

IMPLEMENTATION OF HAND HELD CURSOR CONTROL USING MACHINE LEARNING

A MINOR PROJECT REPORT

Submitted by

**Harish Kumar (RA1911008020026)
Abishek Narayan (RA1911008020049)
Ashwin Krishna (RA1911008020058)**

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY



SRM INSTITUTE OF SCIENCE AND TECHNOLOGY,

RAMAPURAM: CHENNAI 600089

NOVEMBER 2022

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY,
RAMAPURAM**

BONAFIDE CERTIFICATE

Certified that this project report “**IMPLEMENTATION OF HAND HELD CURSOR CONTROL USING MACHINE LEARNING**” is the bonafide work of “**Harish Kumar (RA1911008020026), Abishek Narayan (RA1911008020049), Ashwin Krishna (RA1911008020058)**” who carried out the minor project work under my supervision.

SIGNATURE

K.Danesh

SUPERVISOR

Dept of Information Technology

SRM Institute of Science & Technology

Ramapuram, Chennai – 600089

SIGNATURE

Dr. RAJESWARI MUKESH

HEAD OF THE DEPARTMENT

Dept of Information Technology

SRM Institute of Science & Technology

Ramapuram, Chennai - 600089

SUBMISSION OF MINOR PROJECT REPORT FOR VIVA-VOICE HELD ON _____

INTERNAL EXAMINER - 1

INTERNAL EXAMINER - 2

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	
1.	INTRODUCTION	1
2.	LITERATURE REVIEW	2
3.	SYSTEM ANALYSIS	6
4.	SYSTEM REQUIREMENTS	8
5.	SYSTEM ARCHITECTURE	9
6.	SYSTEM MODULES	10
7.	SYSTEM IMPLEMENTATION	19
8.	RESULT & DISCUSSION	21
9.	CONCLUSION	21
10.	FUTURE ENHANCEMENTS	22
	APPENDIX (Screenshots - Coding)	23
	REFERENCES	26

LIST OF FIGURES & TABLES

S. NO.	FIGURE NO.	FIGURE NAME	PAGE NO.
1.	1	Flowchart of the Proposed Method	9
2.	2	Flowchart of the Algorithm of Skin Detection	11
3.	3	Flowchart of the Algorithm of Hand Tracking	12
4.	4	Flowchart of the Algorithm of Cursor Control	13
5.	5	Display the Frame	15
6.	6	Mouse Movement	16
7.	7	Clicking Console	17
8.	8	Code Snippet 1	18
9.	9	Code Snippet 2	18
10.	10	Table of Recognized Rate of Input Samples	19
11.	11	Bar Graph Representing Recognized Rate of Input Samples	20
12.	12	User Hand Coordinates	20

LIST OF ABBREVIATION

S.NO	ABBREVIATION	EXPANTION
1	HCI	Human Computer Interaction
2	RGB	Red Green Blue
3	GUI	Graphical User Interface
4	ISL	Indian Sign Language
5	ASL	American Sign Language
6	CNN	Convolutional Neural Network

ABSTRACT

This project suggests a method for controlling cursor location with just your hands, independent of any machinery or electronics. Its foundation is HCI (Human Computer Interaction), in which each character is given a different set of motions to conduct actions like clicking and dragging objects with the use of their bare hands. All that will be needed for the system's input is a webcam. The built-in camera on laptops or the webcam on desktops records the user's motions, which are then processed using colour segmentation and detection techniques. The webcam on computers and the built-in camera on laptops both record the user's gesture as it moves. Python and OpenCV must be used as the programming languages to implement the suggested system. The system panels display the output from the camera so that the user can make further adjustments. (Continues incoming shift algorithm hand using adaptive mean shift. A series of visual words are used to extract the hand gesture features of the incoming shift algorithm (continuous adaptive mean shift), which are then categorised using a support vector machine. Mouse, NumPy, Math, and WX are the Python dependencies that were utilised to implement this project. Hand gesture tracking, hand area feature extraction, and feature classification make up the three primary components of this research.

CHAPTER-1

INTRODUCTION

In this project, hand gestures, a generally recognised language, are the most effective and expressive form of human communication. It is sufficiently expressive for both the deaf and the dumb to understand it. This paper suggests a real-time hand gesture system. The system's experimental design uses a fixed camera on a laptop, or a low-cost, high-definition web camera positioned in a fixed location on top of a computer monitor to take snapshots in the Red, Green, and Blue [RGB] colour space from a fixed distance. Feature matching, region extraction, feature extraction, and image pre-processing are the four processes that make up this task. One of the main problems with communicating with the deaf and hard of hearing is the recognition and interpretation of sign language. The pre-processing, background subtraction, and edge detection techniques have been combined to provide an efficient hand gesture segmentation technique in this study. Pre-processing is the process of preparing data for another process, according to the definition. The fundamental goal of the pre-processing step is to change the data into a format that can be processed more quickly and easily. Pre-processing techniques are developed in the proposed work based on various combinations of the subsequent hand gesture image processing operations, such as image capture, noise removal, background subtraction, and edge detection. These image processing methods are discussed in the sections that follow. The hand gestures can be observed with various interfaces, such as "data gloves," which accurately record every abduction angle and digit, as well as position sensors for wrists and optical orientation or electromagnetic, requiring the user to wear trackers or gloves. Initially, the hand gesture images are captured from the vision-based camera. The user's comfort and interface time are typically minimised by glove-based interfaces, which even require the user to be attached to the computer. In contrast, vision-based interfaces allow for freer human involvement. With the help of this research, we hope to develop free hand recognition software for laptops and desktop computers that support web cameras. The project focuses on developing a hand recognition tool that can be used to move the mouse pointer, carry out basic actions like clicking, and carry out additional hand gesture operations like moving files between computers using deft socket programming and carrying out basic yet fascinating actions that can be covered by the hand recognition.

CHAPTER-2

LITERATURE SURVEY

“Virtual Mouse Control Using Coloured Finger Tip and Hand Gesture Recognition”, V. V. Reddy, T. Dhyana Chand, G. V. Krishna and S. Maheswaran, in human-computer interaction, virtual mouse implemented with fingertip recognition and hand gesture tracking based on image in a live video is one of the studies. In this paper, virtual mouse control using fingertip identification and hand gesture recognition is proposed. This study consists of two methods for tracking the fingers, one is by using coloured caps and other is by hand gesture detection. This includes three main steps that are finger detection using colour identification, hand gesture tracking and implementation on on-screen cursor. In this study, hand gesture tracking is generated through the detection of the contour and formation of a convex hull around it. Features of hands are extracted with the area ratio of contour and hull formed. Detailed tests are performed to check this algorithm in real world scenarios.

“Virtual Mouse Using Object Tracking”, M. Shetty, C. A. Daniel, M. K. Bhattar and O. P. Lopes. With new changes seen in computer technology day by day, it has become quite essential for us to find specific new ways of interaction with computer systems as its need is increasing in society every day. Today, every device is making the use of touch screen 5 technology on its systems, which isn't affordable to be used in all applications. A specific interactive module like a virtual mouse that makes use of Object Tracking and Gestures that will help us to interact can be an alternative way for the traditional touch screen and the physical mouse. The objective is to create an Object Tracking application that interacts with the system.

“Hand gesture recognition using deep learning”, S. Hussain, R. Saxena, X. Han, J. A. Khan and H. Shin, In order to offer new possibilities to interact with machine and to design more natural and more intuitive interactions with computing machines, our research aims at the automatic interpretation of gestures based on computer vision. In this paper, we propose a

technique which commands computer using six static and eight dynamic hand gestures. The three main steps are: hand shape recognition, tracing of detected hand (if dynamic), and converting the data into the required command.

“Hand gesture recognition using deep learning”, Abha Ingle, Karan Jadhav, Ratish Kumar Jha, Rakshit Karkera, Hand gesture is an effective form for everyone to communicate. Hand gesture recognition can be seen as a way for computers to better understand human body language, creating a richer relationship between humans and computers rather than the traditional text-based or GUI-based approach. Hand gestures can simplify tasks by giving commands to the computer with simple hand movements. Camera captures the gestures, computer interprets, and proper response action is taken. This project is based on hand gesture recognition, which captures hand gestures and converts them into machine-understandable commands and performs the appropriate tasks. . In this project, we will first capture the gesture using the camera. The captured image is then pre-processed for feature extraction and implements a convolutional neural network using a back-propagation algorithm to effectively recognize gestures and train the machine

“Human hand gesture-based system”, Horatiu-Stefan Girf, Train Ture, the goal of the paper is to improve the recognition of the human hand postures in a Human Computer Interaction application, the reducing of the time computing and to improve the user comfort regarding the used human hand postures. The authors developed an application for computer mouse control. The application based on the proposed algorithm, hand pad colour and on the selected hand feature presents good behaviour regarding the time computing. The user has an increased comfort in use of the system due to the proposed hand postures. Also, the system works well having the same behaviour under very low illuminance level and high illuminance level.

“Vision-based hand gesture recognition using deep learning for the interpretation of sign language”, Sakshi Sharma, Sukhwinder Singh, Hand gestures have been a key component of communication since the dawn of the era. Hand gestures are the basis of sign language, which is a visual form of communication. In this paper model has a compact representation that achieves better classification accuracy with fewer model parameters than other existing CNN architectures. In order to evaluate the effectiveness of this model, VGG-

11 and VGG-16 were also trained and tested in this work. 2 datasets were considered for performance evaluation. This work contains, a large collection of Indian Sign Language (ISL) gestures, and second, a publicly available American Sign Language (ASL) dataset. The performance of the proposed VGG-11 and VGG-16 system is experimentally evaluated and compared with existing state-of-the-art approaches. In addition to accuracy, other efficiency indices were also used to determine the robustness of the proposed work. The findings indicate that the proposed model outperforms existing techniques as it has the potential to classify maximal gestures with minimal error rates. The model is also tested with extended data and found to be invariant to rotation and scaling transformation.

“Gesture Tracking and Recognition Algorithm for Dynamic Human Motion Using Multimodal Deep Learning”, Zhonghua Xia, Jingming Xing, Xiaofeng Li, to address the problems of the traditional human motion gesture tracking and recognition methods, such as poor tracking effect, low recognition accuracy, high frame loss rate, and long-time cost, a dynamic human motion gesture tracking and recognition algorithm using multimode deep learning was proposed. Firstly, the collected human motion images are repaired in the three-dimensional (3D) environment, and the multimodal 3D human motion model is reconstructed using the processed images. Secondly, according to the results of model reconstruction, the camera gesture and other parameters of the keyframe are used to construct the target tracking optimization function so as to achieve the purpose of accurate tracking of human motion. Finally, for multimodal human motion gesture learning, a convolutional neural network (CNN) is developed. The trained CNN is utilized to complete dynamic human motion recognition after convolutional and pooling calculations. The results demonstrate that the proposed algorithm is effective in tracking human motion gestures.

“Hand Gesture Recognition using Deep Learning Neural Networks”, Norah Mashari, Gesture recognition concerns non-verbal motions used as a means of communication in HCI. A system may be utilised to identify human gestures to convey information for device control. This represents a significant field within HCI involving device interfaces and users. The aim of gesture recognition is to record gestures that are formed in a certain way and then detected by a device such as a camera. Hand gestures can be used as a form of communication for many

different applications. It may be used by people who possess different disabilities, including those with hearing-impairments, speech impairments and stroke patients, to communicate and fulfil their basic needs.

“Research on the Hand Gesture Recognition Based on Deep Learning”, Jing-Hao Sun, Ting-Ting Ji, Shu-Bin Zhang, Jia-Ku Yang, with the rapid development of computer vision, the demand for interaction between human and machine is becoming more and more extensive. Since hand gestures are able to express enriched information, the hand gesture recognition is widely used in robot control, intelligent furniture and other aspects. The paper realizes the segmentation of hand gestures by establishing the skin colour model and AdaBoost classifier based on Har according to the particularity of skin colour for hand gestures, as well as the denaturation of hand gestures with one frame of video being cut for analysis. In this regard, the human hand is segmented from the complicated background, the real-time hand gesture tracking is also realized by Cam Shift algorithm.

Background object in the same skin colour. So, it is very important that the colour determination algorithm works accurately. The system works by identifying the colour of the hand and deciding the cursor position accordingly, but there are various conditions and scenarios that make it difficult to run the algorithm in a real environment. The system can work for any colour skin tone, just as it can work precisely under any lighting conditions, just as for the click that a user needs to make a 15-degree angle between his two fingers, the proposed system can easily replace a traditional mouse cursor

CHAPTER-3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

The current system only has a generic mouse and trackpad for controlling monitors, and there is no hand gesture mechanism. It is not possible to remotely view a monitor's screen using a hand motion. The scope is merely constrained to the area of the virtual mouse, despite the fact that it is mostly aiming to accomplish. The current virtual mouse control technology allows us to do standard mouse operations like mouse pointer control, left click, right click, drag, etc. by employing a hand recognition algorithm. The hand recognition technology is not being used any further. Despite the fact that there are numerous systems for hand recognition, the system they used is static hand recognition, which merely recognises the shapes made by hands while defining an action for each shape made. This system is constrained to a small number of defined actions and causes a lot of confusion.

3.2 PROPOSED SYSTEM

Even though there are a number of quick access options for the hand and mouse gesture for laptops in the current system, with our project, we could use the laptop or webcam and by recognising the hand gesture, we could control the mouse and perform basic operations like mouse pointer control, select and deselect using the left click, and a quick access feature for file transfer between the systems connected via network LAN cable. A "Zero Cost" hand identification system for computers was developed. It uses straightforward algorithms to identify the hand, track hand movements, and assign a response to each movement. However, we have mainly focused on mouse pointing and clicking activities as well as a hand movement and action for transferring files between connected devices. Since Python is a simple language, platform independent with flexibility, and portable, the system we are designing will be much more responsive and easier to implement. This is beneficial in creating a programme that is centred in

such a goal for creating a Virtual Mouse and Hand Recognition system. By specifying actions for the hand movement for carrying out a given action, the system can be expanded much further. By using such movements for the set of hand gestures, it might be altered to any additional degree; your creativity is the only limit. Without the need for additional hardware, this virtual mouse hand recognition application employs a straightforward colour cap on the finger to allow users to direct the cursor using basic hand and gesture movements. This is accomplished using camera inputs in conjunction with hand gesture detection based on eyesight.

CHAPTER-4

SYSTEM REQUIREMENTS

4.1 HARDWARE REQUIREMENTS

This project requires a Computer, Laptop or a MAC.

A RAM of 8GB

A processor of Intel core i5 8th generation or higher.

A stable internet connection is required for network stability.

Other hardware materials like Webcam, mouse, keyboard etc.

4.2 SOFTWARE REQUIREMENTS

An Operating system such as Windows or Linux operating systems.

A Web browser such as Chrome, Fire Fox or Opera.

Any python platform for coding process.

Language used for this project is python programming.

Code editors like Visual Studio Code, PyCharm etc.

CHAPTER-5

SYSTEM ARCHITECTURE

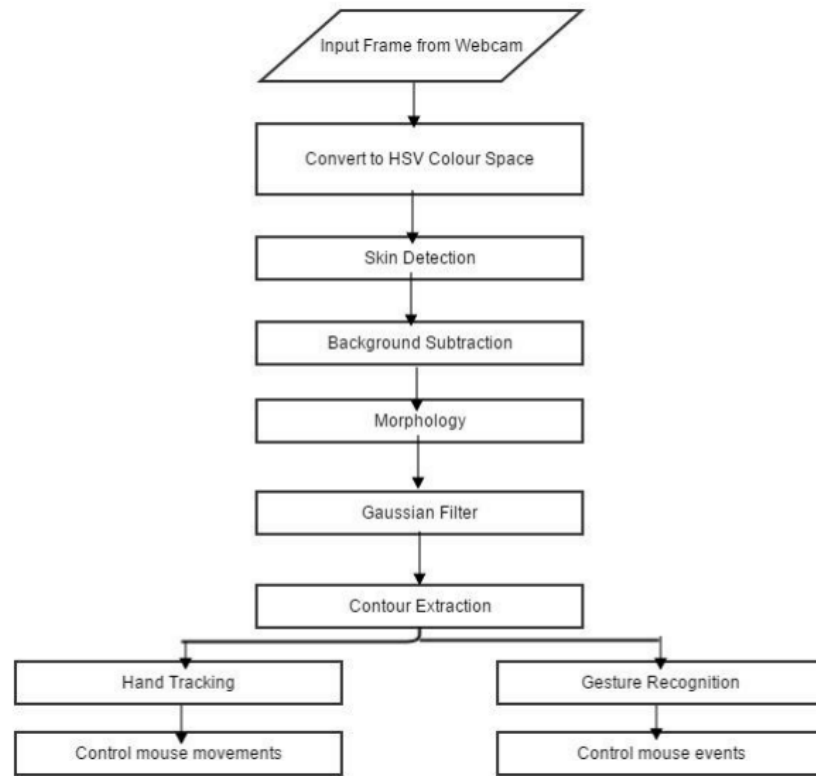


Figure.1 Flowchart of the proposed method

The strategies and procedures employed in the design and development of the vision-based CC system will be described in this part. “**Figure.1**” following depicts the overall system's algorithm. The image can be transformed to chrominance colour space, which is less sensitive to changes in illumination, to lessen the impact of illumination. The HSV colour system was selected because it was thought to be the most effective colour scheme for detecting skin. Then, background subtraction was used to get rid of the face and other background items with flesh tones. To enhance edge detection and smooth the image, a Gaussian filter was used.

CHAPTER-6

SYSTEM MODULE

6.1 SKIN DETECTION

The term "skin detection" refers to the process of identifying skin-coloured pixels in a picture. It is a critical stage in many images processing applications, including hand tracking, face detection, and hand gesture identification. Since it is computationally efficient and offers reliable information that is resistant to scaling, rotation, and partial occlusion, skin identification utilising colour information has recently attracted a lot of attention. Since lighting, camera features, backdrop, and ethnicity all affect how skin appears in photos, detecting skin using colour information can be difficult. The image can be changed to a chrominance colour space, which is less affected by variations in illumination, to lessen the impacts of illumination. A chrominance colour space is one in which the information about intensity (luminance) and colour are distinct (chromaticity). The HSV colour space and the Histogram-based skin detection approach were combined in the suggested method. Three channels—Hue (H), Saturation (S), and Value—make up the HSV colour system (V). The colour information is contained in the H and S channels, while the intensity information is contained in the V channel. The webcam's input image would be in the RGB colour space, so it would need to be changed using the conversion formulae to the HSV colour system. The colour of this region is transformed to a chrominance colour space using a tiny skin region. The region's 32 bin histogram is then discovered and used as the histogram model. The chance that a given pixel belongs to a histogram model is then assessed for each pixel in the image. Another name for this technique is Histogram Back Projection. Back projection is the process of capturing how well individual pixels or groups of pixels fit into a histogram model. The end result would be a back-projected grayscale image, where the intensity would denote how likely a pixel is to be skin-coloured. **Figure.2** under the Findings and Analysis section would show the skin detection results.

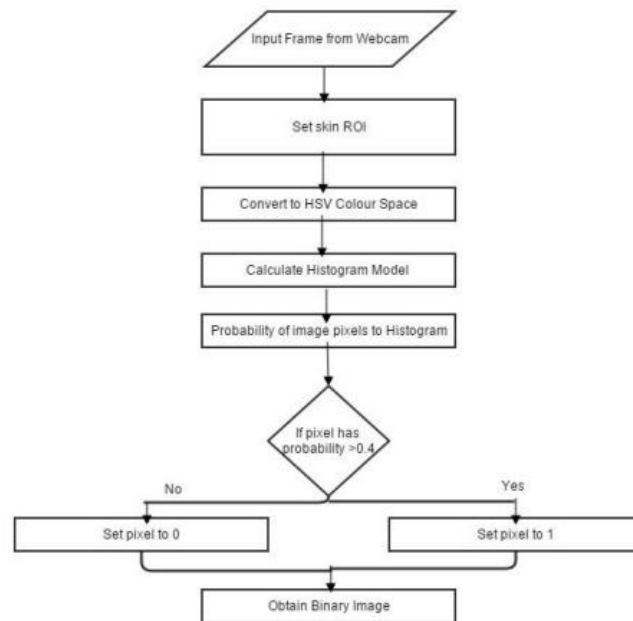


Figure.2 Flowchart of the algorithm of skin detection

6.2 HAND TRACKING

The tip of the index finger moved the cursor in different directions. The centre of the palm must be located before the tip of the index finger may be located. The advantage of the approach employed to determine the hand's centre is that it is straightforward and simple to use. The flow chart of Figure below depicts the algorithm for the procedure. Each point inside the circle was measured for the shortest distance to the contour, and the place with the greatest distance was noted as the hand centre. The radius of the hand was calculated as the distance between the hand's centre and its contour. Each subsequent frame's hand centre was calculated, and with the help of the hand centre, the tip of the index finger was located and used for hand tracking. The following subsection describes how to distinguish between the index and the other fingers. **Figure.3** under the Findings and Analysis section would show the hand tracking results.

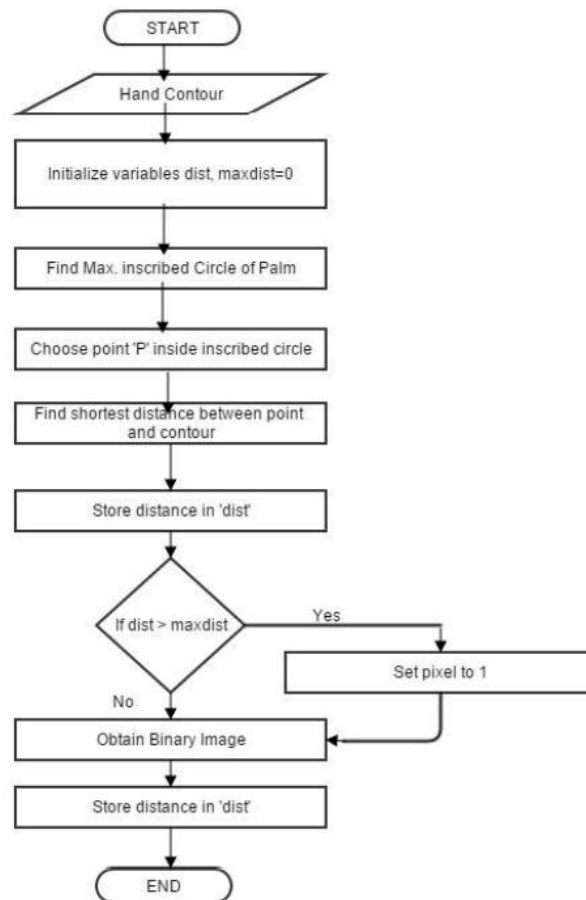


Figure.3 Flowchart of the algorithm of hand tracking

6.3 CURSOR CONTROL

It will be easy to map various hand gestures to different mouse actions after the hand movements have been recognised. It turns out that using the C/C++ programming language to control the computer cursor is not too difficult. The "Send Input" function of the application will enable cursor control by including the User.lib library. The Microsoft Developers Network provided guidelines for using this function correctly. Only computers running Windows 2000 Professional or later are capable of using this feature. As a result, the system now has a new restriction that prevents it from functioning on older Windows operating system versions. **Figure.4** following depicts the cursor control algorithm.

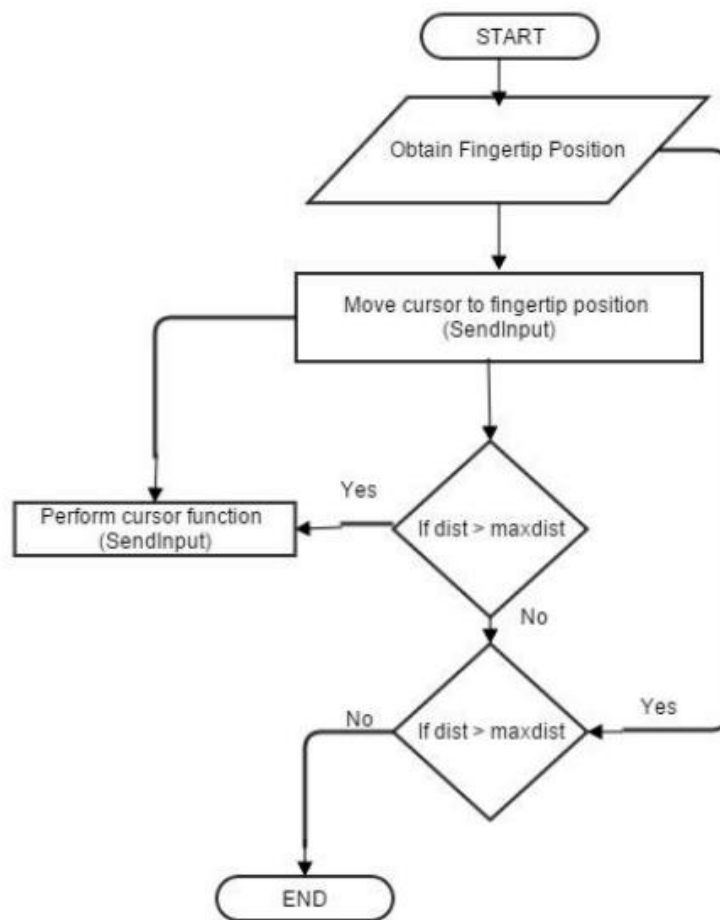


Figure.4 Flowchart of the algorithm of cursor control

CHAPTER-7

IMPLEMENTATION

7.1 CAMERA SETTING

The webcam of the connected laptop or desktop controls the runtime operations. We must establish a Video Capture object in order to record video. Either the device index or the filename of a video can be used as its parameter. Just a number to identify which camera, the device index. We pass it as "0" since we only use one camera. Additional cameras can be added to the system and passed as 1, 2, and so forth. After that, you can take individual frames of video. Don't forget to release the capture, though, at the conclusion. By making small changes to the algorithm, we could also use colour detecting methods on any image.

7.2 CAPTURING FRAMES

It uses an infinite loop to enable the web camera to every time, grabs the frames, and is active during the entire program's duration. We record the live stream. frame by frame, in a stream. After that, we examine each captured frame in HSV colour from RGB (the default) colour space. More than 150 colour-space conversions exist. ways that OpenCV offers. But we'll focus on only two. Either of the following: BGR to Gray and BGR to HSV.

7.3 DISPLAY THE FRAME

High Gui's imshow() method must routinely invoke wait Key in order to work. Calling wait Key completes the event loop processing of the imshow() method. The wait Key () function keeps frame following as **Figure.5**

track of key events for a "delay" (in this case, 5 milliseconds). High Gui processes Windows events like redraw, resizing, input event, etc. Consequently, even with a 1 Ms delay, we call the wait Key function.

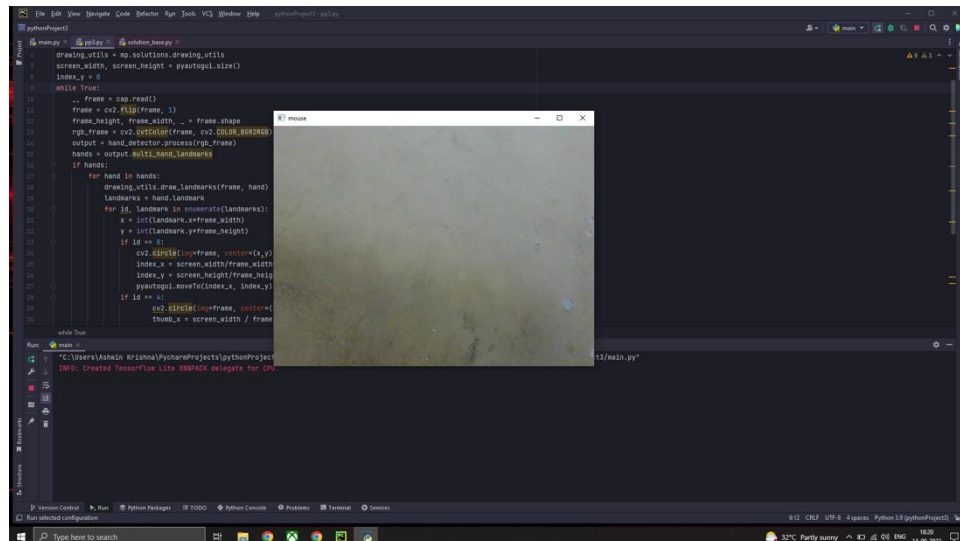


Figure.5 Display the frame

7.4 MOUSE MOVEMENT

We must first determine the centres of both red objects that have been spotted, which can be done quickly by averaging the maximum and minimum points of the bounding boxes. Now that we have two coordinates taken from the centres of the two objects, we can compute their average and obtain the red point in the illustration. The detected coordinate is being converted from camera resolution to actual screen resolution as **Figure.6**. The location was then set as the mouse position. However, it will take some time to reposition the mouse pointer. Therefore, we must wait till the mouse pointer gets to that location. So, we started a loop and are just waiting to see if the current mouse location and the assigned mouse location match. For the open gesture, that is.

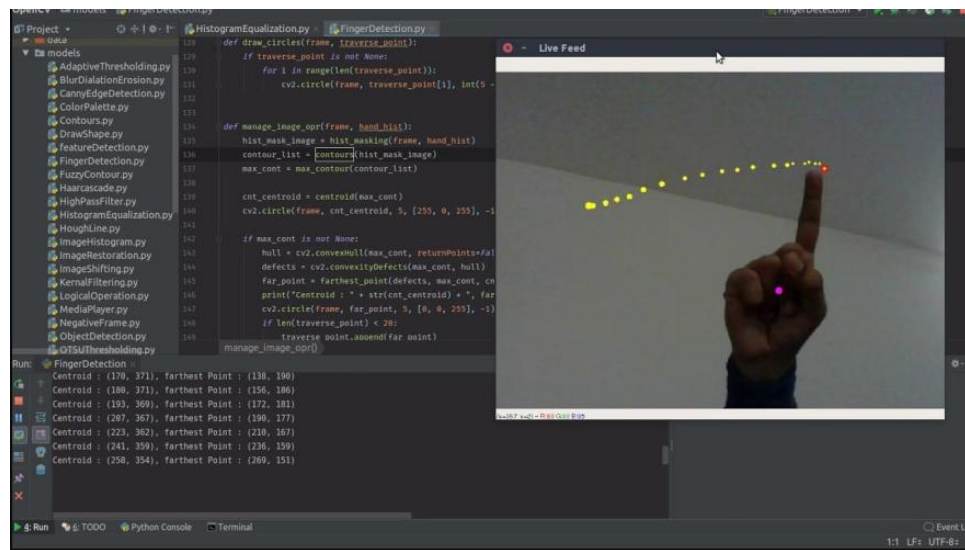


Figure.6 Mouse movement

7.5 CLICKING

Applying the close gesture is the next stage. The action is carried out by clicking and dragging the object. It is comparable to the open gesture, but since there is only one object

present, we simply need to determine its centre. And that will be put in the spot where our mouse pointer will be set console **Figure.7**. We shall employ a mouse press operation in place of a mouse release operation.

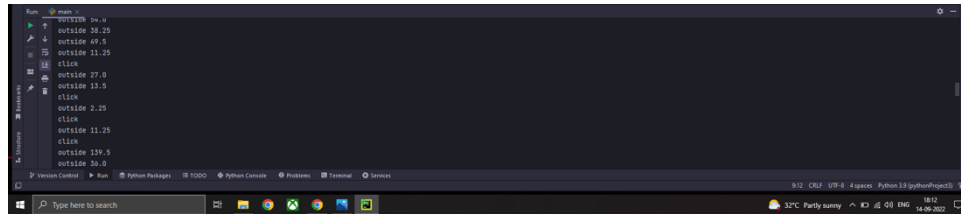


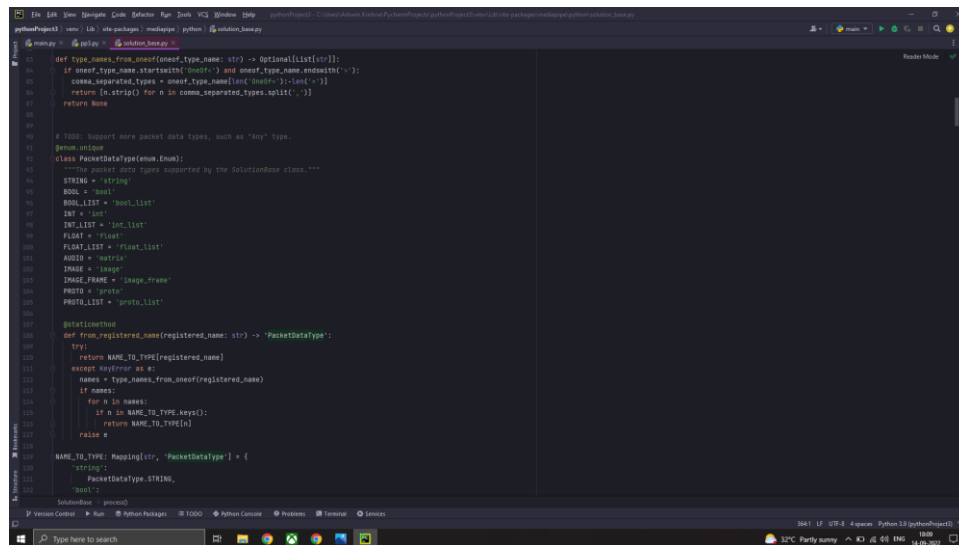
Figure.7 Clicking console

7.6 DRAG

A variable called "pinch flag" is introduced in order to accomplish the dragging. If it was previously clicked, it will be set to 1. Therefore, once we locate the open gesture, we click and then verify that the pinch flag is set to 1. The drag action is carried out if it is set to one; else, the mouse move operation is carried out.

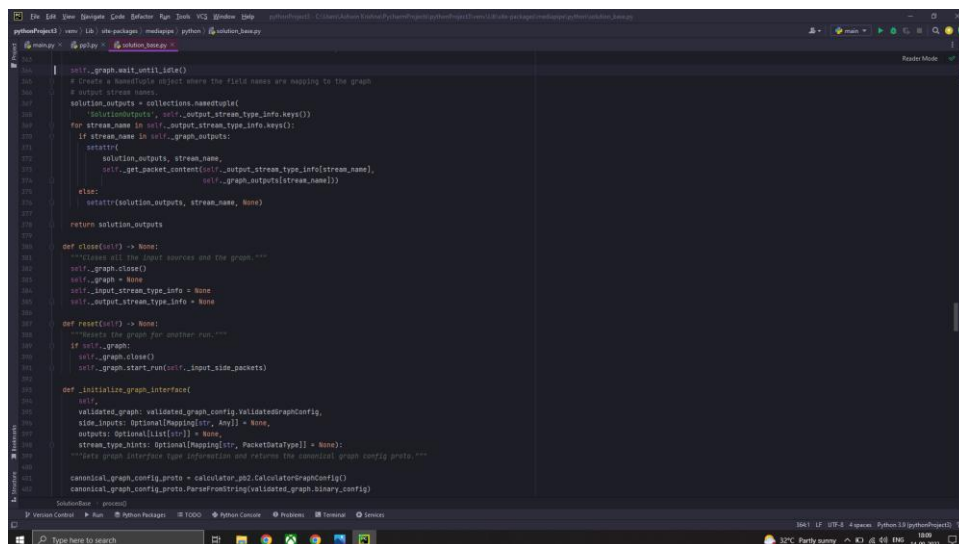
7.7 DND FRAME

My File Drop Target is the first class that is created. We have one overridden method inside of that, On Drop Files. The file paths that are dropped along with the mouse's x/y position are both accepted by this approach.



```
10 def type_names_from_oneof(oneof_type_name: str) -> Optional[List[str]]:
11     if isinstance(oneof_type_name, str) and isinstance(oneof_type_name, str):
12         comma_separated_types = oneof_type_name[oneof_type_name.find(','):]
13         return [n.strip() for n in comma_separated_types.split(',')]
14     return None
15
16 # TODO: Support more packet data types, such as "key" type.
17 @enum.unique
18 class PacketDataType(enum.Enum):
19     """The packet data types supported by the SolutionBase class."""
20     STRING = 'string'
21     BOOL = 'bool'
22     BOOL_LIST = 'bool_list'
23     INT = 'int'
24     INT_LIST = 'int_list'
25     FLOAT = 'float'
26     FLOAT_LIST = 'float_list'
27     MATRIX = 'matrix'
28     IMAGE = 'image'
29     IMAGE_FRAME = 'image_frame'
30     PHOTO = 'photo'
31     PHOTO_LIST = 'photo_list'
32
33     @staticmethod
34     def from_registered_name(registered_name: str) -> 'PacketDataType':
35         try:
36             return NAME_TO_TYPE[registered_name]
37         except KeyError as e:
38             names = type_names_from_oneof(registered_name)
39             if names:
40                 for n in names:
41                     if n in NAME_TO_TYPE.keys():
42                         return NAME_TO_TYPE[n]
43             raise e
44
45 NAME_TO_TYPE: Mapping[str, 'PacketDataType'] = {
46     'string':
47         PacketDataType.STRING,
48     'bool':
49         PacketDataType.BOOL,
50 }
```

Figure.8 Code Snippet (1)



```
100 def __graph_wait_until_ready(self):
101     """Wait until the graph is ready to be used.
102     # output stream names.
103     solution_outputs = collections.namedtuple(
104         'SolutionOutputs', self._output_stream_type_info.keys())
105     for stream_name in self._output_stream_type_info.keys():
106         if stream_name in self._graph_outputs:
107             solution_outputs, stream_name,
108             self._get_packet_content(self._output_stream_type_info[stream_name],
109                                     self._graph_outputs[stream_name])
110         else:
111             setattr(solution_outputs, stream_name, None)
112     return solution_outputs
113
114 def close(self) -> None:
115     """Close all the input sources and the graph."""
116     self._graph.close()
117     self._graph = None
118     self._input_stream_type_info = None
119     self._output_stream_type_info = None
120
121 def reset(self) -> None:
122     """Reset the graph for another run."""
123     if self._graph:
124         self._graph.close()
125         self._graph.start_run(self._input_side_packets)
126
127 def _initialize_graph_interface(self):
128     self._
129     validated_graph: ValidatedGraphConfig, ValidatedGraphConfig,
130     side_inputs: Optional[Mapping[str, Any]] = None,
131     outputs: Optional[List[str]] = None,
132     stream_type_hints: Optional[Mapping[str, PacketDataType]] = None):
133     """Get graph interface type information and returns the canonical graph config proto."""
134     canonical_graph_config_proto = calculator_pb2.CalculatorGraphConfig()
135     canonical_graph_config_proto.ParseFromString(validated_graph.binary_config)
```

Figure.9 Code Snippet (2)

CHAPTER-8

RESULT AND EVALUATION

We made an effort to concentrate on supporting those who are unable to control their limbs and also improvise the movements of interaction between humans and machines. Our goal aimed to produce this technology in the most affordable manner feasible. and also, to construct it under a uniform operating system. The suggested system operates mouse functionalities as shown in the table **Figure.10**. Perform mouse operations by detecting red with the pointer. such as dragging, cursor movement, left click between two connected systems.

EVENTS	No. of input samples	No. of Recognized samples	Recognized rate
Move to cursor up	5	4	0.8
Move to cursor down	5	3	0.6
Move the cursor left	5	3	0.6
Move the cursor right	5	4	0.8
Drag To	5	3	0.6
Move To	5	4	0.8
Click	5	3	0.6
Increased speed of cursor	5	3	0.6

Figure.10 Table of Recognized rate of input samples

This How to use the mouse to control things that are red in colour. The user applies red-coloured objects on the tip of their finger for improved performance When there are just two contours, then it executes a straightforward mouse movement motion. Alternatively, if the number of contours is 1, it presses the left mouse button. Additionally, this system supports the basic transmission of files within the same system between two or more an internet connection. The computer's screen's left side serve as a route of communication between the systems. i.e., the Drag and drop the file you want to copy on the left side. of the display of the computer. The dropped file will then be copied to the receiver system or the destination.

The basic goal of this system is to consume less hardware. elements connected to the computer. Despite the an application can be run on a regular PC using a web browser. camera. but ideally it needs to include a front camera with at least 2MP. with a minimum Pentium processor and 256 MB of RAM.

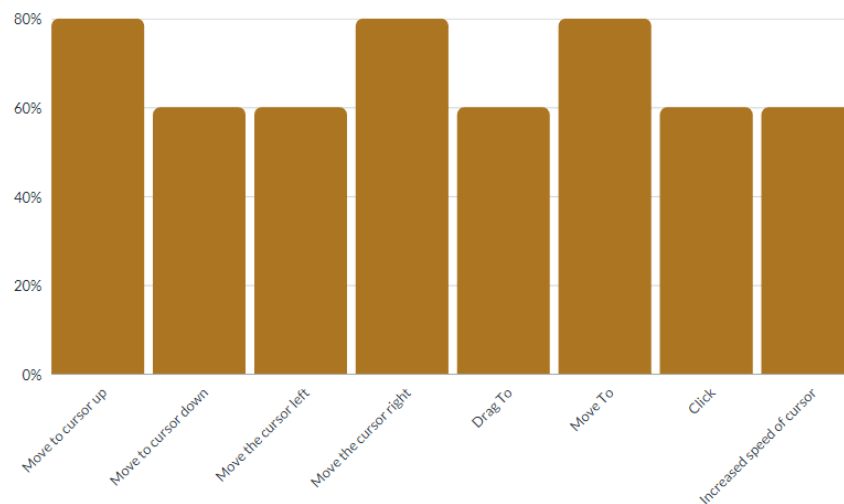


Figure.11 Bar Graph representing Recognized rates of input samples

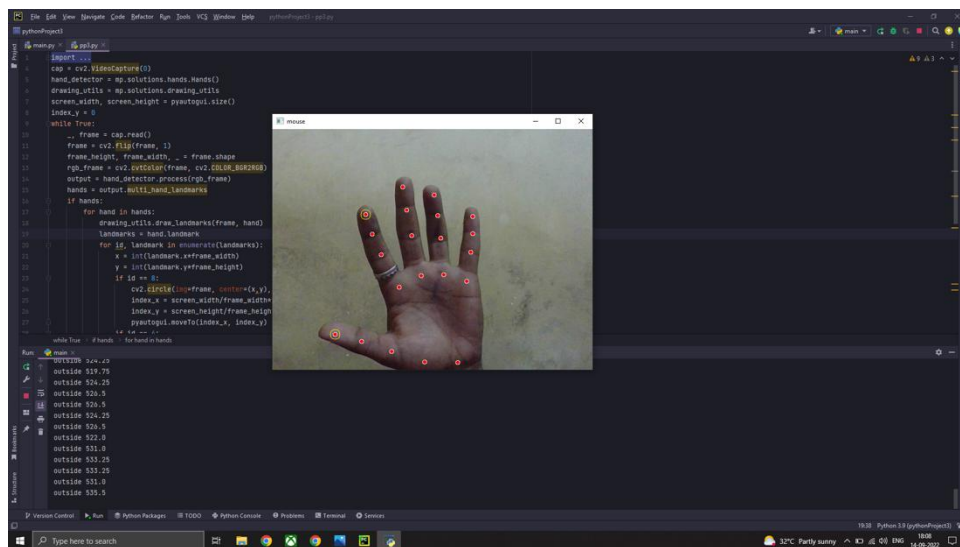


Figure.12 User hand coordinates

CHAPTER-9

CONCLUSION

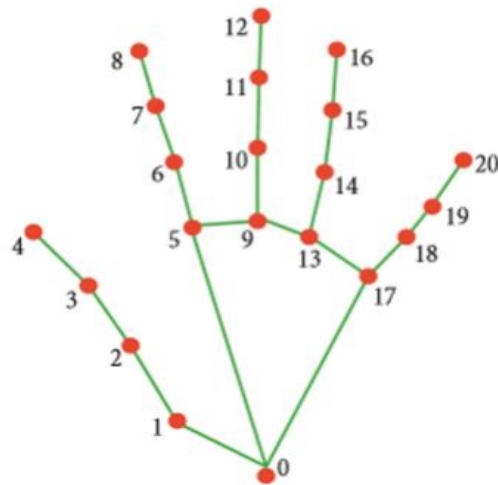
The most effective human-machine interface is provided through gesture recognition. The development of alternative human computer interaction modes depends on gesture recognition. It makes it easier for people to interact more naturally with machines. Many different applications, including robot control and sign language recognition for the deaf and dumb, can use gesture recognition. A wide range of industries, including augmented reality, computer graphics, gaming, prosthetics, and biomedical instruments, can benefit from this technology. In a variation of our system known as Digital Canvas, which is gaining popularity among painters, the artist can use the hand as a brush and the Virtual Mouse technology to produce 2D or 3D images, with a Virtual Reality kit or a monitor serving as the display set. This work can be further extended to make the system more adaptable to various lighting conditions and background complexity.

CHAPTER-10

FUTURE ENHANCEMENT

To make the application more user-friendly, accurate, and versatile in various situations, further features and future effort are required. Improved Performance and Accuracy: The hardware of the system, which includes the processing speed of the processor, the size of the available RAM, and the available functionalities of the webcam, heavily influences how quickly the device reacts. As a result, this system may also function better when used with compatible devices that have a webcam that performs better in particular lighting conditions. The method can also be employed in the future on Android devices, where hand gestures will take the place of the touchscreen notion.

APPENDIX



- | | |
|-----------------------|-----------------------|
| 0. WRIST | 11. MIDDLE_FINGER_DIP |
| 1. THUMB_CMC | 12. MIDDLE_FINGER_TIP |
| 2. THUMB_MCP | 13. RING_FINGER_MCP |
| 3. THUMB_IP | 14. RING_FINGER_PIP |
| 4. THUMB_TIP | 15. RING_FINGER_DIP |
| 5. INDEX_FINGER_MCP | 16. RING_FINGER_TIP |
| 6. INDEX_FINGER_PIP | 17. PINKY_MCP |
| 7. INDEX_FINGER_DIP | 18. PINKY_PIP |
| 8. INDEX_FINGER_TIP | 19. PINKY_DIP |
| 9. MIDDLE_FINGER_MCP | 20. PINKY_TIP |
| 10. MIDDLE_FINGER_PIP | |

Figure.13 Commonly used hand coordinates

```

import cv2

cap = cv2.VideoCapture(0)
hand_detector = mp.solutions.hands.Hands()
drawing_utils = mp.solutions.drawing_utils
screen_width, screen_height = pygame.size()
index_y = 0

while True:
    frame = cap.read()
    frame = cv2.flip(frame, 1)
    frame_height, frame_width, _ = frame.shape
    rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    output = hand_detector.process(rgb_frame)
    hands = output.multi_hand_landmarks

    if hands:
        for hand in hands:
            drawing_utils.draw_landmarks(frame, hand)
            landmarks = hand.landmark
            for id, landmark in enumerate(landmarks):
                x = int(landmark.x * frame_width)
                y = int(landmark.y * frame_height)
                if id == 0:
                    cv2.circle(frame, (x, y), radius=10, color=(0, 255, 255))
                    index_x = screen_width / frame_width * x
                    index_y = screen_height / frame_height * y
                    pygame.mouse.set_pos(index_x, index_y)
                    index_y -= 10
    
```

Figure.14 Code Snippet

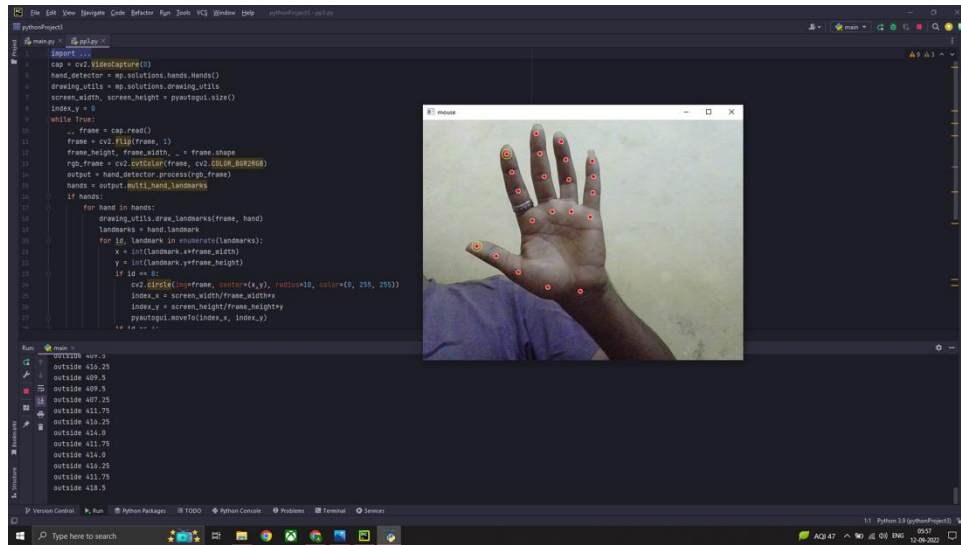


Figure.15 User hand detected

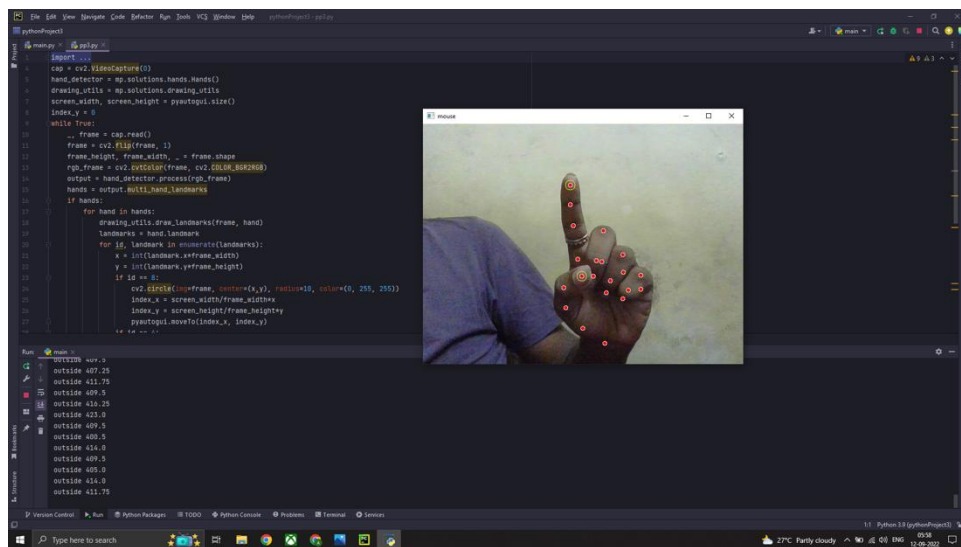


Figure.16 Index Finger Detected

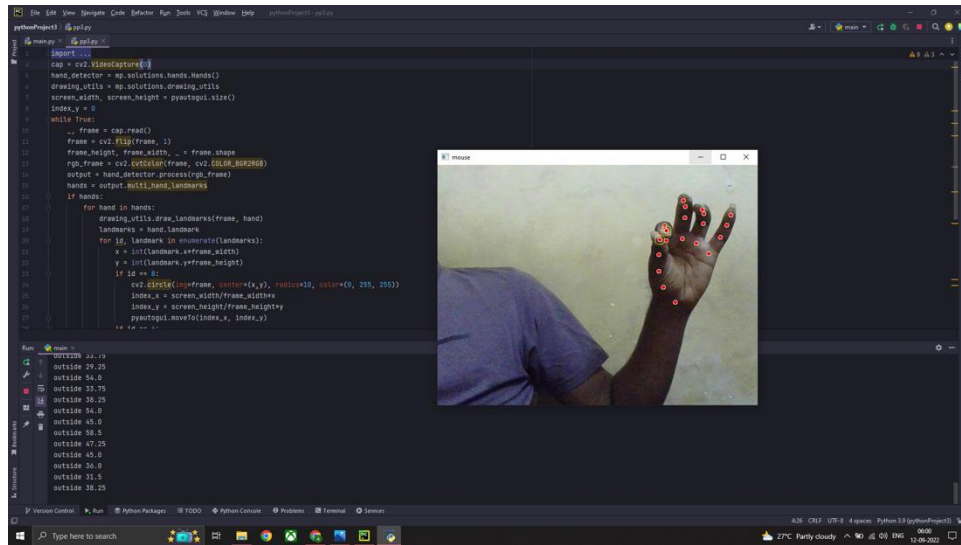


Figure.17 Clicking using index and thumb fingers

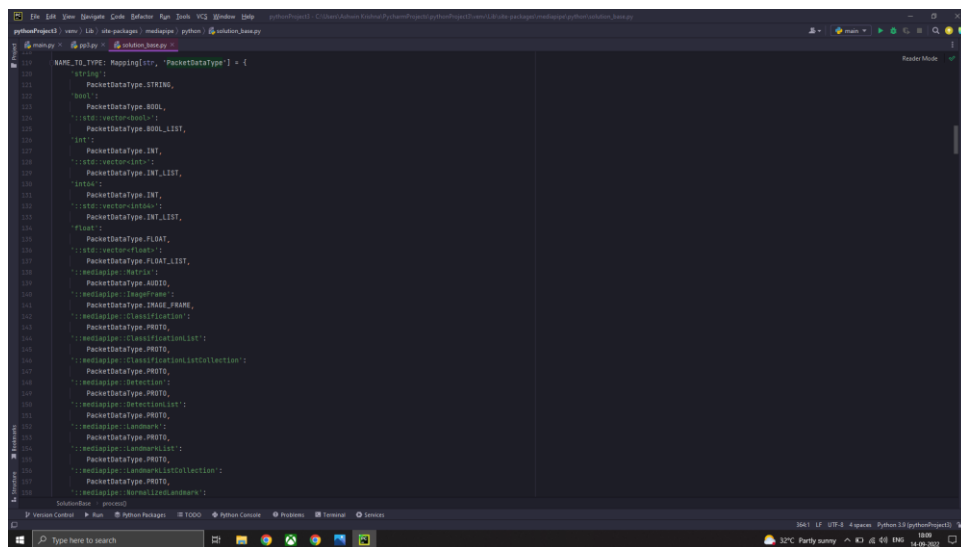


Figure.18 Code console base

ORIGINALITY REPORT

20%

SIMILARITY INDEX

13%

INTERNET SOURCES

4%

PUBLICATIONS

7%

STUDENT PAPERS

PRIMARY SOURCES

1

www.irjet.net

Internet Source

9%

2

Submitted to Visvesvaraya Technological University, Belagavi

Student Paper

5%

3

Dharmaraj Ojha, Rajesh George Rajan. "Histogram based Human Computer Interaction for Gesture Recognition", 2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA), 2019

Publication

2%

4

Roshnee Matlani, Roshan Dadlani, Sharv Dumbre, Shruti Mishra, Abha Tewari. "Virtual Mouse using Hand Gestures", 2021 International Conference on Technological Advancements and Innovations (ICTAI), 2021

Publication

1%

5

thecodacus.com

Internet Source

1%

IMPLEMENTATION OF HAND-HELD CURSOR CONTROL USING MACHINE LEARNING

¹Danesh,² Harish Kumar, ³ Abishek Naryan, ⁴ Ashwin Krishna

^{1, 2, 3, 4} Department of Information Technology, SRM Institute of Science and Technology,
Ramapuram, Chennai,

E-mail: 1 danesh.kn1@gmail.com, 2 hk7282@srmist.edu.in, 3 an5267@srmist.edu.in,

4 ak7160@srmist.edu.in

ABSTRACT

The project proposes an approach to cursor control that does not rely on mechanical or electrical components and may be performed entirely by hand. Based on HCI principles, players control their characters by moving their bodies in certain ways to do common tasks like clicking and dragging. To feed data into the system, just a camera will be required. The user's actions are captured by the laptop's or desktop's camera and then analysed using detection methods. Laptop and desktop webcams capture the user's hand movements in real time. Implementing the proposed system requires the programming languages Python and OpenCV. In order to fine-tune the system, users may see the camera's output on the system's panels. The focus of this study is on three main areas: monitoring hand gestures; extracting hand area features; and classifying those extracted features.

INTRODUCTION

Humans have been using hand tracking for a very long time. This piece of art represents an honest action. First, you'll need to pick up a low-priced camera that may be utilised for data entry. The complete method includes the steps of acquiring data, analysing those images, isolating certain regions, and comparing those extracted features. Given that it employs a camera to keep an eye on users' hands, it may potentially be considered cutting-edge gear. The end goal of this work is to allow users to operate their computers and

other devices with gestures rather than by touching the screen or pointing and clicking a mouse. A gesture is a kind of nonverbal communication that includes the use of hand, facial, or body movement to convey meaning. As an input for controlling machines, gestures use a mathematical algorithm to halt human movement. The variety of image processing algorithms we use in this effort helps us keep costs down.

EXISTING SYSTEM

There is no way to employ hand gestures in the present system, and the only means of manipulating displays are the standard mouse and trackpad. There is no way to see the screen of a monitor from a distance by waving your hand in front of the screen. As ambitious as the goal may seem, its reach is limited to the confines of the digital mouse. Using a hand recognition algorithm, today's virtual mouse control technology enables us to do common mouse actions such as pointing, clicking, dragging, and more. There is no longer any purpose for the hand recognition software. They employed a static hand recognition system, which identifies hand forms without learning anything about the user and instead assigns a predefined action to each. It's difficult to understand what to do because of the limited number of options available in this system.

PROPOSED SYSTEM

Despite the fact that the present system provides a number of shortcuts for the finger and pointer motion for laptops, our solution

allows us to utilise the laptop or camera and, by detecting the hand movements, operate the mouse and conduct fundamental operations such as directing the mouse pointer, selecting and deselecting with the left click, The first "free" computer hand identification system has been created. Simple algorithms are used to detect the hand, follow its motions, and respond appropriately. However, much of our attention has been paid to the usage of a mouse for pointing and clicking, as well as a hand motion and action for moving information across devices. Python's ease of use, platform independence, flexibility, and portability make it an ideal choice for developing the system we've been working on. In the context of developing a system that recognise the user's mouse and hands virtually, this is helpful information. The system may be enhanced further by defining the motion of the hands required to complete a certain task. The range of possible modifications to the hand gesture set afforded by such motions is, in principle, infinite; the only restriction is the extent to which your imagination will allow. In order to control the pointer using simple hand and gesture motions, this software-only alternative to a traditional mouse Input from cameras and visual analysis of hand movements help achieve this goal.

System architecture

Here we shall detail the methods and processes that went into creating the vision-based CC system. The algorithm for the whole system is shown in the following. It is possible to reduce the effect of changes in lighting on the picture by converting it to chrominance colour space. The HSV colour space was chosen as the most optimal for skin detection. Then, face and other flesh-toned background objects were removed using background subtraction. A Image enhancement was used in order to improve semantic segmentation and smooth the picture.

7.1 CAMERA SETUP

The webcam of the linked desktop computer controls the runtime actions. We must establish an Image Capture element in order to detect video. Video filename or device index may be supplied as its argument. Only a number to recognise which webcam, the device index. We transfer it as "0" because we can use webcam. . Then, you can extract still images from the video. At the end, though, you must remember to free the captive..

7.2 PRESENT THE FRAME

It uses an infinite loop to enable the web camera to every time, grabs the frames, and is active during the entire program's duration. We record the live stream. frame by frame, in a stream. After that, we examine each captured frame in HSV colour from RGB (the default) colour space. More than 150 colour-space conversions exist. ways that OpenCV offers. But we'll focus on only two. Either of the following: BGR to Gray and BGR to HSV.

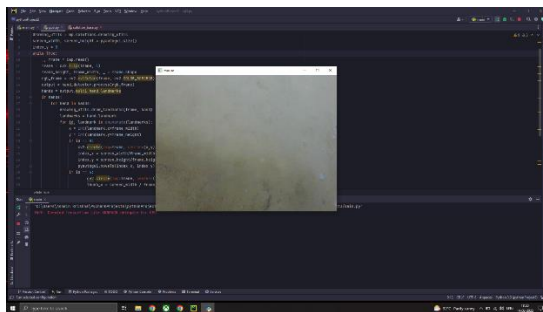


Figure.5 Display the frame

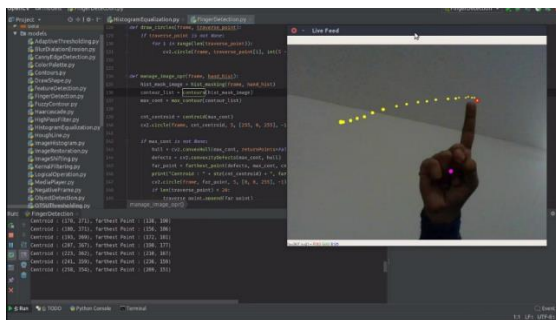


Figure6 Mouse movement

7.5 CLICKING

Applying the close gesture is the next stage. The action is carried out by clicking and

dragging the object. It is comparable to the open gesture, but since there is only one object

present, we simply need to determine its centre. And that will be put in the spot where our mouse pointer will be set console **Figure.7.** We shall employ a mouse press operation in place of a mouse release operation.

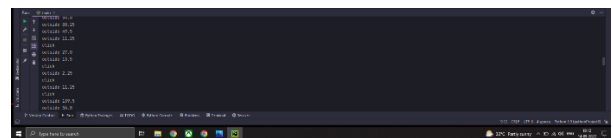


Figure7 Clicking console

7.6 DRAG

A variable called "pinch flag" is introduced in order to accomplish the dragging. If it was previously clicked, it will be set to 1. Therefore, once we locate the open gesture, we click and then verify that the pinch flag is set to 1. The drag action is carried out if it is set to one; else, the mouse move operation is carried out.

7.7 Frame DND

My File Drop Target is the first class that is created. We have one overridden method inside of that, On Drop Files. The file paths that are dropped along with the mouse's x/y position are both accepted by this approach.

RESULT

We made an effort to concentrate on supporting those who are unable to control their limbs and also improvise the movements of interaction between humans and machines. Our goal aimed to produce this technology in the most affordable manner feasible. and also, to construct it under a uniform operating system. The suggested system operates mouse functionalities as shown in the table **TABLE.1**. Perform mouse operations by detecting red with the pointer. such as dragging, cursor movement, left click between two connected systems.

EVENTS	No. of input samples	No. of Recognized samples	Recognized rate
Move to cursor up	5	4	0.8
Move to cursor down	5	3	0.6
Move the cursor left	5	3	0.6
Move the cursor right	5	4	0.8
Drag To	5	3	0.6
Move To	5	4	0.8
Click	5	3	0.6
Increased speed of cursor	5	3	0.6

TABLE.1 Table of Recognized rate of input samples

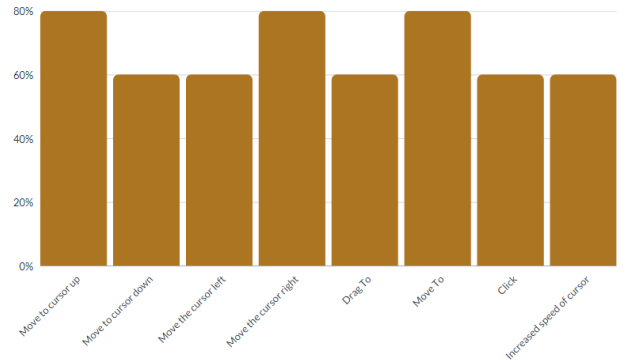


Figure.11 Bar Graph representing Recognized rates of input samples

CONCLUSION

The most effective human-machine interface is provided through gesture recognition. The development of alternative human computer interaction modes depends on gesture recognition. It makes it easier for people to interact more naturally with machines. Many different applications, including robot control and sign language recognition for the deaf and dumb, can use gesture recognition.

FUTURE ENHANCEMENT

To make the application more user-friendly, accurate, and versatile in various situations, further features and future effort are required.

Improved Performance and Accuracy: The hardware of the system, which includes the processing speed of the processor, the size of the available RAM, and the available functionalities of the webcam, heavily influences how quickly the device reacts. As a result, this system may also function better when used with compatible devices that have a webcam that performs better in particular lighting conditions. The method can also be employed in the future on Android devices, where hand gestures will take the place of the touchscreen notion.

A13 Journal

ORIGINALITY REPORT

7%

SIMILARITY INDEX

2%

INTERNET SOURCES

1%

PUBLICATIONS

4%

STUDENT PAPERS

PRIMARY SOURCES

1

Submitted to Visvesvaraya Technological University, Belagavi

Student Paper

4%

2

thecodacus.com

Internet Source

2%

3

Roshnee Matlani, Roshan Dadlani, Sharv Dumbre, Shruti Mishra, Abha Tewari. "Virtual Mouse using Hand Gestures", 2021 International Conference on Technological Advancements and Innovations (ICTAI), 2021

Publication

1%

Exclude quotes On

Exclude bibliography On

Exclude matches < 10 words



Ashwin Krishna (RA1911008020058) <ak7160@srmist.edu.in>

Regarding the status of your paper submitted for ICDEIS'22

ICDEIS Coordinator <srmicdeis@gmail.com>

Mon, 31 Oct, 23:47

To: <danesh.kn1@gmail.com>, <ak7160@srmist.edu.in>

Dear Author,

🌸 Greetings from SRM Institute of Science and Technology Ramapuram Campus! 🌸

Your paper id: 1234 has been accepted for publication in our conference. On or before **15th November 2022**, finish the registration process and kindly send your camera ready paper in IEEE - 2 columns format. After making the payment, send the payment proof along with the camera ready paper to this mail id with the subject mentioned as your paper id-camera ready paper.

Following is the registration detail:

REGISTRATION CHARGES

Rs.1750/- PER PAPER FOR ACADEMICIANS (Students and Faculty Members)

Rs.2250- PER PAPER FOR INDUSTRY PARTICIPANTS

PAYMENT DETAILS

ACCOUNT NAME : SRMIST

ACCOUNT NUMBER : 117109000032971

IFSC CODE : CIUBO000517

BANK NAME : CITY UNION BANK

BRANCH: RAMAPURAM

Coordinator
ICDEIS 2022