choose 64-bit because 64-bit Windows is the most prevalent. [22]

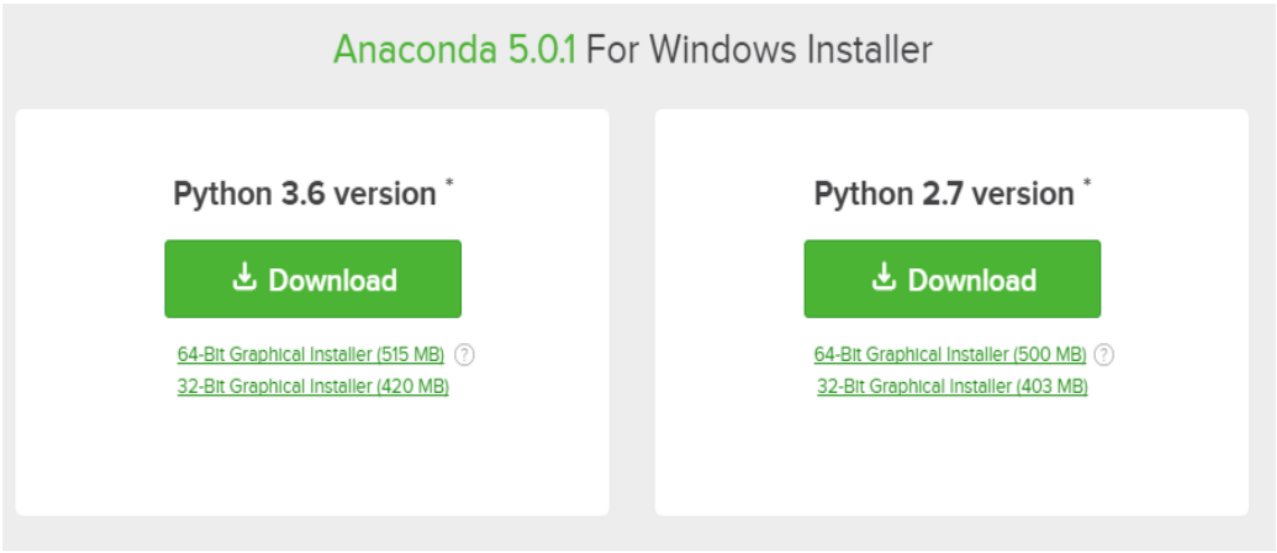


Figure 15: Different Versions of Python.

https://problemsolvingwithpython.com/01-Orientation/01.03-Installing-Anaconda-on-Windows/

At the beginning of the install, we need to click **Next** to confirm the installation.

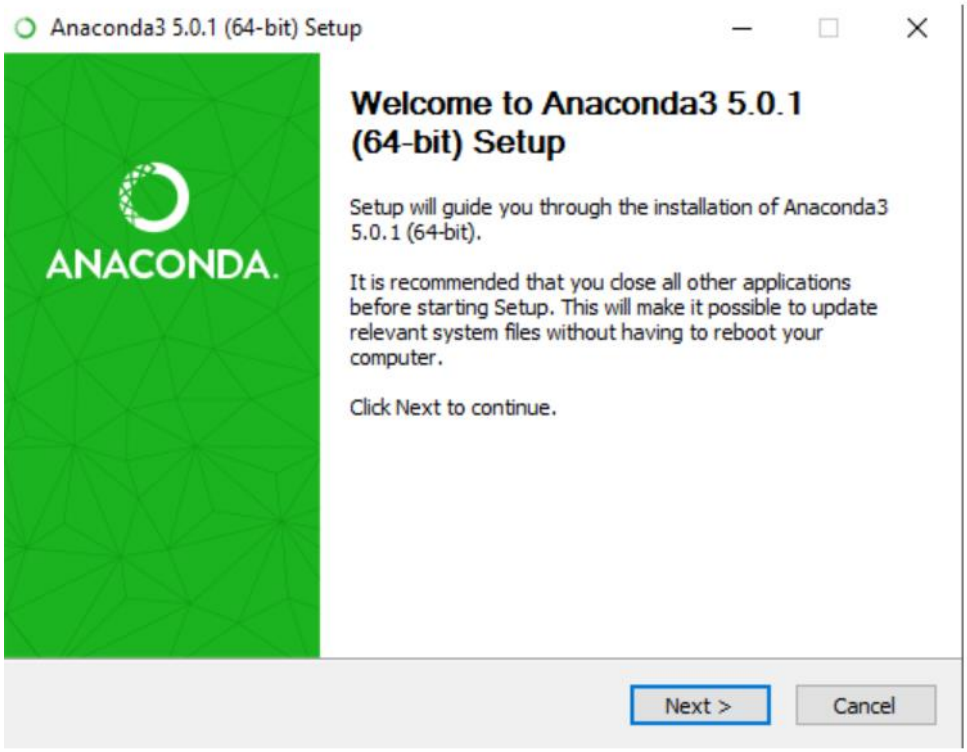


Figure 16: 1st window to install Anaconda.

https://problemsolvingwithpython.com/01-Orientation/01.03-Installing-Anaconda-on-Windows/

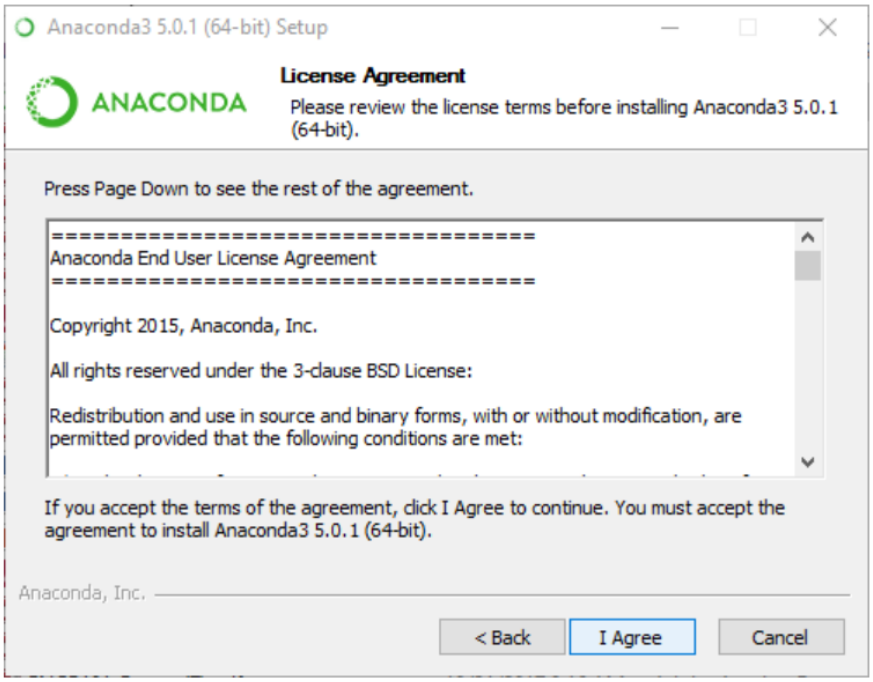


Figure 17: accepting the license agreement.

https://problemsolvingwithpython.com/01-Orientation/01.03-Installing-Anaconda-on-Windows/

At the Advanced Installation, we recommend **not checking** "Add Anaconda to my PATH environment variable."

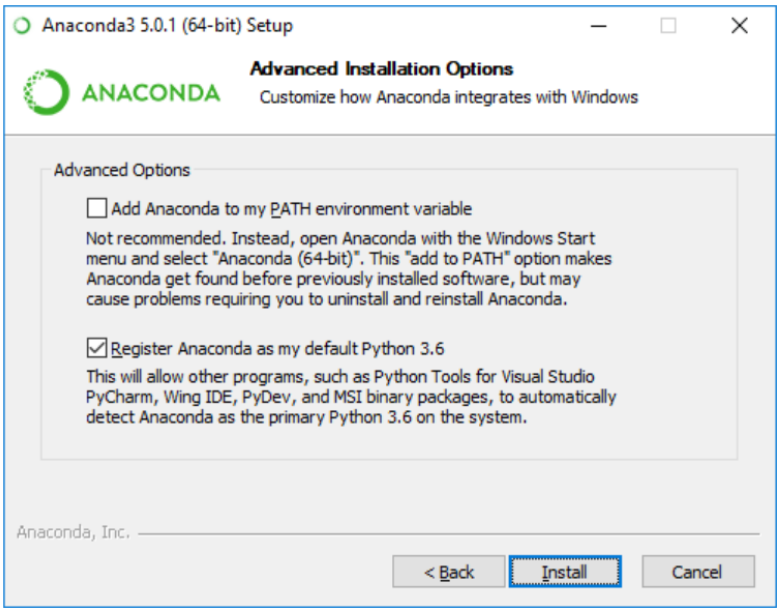
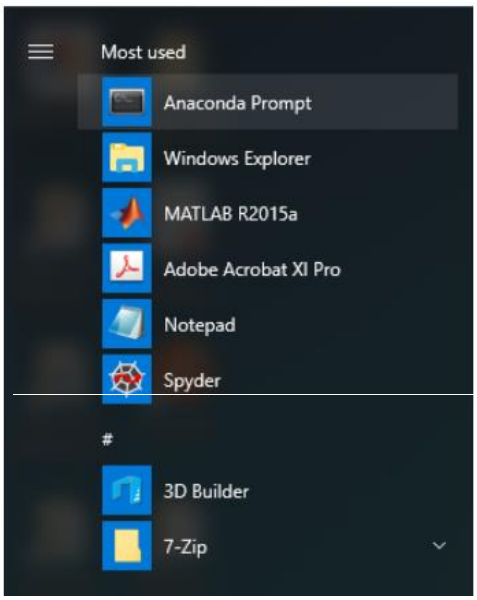


Figure 18: Advanced installation options.

https://problemsolvingwithpython.com/01-Orientation/01.03-Installing-Anaconda-on-Windows/

9.3.4. Open the Anaconda Prompt



After the installation of Anaconda is complete, we can go to the Windows start menu and select the Anaconda Prompt.

This opens the "**Anaconda Prompt.**" **Anaconda** is the Python distribution, and **Anaconda Prompt** is a command-line shell (a program where we can type in commands instead of using a mouse). The black screen and text that make up the **Anaconda Prompt** doesn't look like much, but it is beneficial for problem solvers using Python.

Figure 20: check Python version.

9.4. Step 2 : Install OpenCV

we open Anaconda **Prompt,** then we type commands to install necessary libraries:

conda install -c conda-forge opencv

or if we want to install a specific version of python, we use the following command:

pip install opencv-contrib-python==3.9

so here, we can alternatively choose to install through the anaconda navigator graphical interface. We Type in "OpenCV" in the search packages search bar. we Choose to install all the listed packages:

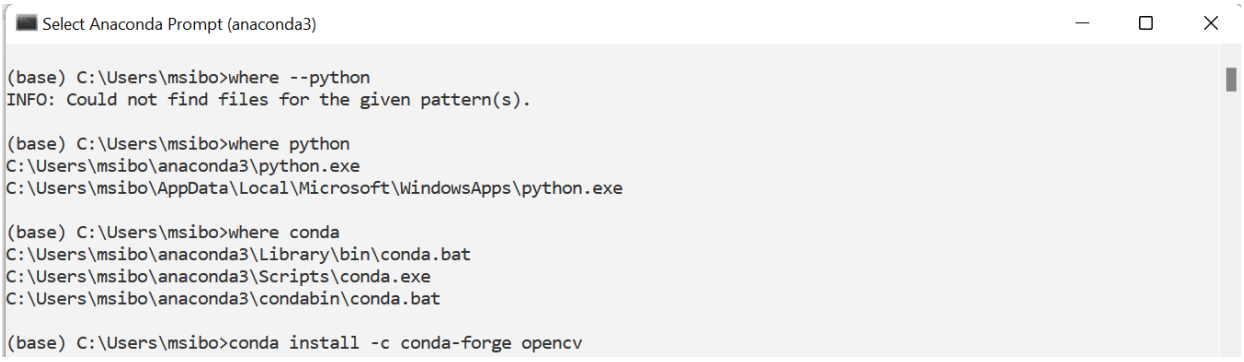


Figure 21: installing OpenCV through anaconda prompt.

The prompt show that it is "solving environment". It takes some time to finish the installation process:

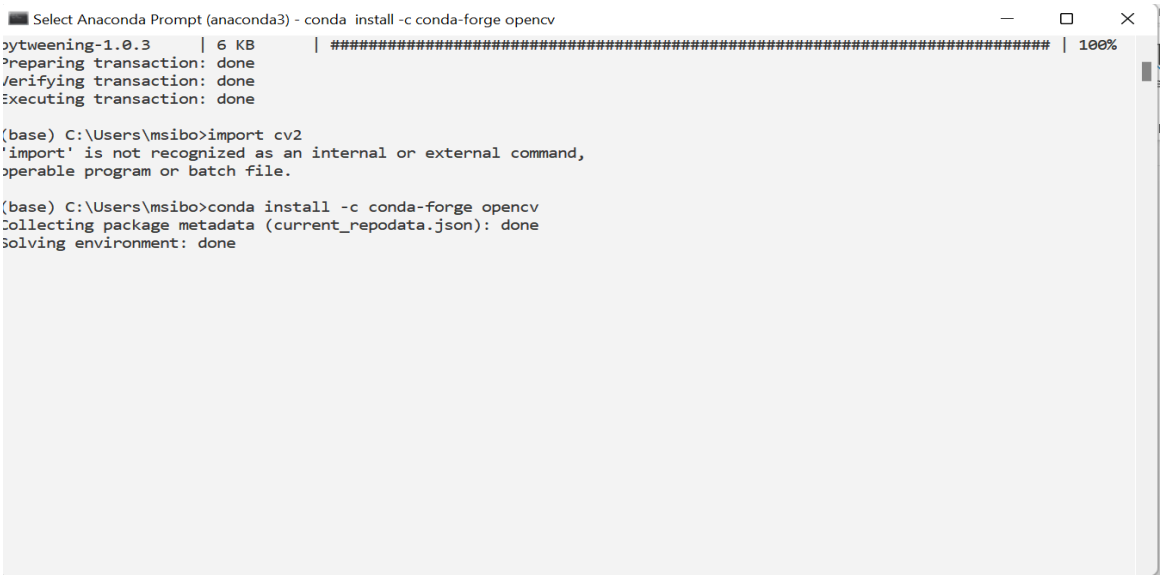


Figure 22: solving environment

Figure 23: the packages that will be installed .

Once the environment is done by conda it will list the packages that will be installed, namely: ***opencv, libopencv, py-opencv.*** etc

we enter **y** to proceed with the installation.

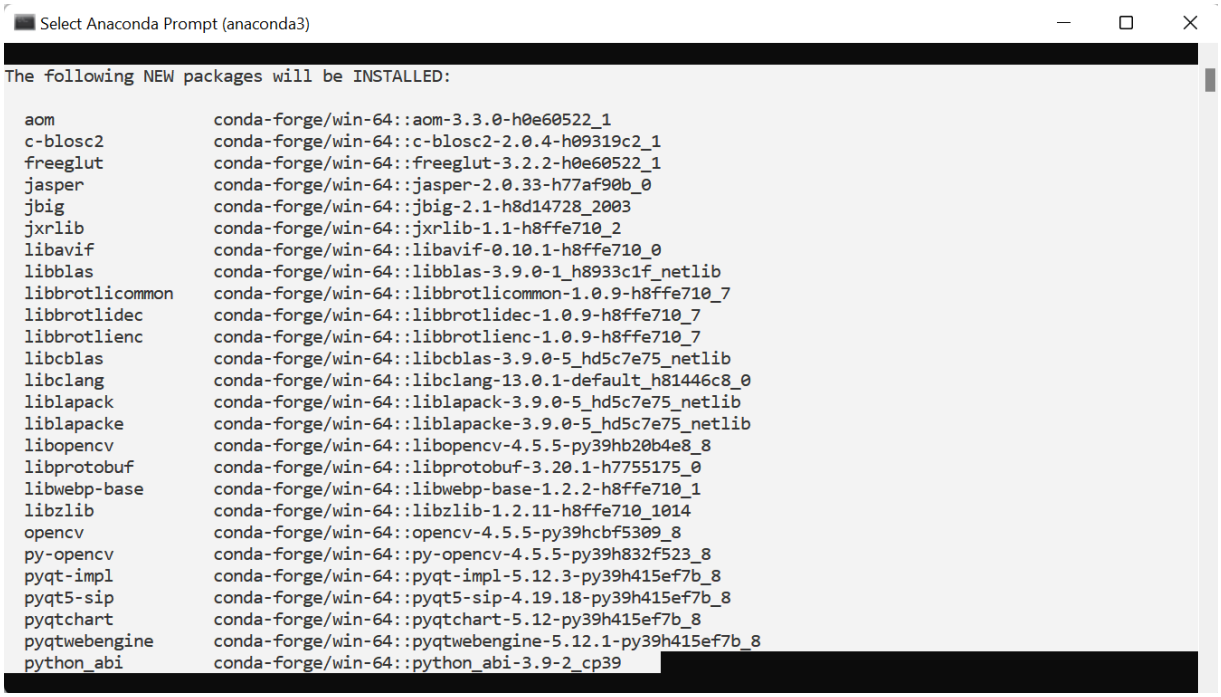


Figure 24: conda downloads and installs all required packages.

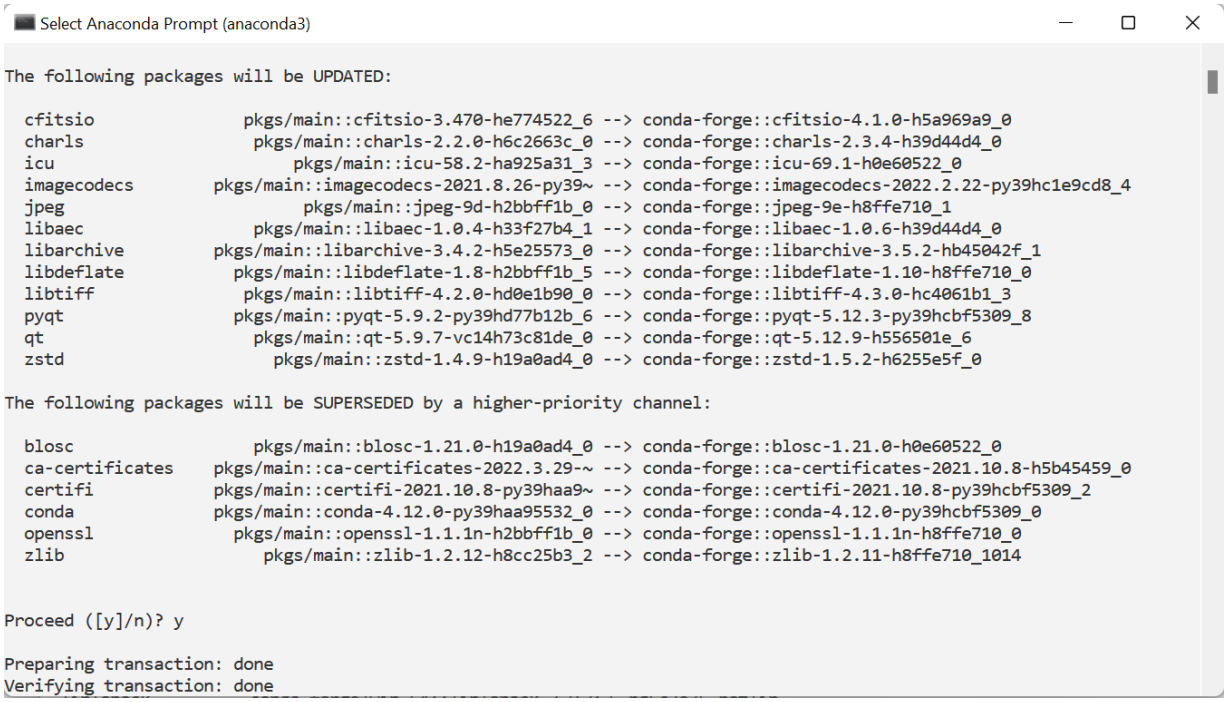


Figure 25:installation is successful.

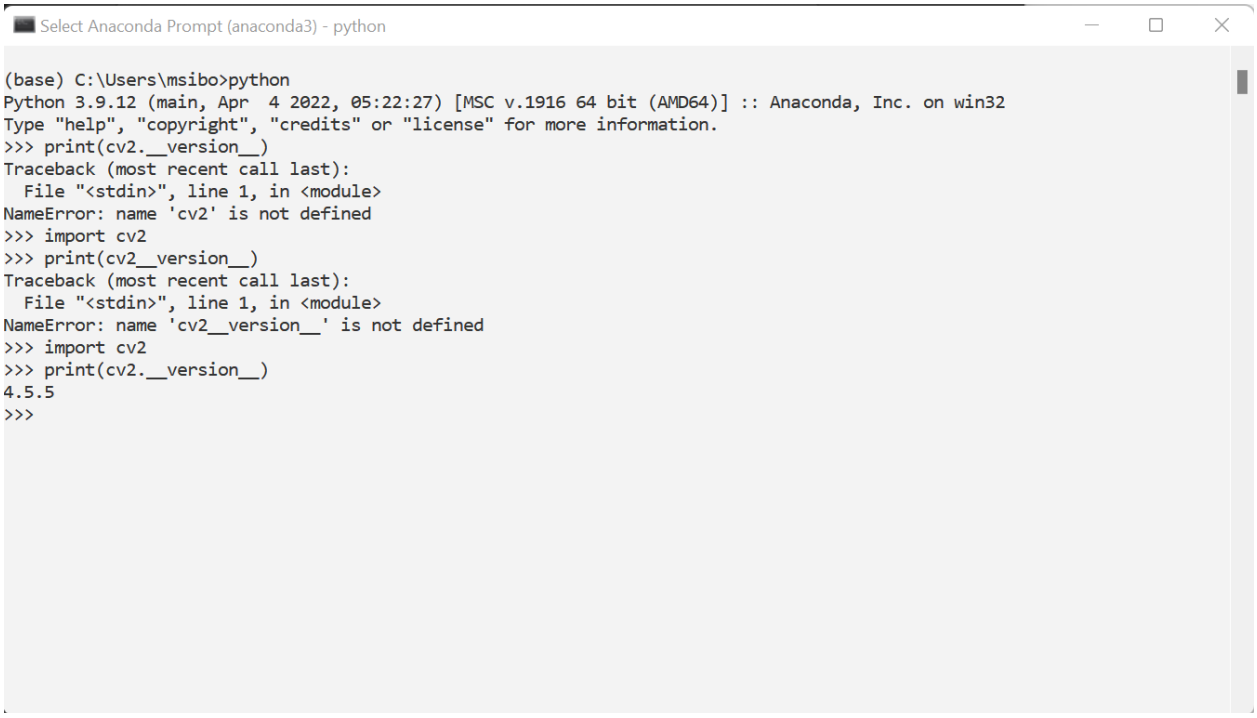


Figure 26:verify the version of opencv .

9.4.1. Install pyautogui

PyAutoGUI allows Python programs to control the mouse and keyboard in order to automate interactions with other apps. The API is intended to be straightforward. **PyAutoGUI** operates on Python 2 and 3 and is compatible with Windows, macOS, and Linux.

PyAutoGUI includes the following features:

- Moving the mouse and clicking on other apps' windows
- Keystrokes are sent to apps (for example, to fill out forms).
- Take screenshots and given an image (such as a button or checkbox), locate it on the screen.
- Locate the window of an application and move, resize, maximize, minimize, or dismiss it (Windows-only, currently).
- Show the alert and message boxes.

To install this tool we use the following conda commande :

conda install -c conda-forge pyautogui

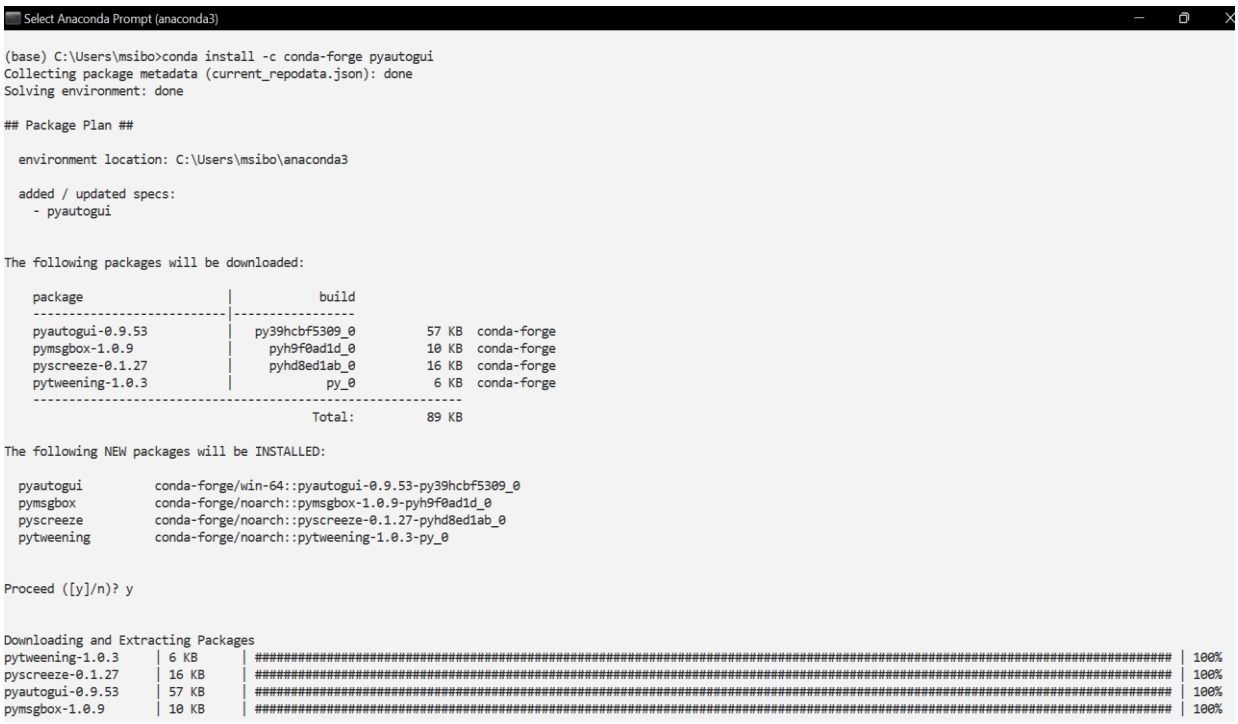


Figure 27: install Pyautogui tool

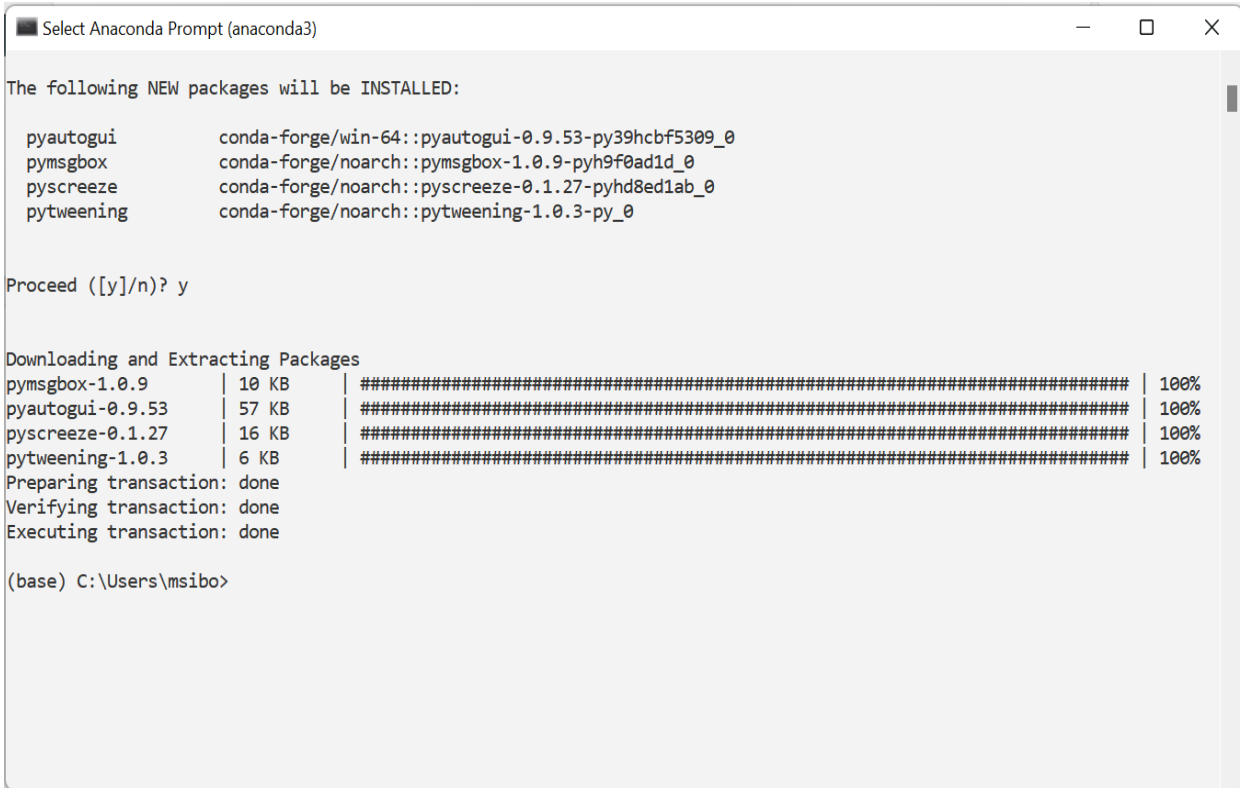


Figure 28: the pyautogui installation has been done

9.5. Step 3 : Capture Video from Camera

We need to capture the live feed using a camera. OpenCV provides a pretty straightforward interface for this. Let's take a video from the camera (we can use the laptop's built-in webcam or an external USB camera), convert it to grayscale video, and show it. To get started, we need a basic assignment. To record a video, you must first build a VideoCapture object. It can accept either the device index or the name of a video file as an input. The device index is a number that indicates which camera is being used. Normally, only one camera is attached (as in my case). As a result, I

merely pass (0 or -1). You may choose the second camera instead of the first, and so on. After that, you may grab the video frames. Finally, don't forget to release the capture.

convert the captured video into HSV format.

```
# All packages needed for the program are imported ahead

import cv2
cap = cv2.VideoCapture(0)
while(1):

     # Capture frame-by-frame
    _, frameinv = cap.read()

    # flip horizontaly to get mirror image in camera
    frame = cv2.flip( frameinv, 1)

    # Our operations on the frame come here
    hsv = cv2.cvtColor( frame, cv2.COLOR_BGR2HSV)

    # Display the resulting frame
    cv2.imshow('Frame', hsv)

    k = cv2.waitKey(10) & 0xFF
    if k == 27:
        break
cap.release()
cv2.destroyAllWindows()
```

Figure 29: converting the captured video into HSV format.

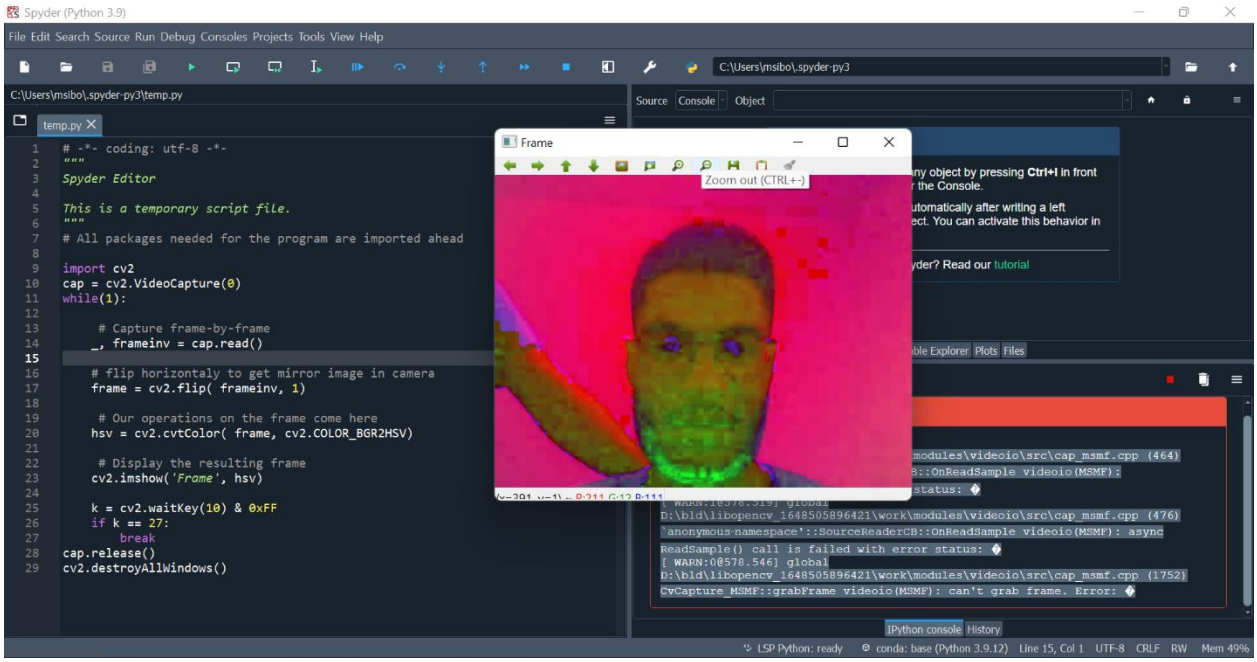
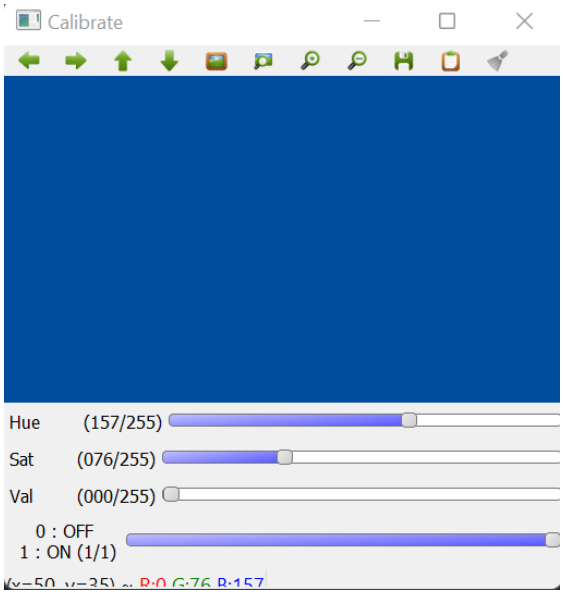


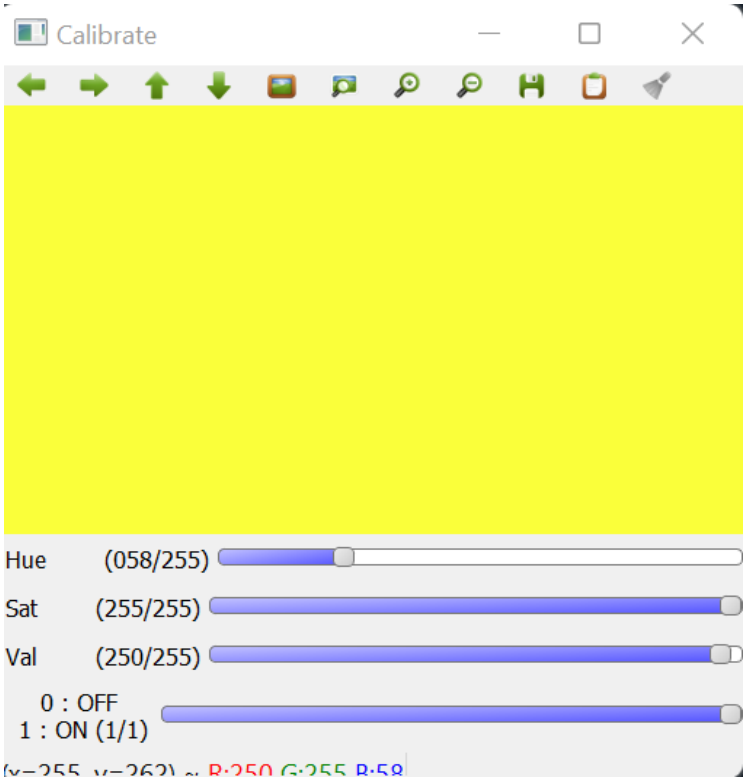
Figure 30: capture video from camera

9.6. Step 4: Calibrate Color

The user may now independently calibrate the color ranges for three fingers. This may be accomplished by calling the Calibratecolor() function directly at the start of the application. The user can also choose to utilize the default settings.



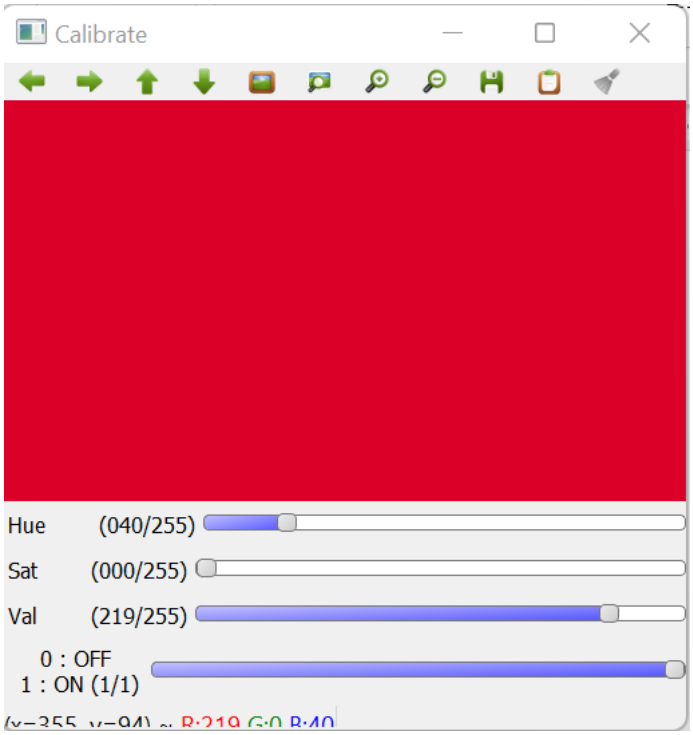
The aim of blue color calibration is to allow the system to easily detect the blue color in the finger and to assure the perfect function of the mouse controlling.



The aim of yellow color calibration is to allow the system to easily detect the yellow color in the finger and to assure the free movement functionality of the cursor.

Figure 33: yellow color calibration

The aim of red color calibration is to allow the system to easily detect the red color in the finger and to assure the right and the left click functionality of the cursor.



```
import cv2
import numpy as np

def nothing(x):
    pass

# Create a black image, a window
kernel = np.zeros((300,512,3), np.uint8)
name = 'Calibrate'
cv2.namedWindow(name)

# create trackbars for color change
cv2.createTrackbar('Hue', name, 0, 255, nothing)
cv2.createTrackbar('Sat', name, 0, 255, nothing)
cv2.createTrackbar('Val', name, 0, 255, nothing)

# create switch for ON/OFF functionality
switch = '0 : OFF \n 1 : ON'

cv2.createTrackbar(switch, name,0,1,nothing)

while(1):
    cv2.imshow(name,kernel)
    k = cv2.waitKey(1) & 0xFF
    if k == 27:
        break

    # get current positions of four trackbars
    hue = cv2.getTrackbarPos('Hue', name)
    sat = cv2.getTrackbarPos('Sat', name)
    val = cv2.getTrackbarPos('Val', name)
    s = cv2.getTrackbarPos(switch,name)

    if s == 0:
        kernel[:] = 0
    else:
        kernel[:] = [hue,sat,val]

cv2.destroyAllWindows()
```

9.7. Step 5: Remove Noise and Define Functions in the Video Feed

Using the cv2.inRange() method, just the three fingers are retrieved from the video, one by one, depending on the calibrations. We use a two-step morphism, erosion and dilation, to eliminate noise from the video input. The program then sends the noise-filtered image, referred to as the mask, to locate the centers.

```
# cv2.inRange function is used to filter out a particular color from the frame
# The result then undergoes morphosis i.e. erosion and dilation
# Resultant frame is returned as mask

def makeMask(hsv_frame, color_Range):

        mask = cv2.inRange( hsv_frame, color_Range[0], color_Range[1])
        # Morphosis next ...
        eroded = cv2.erode( mask, kernel, iterations=1)
        dilated = cv2.dilate( eroded, kernel, iterations=1)

        return dilated
```

Figure 35: Removing Noise & Defining Functions in the Video Feed

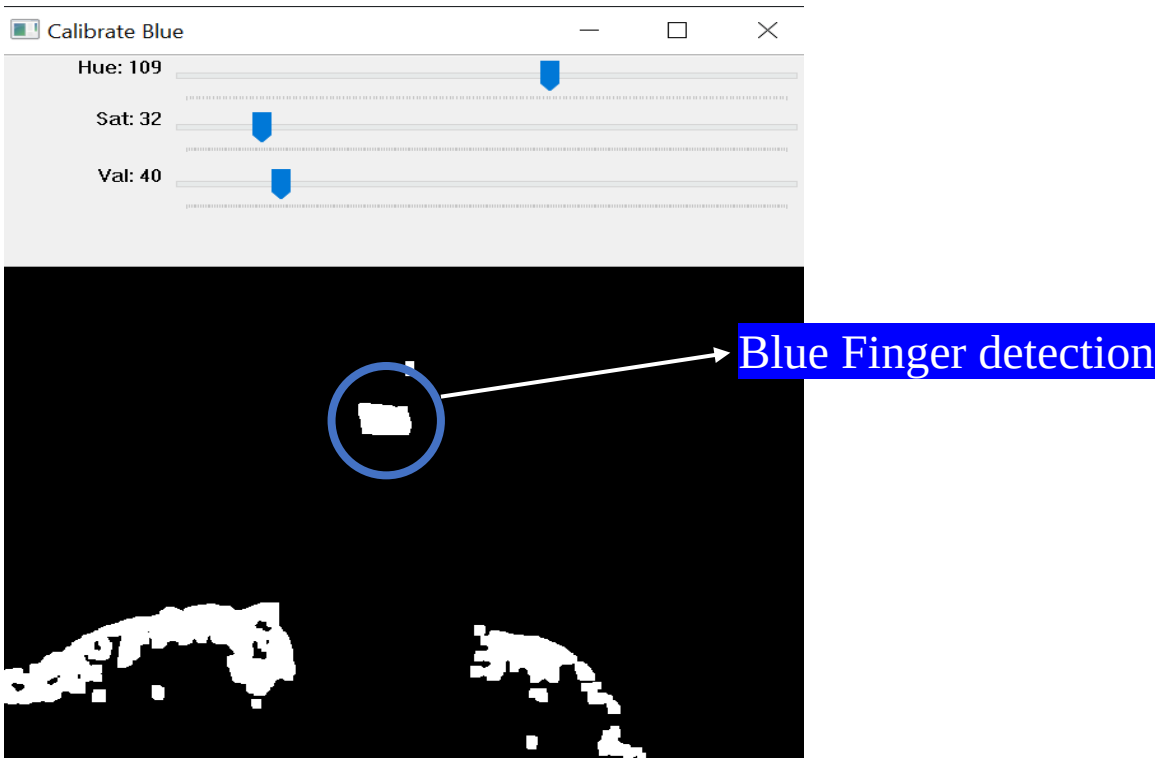


Figure 36: Blue color calibration in removing noise mode

Figure 37: red color calibration in removing noise mod

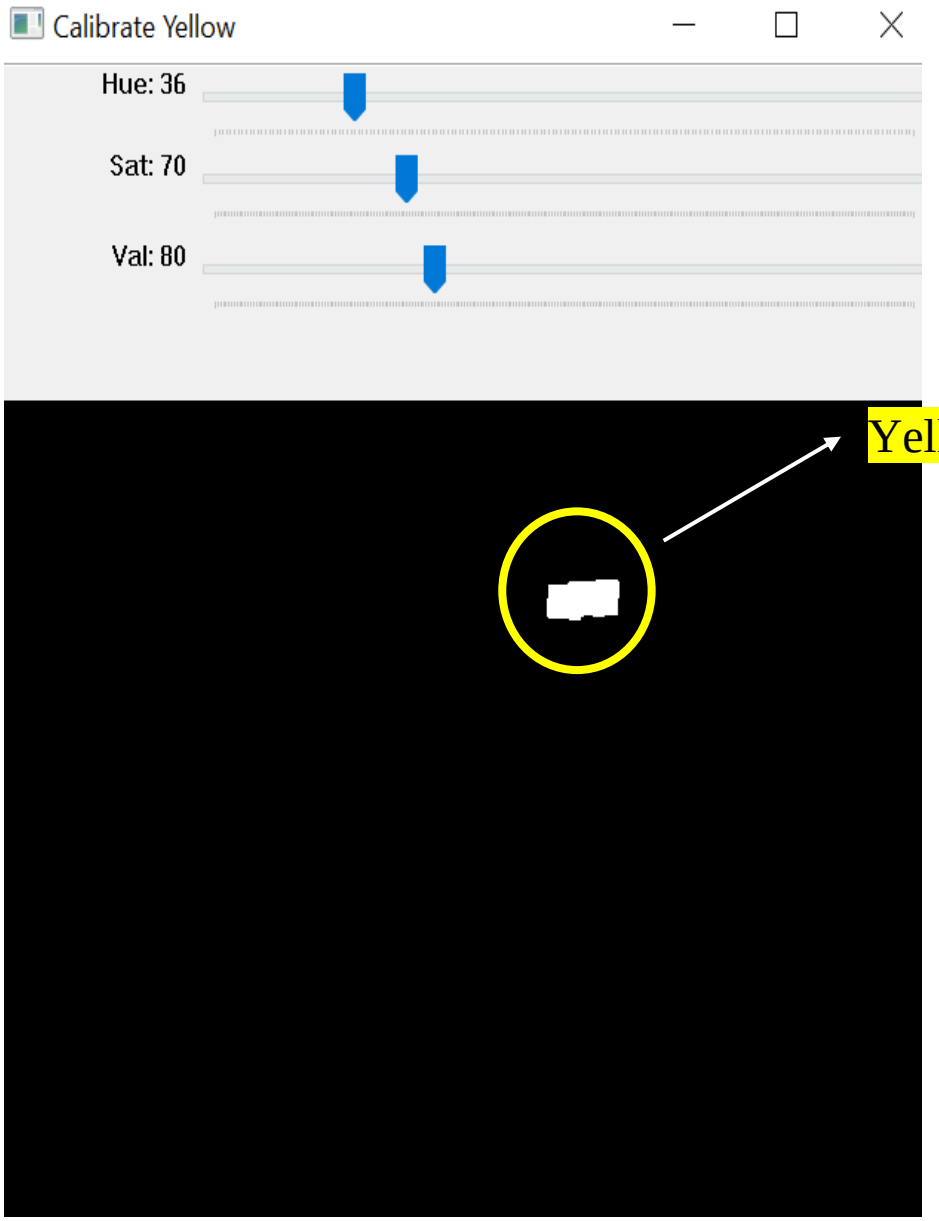


Figure 38: Yellow color calibration in removing noise mode

After we have done the three-color calibration, our system can be launched as the following:

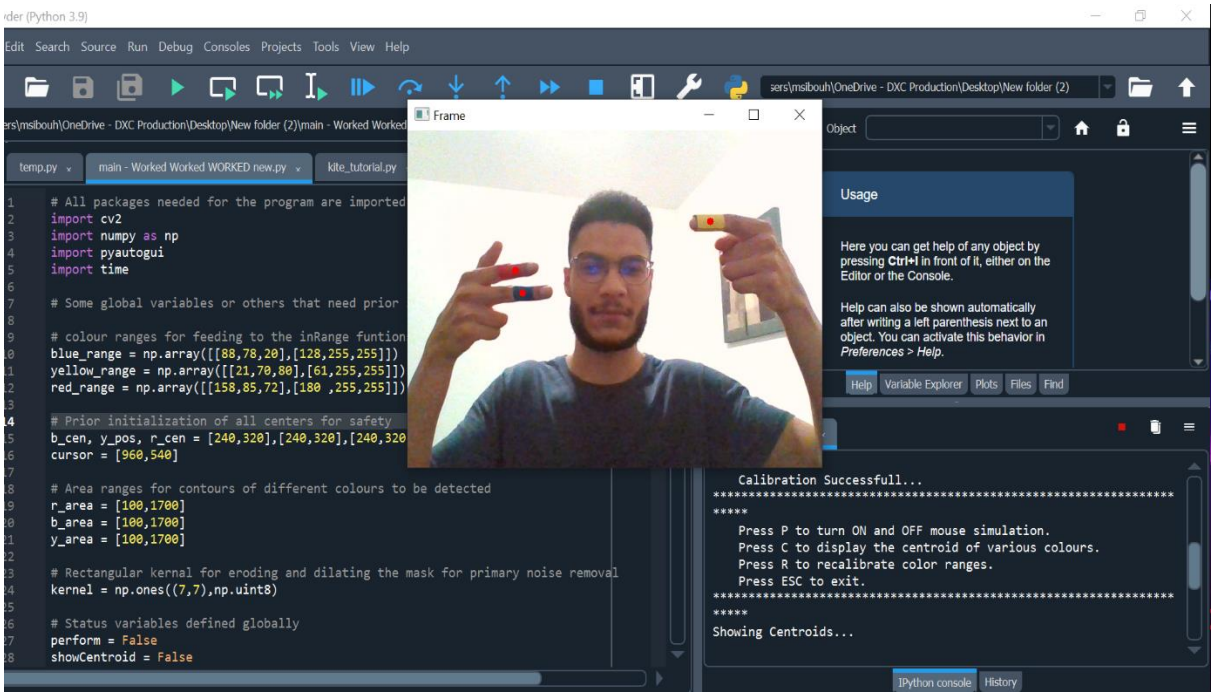
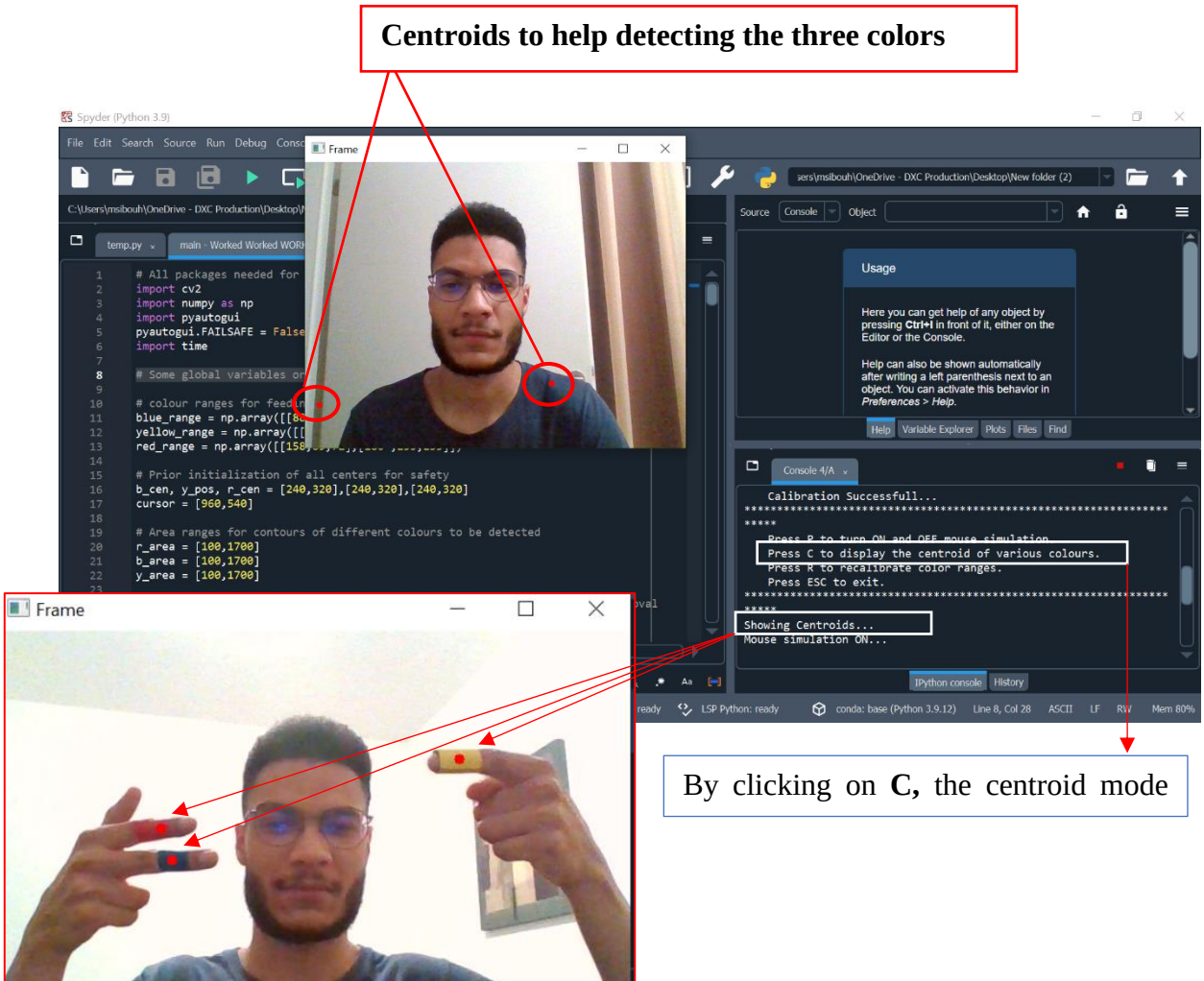


Figure 39: calibration done, and the system detects the 3 finger colors

9.8. Step 6: Find Contours & Draw Centroids

Each of the three centers' locations entails:

- **Identifying** contours in the mask that are relevant to that color range.
- **Using area** filters to remove contours from unnecessary regions.
- **Find the most** significant contour among the remaining ones and use the moments approach to determine its center.



```
# Contours on the mask are detected.. Only those lying in the previously set area
# range are filtered out and the centroid of the largest of these is drawn and returned
def drawCentroid(vid, color_area, mask, showCentroid):

        contour, _ = cv2.findContours( mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
        l=len(contour)
        area = np.zeros(l)
        # filtering contours on the basis of area rane specified globally
        for i in range(l):
                if cv2.contourArea(contour[i])>color_area[0] and cv2.contourArea(contour[i])
<color_area[1]: area[i]="cv2.contourArea(contour[i])" else:="" a="sorted(" area,="" reverse="True)" <="" p="
        # bringing contours with largest valid area to the top
        for i in range(l):
                for j in range(1):
                        if area[i] == a[j]:
                                swap( contour, i, j)

        if l > 0 :
                # finding centroid using method of 'moments'
                M = cv2.moments(contour[0])
                if M['m00'] != 0:
                        cx = int(M['m10']/M['m00'])
                        cy = int(M['m01']/M['m00'])
                        center = (cx,cy)
                        if showCentroid:
                                cv2.circle( vid, center, 5, (0,0,255), -1)

                        return center
        else:
                # return error handling values
                return (-1,-1)
```

9.9. Step 7: Final Step (Set Position, Choose & Perform Actions)

The next step is to define the cursor's position on the screen. The thumb controls the cursor's location, which is highlighted in yellow. so to achieve this goal, the following strategies were employed:

- The cameras we use typically record video at a resolution of 640x480 pixels. Assume that this frame was linearly transferred to a 1920x1080 pixel display panel. If we have a right-handed user, he will find it more challenging to access the left edge of the screen than the right edge and accessing the bottom section of the screen would also cause wrist strain.
- We realized that rather than mapping the entire video frame to the screen, we could explore a rectangular sub-portion more skewed towards the right (for right-handed users) and higher regions of the frame to increase comfort. This 480x270 pixel sub-portion is then linearly translated to the screen with a scaling factor of 4.

```
cursor[0] = 4*(yp[0]-110)
cursor[1] = 4*(yp[1]-120)
```

The centers continue to vibrate around a mean location due to sounds acquired by the camera and motions in hand. These vibrations cause many issues with the precision of the cursor location when scaled up. To alleviate cursor shakiness, we employ differential position allocation for the cursor. We compare the new center to the cursor's last location. If the change is smaller than 5 pixels, the cause is likely noise. As a result, the new cursor position is more similar to the prior one. A wider difference between the old position and the new center, on the other hand, is deemed deliberate movement, and the new cursor position is placed close to the new center. Examine the setCursorPosition() method in the code for further information.

```
'''
This function takes as input the center of yellow region (yc) and
the previous cursor position (pyp). The new cursor position is calculated
in such a way that the mean deviation for desired steady state is reduced.
'''
def setCursorPos( yc, pyp):

        yp = np.zeros(2)

        if abs(yc[0]-pyp[0])<5 and abs(yc[1]-pyp[1])<5:
                yp[0] = yc[0] + .7*(pyp[0]-yc[0])
                yp[1] = yc[1] + .7*(pyp[1]-yc[1])
        else:
                yp[0] = yc[0] + .1*(pyp[0]-yc[0])
                yp[1] = yc[1] + .1*(pyp[1]-yc[1])

        return yp
```

Figure 41 : input of the center of yellow region and the previous cursor position

The three centers are now sent to determine what action should be taken based on their respective locations. This is done in the code with the **chooseAction()** function. Depending on its result, the **performAction()** function uses the **PyAutoGUI library** to do one of the following:

PyAutoGUI library:

1. free cursor movement
2. left-click
3. right-click
4. drag/select
5. scroll up
6. scroll down

```
def chooseAction(yp, rc, bc):
        out = np.array(['move', 'false'])
        if rc[0]!=-1 and bc[0]!=-1:

                if distance(yp,rc)<50 and distance(yp,bc)<50 and distance(rc,bc)<50 :
                        out[0] = 'drag'
                        out[1] = 'true'
                        return out
                elif distance(rc,bc)<40:
                        out[0] = 'right'
                        return out
                elif distance(yp,rc)<40:
                        out[0] = 'left'
                        return out
                elif distance(yp,rc)>40 and rc[1]-bc[1]>120:
                        out[0] = 'down'
                        return out
                elif bc[1]-rc[1]>110:
                        out[0] = 'up'
                        return out
                else:
                        return out
        else:
                out[0] = -1
                return out
```

```
def performAction( yp, rc, bc, action, drag, perform):
        if perform:
                cursor[0] = 4*(yp[0]-110)
                cursor[1] = 4*(yp[1]-120)
                if action == 'move':
                        if yp[0]>110 and yp[0]<590 and yp[1]>120 and yp[1]<390:
                                pyautogui.moveTo(cursor[0],cursor[1])
                        elif yp[0]<110 and yp[1]>120 and yp[1]<390:
                                pyautogui.moveTo( 8 , cursor[1])
                        elif yp[0]>590 and yp[1]>120 and yp[1]<390:
                                pyautogui.moveTo(1912, cursor[1])
                        elif yp[0]>110 and yp[0]<590 and yp[1]<120:
                                pyautogui.moveTo(cursor[0] , 8)
                        elif yp[0]>110 and yp[0]<590 and yp[1]>390:
                                pyautogui.moveTo(cursor[0] , 1072)
                        elif yp[0]<110 and yp[1]<120:
                                pyautogui.moveTo(8, 8)
                        elif yp[0]<110 and yp[1]>390:
                                pyautogui.moveTo(8, 1072)
                        elif yp[0]>590 and yp[1]>390:
                                pyautogui.moveTo(1912, 1072)
                        else:
                                pyautogui.moveTo(1912, 8)
                elif action == 'left':
                        pyautogui.click(button = 'left')
```

```
                elif action == 'right':
                        pyautogui.click(button = 'right')
                        time.sleep(0.3)
                elif action == 'up':
                        pyautogui.scroll(5)
#                       time.sleep(0.3)
                elif action == 'down':
                        pyautogui.scroll(-5)
#                       time.sleep(0.3)
                elif action == 'drag' and drag == 'true':
                        global y_pos
                        drag = 'false'
                        pyautogui.mouseDown()

                        while(1):
                                k = cv2.waitKey(10) & 0xFF
                                changeStatus(k)
                                _, frameinv = cap.read()
                                # flip horizontaly to get mirror image in camera
                                frame = cv2.flip( frameinv, 1)
                                hsv = cv2.cvtColor( frame, cv2.COLOR_BGR2HSV)
                                b_mask = makeMask( hsv, blue_range)
                                r_mask = makeMask( hsv, red_range)
                                y_mask = makeMask( hsv, yellow_range)
```

```
                                py_pos = y_pos
                                b_cen = drawCentroid( frame, b_area, b_mask, showCentroid)
                                r_cen = drawCentroid( frame, r_area, r_mask, showCentroid)
                                y_cen = drawCentroid( frame, y_area, y_mask, showCentroid)

                                if      py_pos[0]!=-1 and y_cen[0]!=-1:
                                        y_pos = setCursorPos(y_cen, py_pos)
                                performAction(y_pos, r_cen, b_cen, 'move', drag, perform)
                                cv2.imshow('Frame', frame)
                                if distance(y_pos,r_cen)>60 or distance(y_pos,b_cen)>60 or distance(r_cen,
b_cen)>60:
                                        break
                        pyautogui.mouseUp()
```

Figure 42: choose whether the user desires free movement of cursor left click, right click or dragging

9.10. Virtual cursor movements with fingers

The Following are the results of various hand gestures that result into mouse operations.

1. Free Cursor Movement

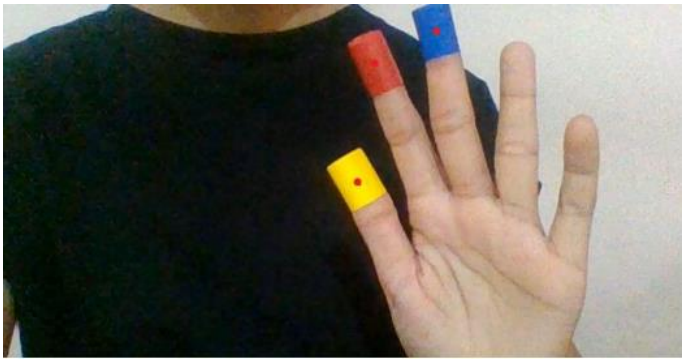


Figure 43: Free movement

2. Left Click and Right Click

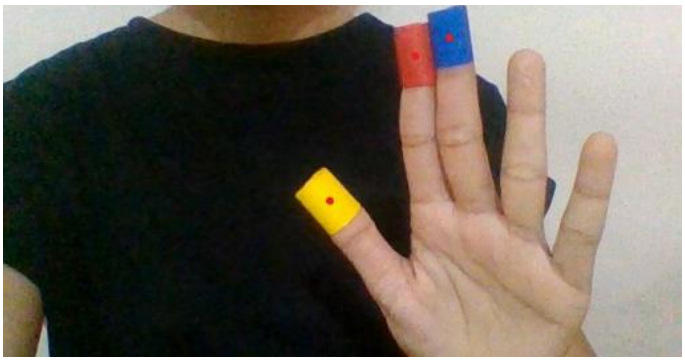


Figure 44: Right click

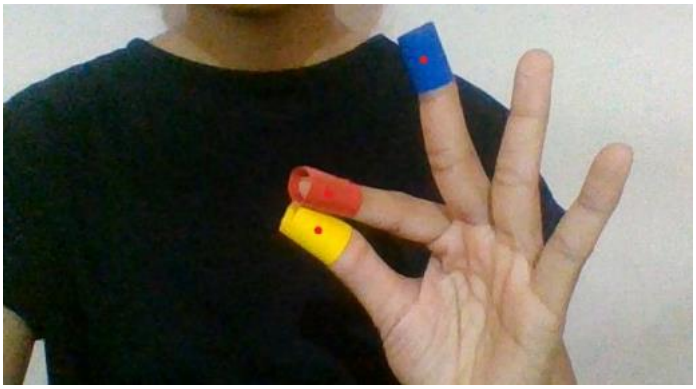


Figure 45: Left click

3. Drag and Select

Figure 46: Drag/Select

4. Scroll Up and Scroll Down

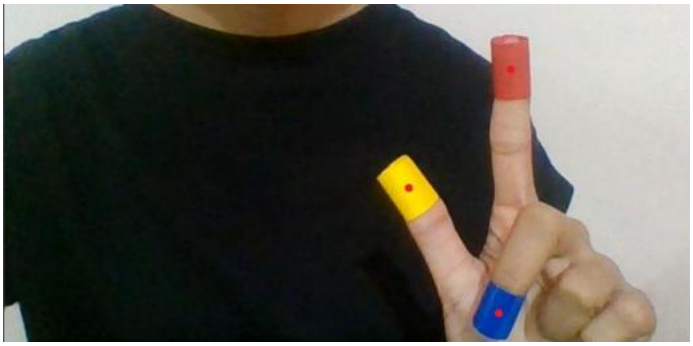


Figure 47: Scroll up

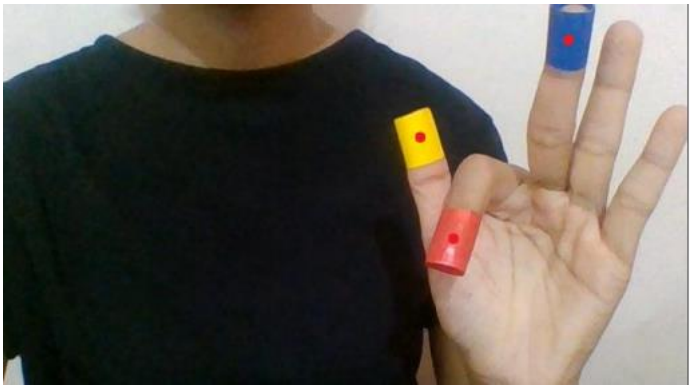


Figure 48: Scroll down

10. POSSIBLE IMPROVEMENTS

Gesture recognition is the process through which images and videos are used to collect gestures and perform corresponding actions based on controlling mouse pointer. Gesture recognition is the domain which is increasing its area day by day. It can be utilized in every field as computer system is part of every system because of its convergence with other technologies. Gesture recognition is the process through which user interface can be controlled with the help of defined gestures. As mouse is the pointing device to operate the graphical user interface but with the advancement we can operate the console or interface without using traditional devices. User login and attendance is controlled and operated by gestures performed by hand gestures. To use this system efficiently, hardware and software requirements should be satisfied. In this system only hand gestures are used. Left and right hand gestures are used to operate interface. Mouse is also performing single click with the help of gestures.

It can be used in other domains like information exploring on airports, railways, hospitals etc. Gesture set can be extended as having less number of gestures. Performance can be improved by using better resources as are quite expensive. Only hand gestures are used here, facial and body postures can also be used as future work. In this system future work can be based on either on domains or on algorithms.

Based on domain

- It can be utilized on Airports, Railways Reservation system to explore information or for the reservation purpose.
- It can be used in malls, shopping marts and plazas to try new cloths, accessories, footwear's without wearing them.
- Can be used in universities or campus to explore information, virtually viewing campus
- Finger, body and facial recognition techniques can be used.
- With the change in hardware speed & performance can be improved.
- Different platforms can be used as its implemented with C#, matlab or OpenCv can be used.
- Different algorithms can be used based on different techniques.

11. CONCLUSION

Keyboard and mouse actually form an integral part of the computer system. Our system architecture can facilitate the use of computer for the paralyzed people.

We have developed a virtual system where people can communicate with the computer without using any physical keyboard and mouse. This can lead to a new age of Human Computer Interaction in which physical contact with the computer would not be necessary at all. The use of object detection and image processing in OpenCV for the implementation of our work has proved to be practically successful and the task of keyboard and mouse is achieved with good precision. This system can be beneficial to certain people who have no control over their limbs. Most of the applications require additional hardware which are often very expensive. The motive of this work is to create this technology as cheaply as possible and to create it under a standardized operating system as well. Though, our system can be used as an alternative for physical keyboard and mouse, it still may perform less accurately in a low light condition. This is a concern for further research. Moreover, the work can be extended for a wide variety of environments and can be tested using the sophisticated existing models

The suggested system employs a real-time camera to control the mouse pointer and implement its purpose. We integrated mouse movement, icon selection, and features such as right, left, double click, and scrolling. This system uses picture comparison and motion detection technologies to perform mouse pointer motions and icon selection. Based on the data, we can predict that our system will run more efficiently if the algorithms can operate in all situations. This approach might be handy for presentations and for saving office space. We want to add more functions in the future, such as extending and shrinking windows, shutting windows, and so on, by utilizing the palm and several fingers.

12. SUMMARY

A Web camera is running on the mouse cursor. This will also lead to new levels of human-computer interaction (HCI), which does not require physical contact with the device. This machine can perform all mouse tasks centered on color recognition. This device is capable of being useful for interacting with contactless input modes. For people who don't use a touchpad, it's helpful. The architecture of the device proposed would dramatically change people's interactions with computers. Everyone is compatible with the Webcam, the microphone, and the mouse. It would eliminate the need for a mouse completely. It can also be used in gaming or any other independent application. **Free movement, left-click, right-click, drag/select, scroll-up, scroll-down** are all operations that can be performed using only gestures in this multi-Functional system. Most of the applications necessitate additional hardware, which can be quite costly. The aim goal was to develop this.

13. LIST OF FIGURES

14. REFERENCES

[1] The State of HCI in Ibero-American Countries. https://www.jucs.org/jucs_14_16/the_state_of_hci_in/jucs_14_16_2599_2613_granollers.pdf,

Last visited :03.01.2021

[2] Human-computer Interaction – Information, People, and .... https://psu.pb.unizin.org/ist110/chapter/5-2-human-computer-interaction/, Last visited :03.20.2021

[3] Challenges in Human-Computer Interaction Design for Mobile .... http://www.iaeng.org/publication/WCECS2009/WCECS2009_pp236-241.pdf, Last visited :04.01.2022

[4] CORE. http://core.ac.uk/display/21331086, Last visited :04.01.2021

[6] Hand Gesture Recognition System Using Camera. https://www.ijert.org/research/hand-gesture-recognition-system-using-camera-IJERTV3IS10567.pdf, Last visited :03.15.2021

[7] Gesture Recognition for Human-Robot Interaction. https://rahuls321.github.io/pdfs/Internship_Report_TCS.pdf, Last visited :04.01.2022

[8] Hand Gesture Recognition System - IJCTT. https://www.ijcttjournal.org/2017/Volume47/number-4/IJCTT-V47P133.pdf , Last visited :04.20.2022

[9] Sensors | Free Full-Text | Accurate Hand Detection ... - MDPI. https://www.mdpi.com/1424-8220/20/1/192 , Last visited :05.02.2021

[10] Hand Gesture Detection & Recognition System. http://www.diva-portal.org/smash/get/diva2:519237/FULLTEXT01.pdf , Last visited :03.26.2022

[11] Daniel Thalman, Gesture Recognition Motion Capture, Motion Retargeting, and Action Recognition, Last visited :03.28.2021

[12] A given interaction task may be examined to determine which style is best suited to the task. Mulder gave an overview of hand gestures in human-computer interaction, including hand movement categorization, common hand gestures, and interface design. Last visited :04.26.2022

[13] Kay M. Stanney HANDBOOK OF VIRTUAL ENVIRONMENTS Design, Implementation, and Applications, Gesture Recognition Chapter #10 by Matthew Turk, Last visited :03.11.2022

[14] Hatice Gunes, Massimo Piccardi, Tony Ja, 2007, Research School of Information Sciences and Engineering Australian National University Face and Body Gesture Recognition for a Vision-Based Multimodal Analyzer , Last visited :03.15.2022

[15] J. Heinzm ann and A. Zelinsky Robust Real - Time Face Tracking and Gesture Recognition , Last visited :04.30.2022

[16] Pang, Shanchen, et al. "An Artificial Intelligent Diagnostic System on Mobile Android Terminals for Cholelithiasis by Lightweight Convolutional Neural Network." PLoS One, vol. 14, no. 9, Public Library of Science, Sept. 2019, p. e0221720. Last visited :02.20.2021

[17] COMPARATIVE STUDY OF HAND GESTURE RECOGNITION SYSTEM. https://www.slideshare.net/cscpconf/comparative-study-of-hand-gesture-recognition-system ,

Last visited :04.30.2022

[18] Gesture Controlled Mouse Using Arduino. http://www.ijies.net/finial-docs/finial-pdf/24082161036.pdf , Last visited :03.10.2022

[19] The Gesture Interface: A Compelling Competitive Advantage .... https://www.edge-ai-vision.com/2013/04/the-gesture-interface-a-compelling-competitive-advantage-in-the-technology-race/ ,

Last visited :02.26.2021

[21] (PDF) Hand gesture recognition on python and opencv. https://www.researchgate.net/publication/349497837_Hand_gesture_recognition_on_python_and_opencv ,

Last visited :04.30.2022

[22] Installing Anaconda on Windows - Problem Solving with Python. https://problemsolvingwithpython.com/01-Orientation/01.03-Installing-Anaconda-on-Windows/, Last visited :04.17.2022

[23] Difference Between Anaconda and Python (With Table) – Ask .... https://askanydifference.com/difference-between-anaconda-and-python/ , Last visited :02.16.2022