VIRTUAL MOUSE USING HAND GESTURE RECOGNITION USING DEEP LEARNING

A PROJECT REPORT

Submitted by

METHUN R	**720721115028**
KAAMESH S	**720721115021**
SANTHOSH KUMAR N	**720721115047**
AHAMED AKHIL M N	**720721115005**

in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

In

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

HINDUSTHAN COLLEGE OF ENGINEERING AND TECHNOLOGY

Approved by AICTE, New Delhi, Accredited with 'A++' Grade by NAAC

(An Autonomous Institution, Affiliated to Anna University, Chennai) Valley Campus, Pollachi Highway, Coimbatore – 641 032

OCTOBER 2024

Hindusthan College of Engineering And Technology
Approved by AICTE, New Delhi, Accredited with 'A++' Grade by NAAC
(An Autonomous Institution, Affiliated to Anna University, Chennai)
Valley Campus, Pollachi Highway, Coimbatore – 641 032

BONAFIDE CERTIFICATE

Certified that this project report **"VIRTUAL MOUSE USING HAND GESTURE RECOGNITION USING DEEP LEARNING"** is the bonafide work of **"METHUN R (720721115028), KAAMESH S (720721115021), SANTHOSH KUMAR N (720721115047), AHAMED AKHIL M N (720721115005)"**, who carried out the project work under my supervision.

SIGNATURE
Dr. R. VIDHYA, M.E., Ph.D.,
HEAD OF THE DEPARTMENT
Artificial Intelligence and
Machine Learning.
Hindusthan college of Engineering
and Technology, Coimbatore-32.

SIGNATURE
MS. M. DEVI, M.E.,
SUPERVISOR
Artificial Intelligence and
Machine Learning.
Hindusthan college of Engineering
and Technology, Coimbatore-32.

Submitted for the Autonomous Institution Project Viva-Voce conducted on

_____.

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION

We, hereby jointly declare that the project work entitled "**VIRTUAL MOUSE USING HAND GESTURE RECOGNITION USING DEEP LEARNING**" submitted to the Autonomous Institution Project Viva voice-October-2024 in partial fulfilment for the award of the degree of "BACHELOR OF TECHNOLOGY IN ARTIFICIAL INTELLIGENCE AND TECHNOLOGY", is the report of the original project work done by us under the guidance of Ms. M. Devi, M.E., Assistant Professor, Department of Artificial Intelligence and Machine Learning, Hindusthan College of Engineering and Technology, Coimbatore.

Name	**Signature**
Methun R	
Kaamesh S	
Santhosh Kumar N	
Ahamad Akhil M N	

I certify that the declaration made by the above candidates are true.

Project Guide,
Ms. M. Devi, M.E.,
Assistant Professor,
Department of AIML,
Hindusthan College of Engineering and Technology,
Coimbatore –32.

ABSTRACT

The development of technology for supporting learning systems at this time takes place very rapidly. Human Computer Interaction provides users with the ability to control presentations in a natural way by their body gestures. we propose a simple system that can be used to control presentation by hand gestures using computer vision.

This system primarily employs a web camera to record or capture photos and videos, and this application regulates the system's presentation based on the input. The primary purpose of the system is to change its presentation slides, I also had access to a pointer that allowed me draw on slides, in addition to that erase. To operate a computer's fundamental functions, such as presentation control, we may utilise hand gestures. People won't have to acquire the often burdensome machine-like abilities as a result.

These hand gesture systems offer a modern, inventive, and natural means of nonverbal communication. These systems are used widely in human computer interaction. This project's purpose is to discuss a presentation control system based on hand gesture detection and hand gesture recognition. A high resolution camera is used in this system to recognise the user's gestures as input.

The main objective of hand gesture recognition is to develop a system that can recognise human hand gestures and use that information to control a presentation. With real-time gesture recognition, a specific user can control a computer by making hand gestures in front of a system camera that is connected to a computer. With the aid of Open CV Python and Media Pipe, we are creating a hand gesture presentation control system in this project. Without using a keyboard or mouse, this system can be operated with hand gesture.

ACKNOWLEDGEMENT

We take this opportunity to express our wholehearted thanks and our profound respect to all those who guided and inspired us in the completion of this project work.

We extend our sincere thanks to the Founder and Chairman of Hindusthan Educational and Charitable Trust **Shri.T.S.R.Khannaiyann** and the Managing Trustee **Smt. Sarasuwathi Khannaiyann** and Executive Trustee & Secretary **Mrs. Priya Satish Prabhu** for providing essential infrastructure.

We would like to reveal our profound thanks to our respected Principal, **Dr.J.Jaya, M.Tech, Ph.D.,** who happens to be striving force in all endeavours.

We would like to express our gratitude to the Head of the Department **Dr.R.VIDHYA, M.E., Ph.D.,** for bringing out the project successfully and for strengthening the ray of hope.

We would like to express our sincere thanks and deep sense of gratitude to our Guide **MS.DEVI, M.E.,** Assistant Professor of Artificial Intelligence and Machine Learning, for his valuable guidance, suggestions and constant encouragement which paved way for the successful completion of the project work.

We express our immense pleasure and thankfulness to our Class Advisor and Project Coordinator, **MR.P.ARULSELVAM,M.E.,** Assistant Professor, and all other Faculty members of the Department of Artificial Intelligence and Machine Learning, technical staffs and friends who helped us for the successful completion of this project.

Table of Contents

CHAPTER 1: INTRODUCTION

In an increasingly digital world, the way we interact with technology is evolving rapidly. Traditional input devices, such as mice and keyboards, while effective, often impose limitations on user experience, particularly for individuals with physical disabilities or in situations where hands-free control is advantageous. To address these limitations, we propose the development of a **Virtual Mouse Using Hand Gestures**, a system that transforms hand movements into a sophisticated means of cursor control.

This project seeks to harness the power of computer vision and machine learning to create a gesture recognition interface that allows users to navigate their digital environments using simple hand gestures. By utilizing a camera—either embedded in a computer or as part of a mobile device—our system will interpret gestures such as swipes, pinches, and taps to perform actions traditionally executed with a mouse.

Objectives of the Project:

1. **Gesture Recognition**: The core of our project focuses on accurately recognizing and interpreting a variety of hand gestures in real time. By implementing advanced image processing techniques and machine learning algorithms, we aim to ensure that our system is responsive and precise, enabling users to control their devices with ease.

2. **User-Friendly Interface**: An intuitive user interface is essential for promoting user engagement and reducing the learning curve associated with new technology. We will design our system to be accessible and straightforward, ensuring that users can easily understand and master the gestures required for effective interaction.

3. **Accessibility**: One of our primary goals is to enhance accessibility for individuals with limited mobility. By providing an alternative input

method, we hope to empower users who face challenges with traditional devices, enabling them to interact more freely with computers and digital content.

The significance of this project extends beyond convenience; it represents a step toward a more inclusive digital future. Gesture-based control has the potential to revolutionize how users interact with technology, making computing more natural and intuitive. Our system can be applied across various domains, including gaming, virtual reality, and assistive technologies, enhancing user experiences in each.

By the conclusion of this project, we aim to deliver a robust prototype that effectively demonstrates the capabilities of gesture-based input as a viable alternative to conventional mouse control. We anticipate that this innovation will pave the way for further advancements in human-computer interaction, fostering an environment where technology is accessible to everyone. Through rigorous testing and user feedback, we aspire to refine our system, ensuring it meets the diverse needs of users in a wide range of contexts.

The virtual mouse system operates by continuously capturing video frames from the webcam and analysing them to recognize hand gestures. Once a gesture is detected, an algorithm interprets it as a specific mouse action and moves the cursor accordingly on the computer screen.

Additional functionalities, such as left and right-click actions, can be implemented using PyAutoGUI, a cross-platform GUI automation library. This approach allows for intuitive and hands-free computer interaction, with potential applications extending beyond mouse emulation to include sign language recognition, database interaction, and motion control systems.

CHAPTER 2: LITERATURE REVIEW

A Review on the project is This paper aims to cover the various prevailing methods of deafmute communication interpreter system. The two broad classification of the communication methodologies used by the deaf –mute people are - Wearable Communication Device and Online Learning System. Under the Wearable communication method, there are Glove based system, Keypad method and Handicom Touch-screen. All the above mentioned three subdivided methods make use of various sensors, accelerometer. a suitable micro-controller. a text to speech conversion module, a keypad and a touch-screen

A Review on the project is This paper aims to cover the various prevailing methods of deafmute communication interpreter system. The two broad classification of the communication methodologies used by the deaf –mute people are - Wearable Communication Device and Online Learning System. Under the Wearable communication method, there are Glove based system, Keypad method and Handicom Touch-screen. All the above mentioned three subdivided methods make use of various sensors, accelerometer. a suitable micro-controller. a text to speech conversion module, a keypad and a touch-screen

- **Title :** PYTHON BASED HAND GESTURE
- **Authors :** Ali A. Abed , Sarah A. Rahman
- **Year :** International Journal ,Volume 173 – No.4 , 2017
- **Algorithm:** Raspberry Pi.
- **Drawback :** It founds convexity defects, which is the deepest point of deviation on contour By this, it can find the number of fingers extended and then it can perform different functions according to the number of fiextended

• **Result :** Controlling the number of fingersi n front of camera leads to command avoid obstacles in the way of the robot. A recognition rate of about98% is reached.

- **Title:** HAND GESTURE MOVEMENT RECOGNITION SYSTEM
- **Authors:** Kundan Kumar Dubey , K. Narmatha
- **Year:** IRJCS/RS/Vol.06/Issue04/APCS10090 ,2019
- **Algorithm:** Convolution neural network.
- **Drawback:** It is a frequent problem in machine learning For the proposed and recognition task, the region of interest is relatively small, causing the deceptive behaviors in the cnn learning, such as making an attempt to infer the hand gesture from non-related image areas.
- **Result**: Network training is quite fast, requiring only ~50 minutes on hand gesture classification on a single image using the propose network requires about 2.96 milliseconds (ms) on GPU. Classification running times can be substantially improved through running the network on pics batches (requiring 0.73 ms per photo with a batch size of 256).

• **Title:** Hand Gesture Recognition for Human Computer Interaction
• **Authors:** Aashni Haria , Shristi Poddar
• **Year :** Procedia Computer science in 2017
• **Algorithm :** Morphology,Pavildis
• **Drawback :** We were able to create a robust gesture recognition system that did not utilize any markers, hence making it more users friendly and low cost. In this gesture recognition system, we have aimed to provide gestures, covering almost all aspects of HCI such as system functionalities, launching of applications and opening some popular websites.

• **Result :** In future we would like to improve the accuracy further and add more gestures to implement more functions. Finally, we target to extend our domain scenarios and apply ourtrackin g mechanism into a

variety of hardware including digital and mobile devices. We also aim to extend this mechanism to a range of users including disabled users

- **Title :** HAND GESTURE RECOGNITION:A LITERATURE REVIEW
- **Authors :** Rafiqul Zaman Khan , Noor Adnan Ibraheem
- **Year :** IJAIA, Vol.3, No.4, July 2012.
- **Algorithm :** neural networks
- **Drawback :** Orientation histogram method applied in [19] have some problems which are; similar gestures might have different orientation histograms and different gestures could have similar orientation histograms, besides that, the proposed method achieved well for any objects that dominate the image even if it is not the hand gesture.

- **Result :** In this paper various methods are discussed for gesture recognition, these methods include from Neural Network, HMM, fuzzy c-means clustering, besides using orientation histogram for features representation. For dynamic gestures HMM tools are perfect and have shown its efficiency especially for robot control.

- **Title :** GESTURE CONTROL TECHNOLOGY
- **Authors :** Stephen M
- **Year:** 2018
- **Algorithm :** Practical
- **Drawback :** As mentioned in the 2011 Horizon Report, gesture-based computing is one of the key trends in education technology, and we are soon to see the implementation of this technology as it develops further. In higher education, gesture control technology could not only improve the learning

experiences for the students, but also provide a new teaching method for lecturers

- **Result :** Gesture control technology has shown a great potential in education. Hui-Mei Hsu (2011) presented a complete analysis on the use of Kinect in education and concluded that it should really achieve the expected results, enhancing classroom interaction and participation, as well as improving the way that teachers can manipulate multimedia materials during the classes.

- **Title :** Hand Gesture Recognition
- **Authors :**Moh.Harris , Ali Suryaperdana Agoes
- **Year :** Atlantis press,, volume 207,2021
- **Algorithm :** 3rd-degree polynomial equation
- **Drawback :** The main objective of the research is to recognize hand gestures to display one of the menus that a user has chosen through a Kinect. We used 10 captured hand gestures which each hand gesture directly set one menu out
- **Result :** I. The measure of the performance on the model in machine learning used Confusion Matrix. In Python, we can use the library scikit-learn to develop a confusion matrix. Experiment datasets were obtained before we used them to predict the hand gestures. The Confusion Matrix was also used to observe an accuracy achieved for the model was made.

- **Title :** SMART PRESENTATION CONTROL BY HAND GESTURES
- **Authors :** Hajeera Khanum , Dr. Pramod H
- **Year :** Journal, Volume: 09 , 07 /July 2022
- **Algorithm :** polynomial

- **Drawback** : In order to generate a better result, we have implemented a Hand Gestures Recognition System. The webcam is turned on while the software is running, and the kind of gesture used to detect the shape of the hand and give us the desired output is static. This project uses the curve of the hand to regulate loudness. The system receives input, captures the item, detects it, and then recognises hand gestures

- **Result :** This project showcases a programme that enables hand gestures as a practical and simple method of software control. A gesture- based presentation controller doesn't need any special markers, and it can be used in real life on basic PCs with inexpensive cameras since it doesn't need particularly high quality cameras to recognise or record the hand movements.

EXISTING SYSTEM

The author has developed an ANN application used for classification and gesture recognition, Gesture Recognition Utilizing Accelerometer. The Wii remote, which rotates in the X, Y, and Z directions, is essentially employed in this system. The author has utilised two tiers to construct the system in order to reduce the cost and memory requirements. The user is verified for gesture recognition at the first level. Author's preferred approach for gesture recognition is accelerometer based. That system signals are analysed at the second level utilising automata to recognise gestures (Fuzzy). The Fast Fourier technique and k means are then used to normalise the data. The accuracy of recognition has now increased to 95%. Recognition of Hand Gestures Using Hidden Markov Models - The author of this work has developed a system that uses dynamic hand movements to detect the digits 0 through 9. In this work, the author employed two stages. Pre-processing is done in the first phase, while categorization is done in the second. There are essentially two categories of gestures The author has employed inexpensive cameras to keep costs down for the consumers. Robust Part-Based Hand Gesture Recognition Using Kinect Sensor. Although a kinect sensor's resolutions lower than that of other cameras, it is nevertheless capable of detecting and capturing large pictures and objects. Only the fingers, not the entire hand, are paired with FEMD to deal with the loud hand movements. This technology performs flawlessly and effectively in uncontrolled settings. The experimental result yields an accuracy of 93.2%. The key gesture and the link gestures are employed in continuous gestures for the goal of spotting. Discrete Hidden Markov Model (DHMM) is employed for classification in this work. The Baum-Welch algorithm is used to train this DHMM. HMM has an average recognition rate range of 93.84 to 97.34%.

CHAPTER 3: METHODOLOGY

The Methodology for Gesture Controlled Virtual Mouse incorporates the steps of Analysis, Planning, Content, Design, Programming, Testing, Marketing and Advertising

➢ **Analysis:** In This step, the project team would gather information about the requirement for the gesture controlled virtual mouse. This might involve researching existing products, interviewing potential users, and identifying technical requirements such as hardware and software specifications.

➢**Planning:** - Based on the analysis, the team would develop a plan for the project, including timelines, milestones, and resource allocation. When the project plans are documented, the project deliverables and requirements are defined, and the project schedule is created.

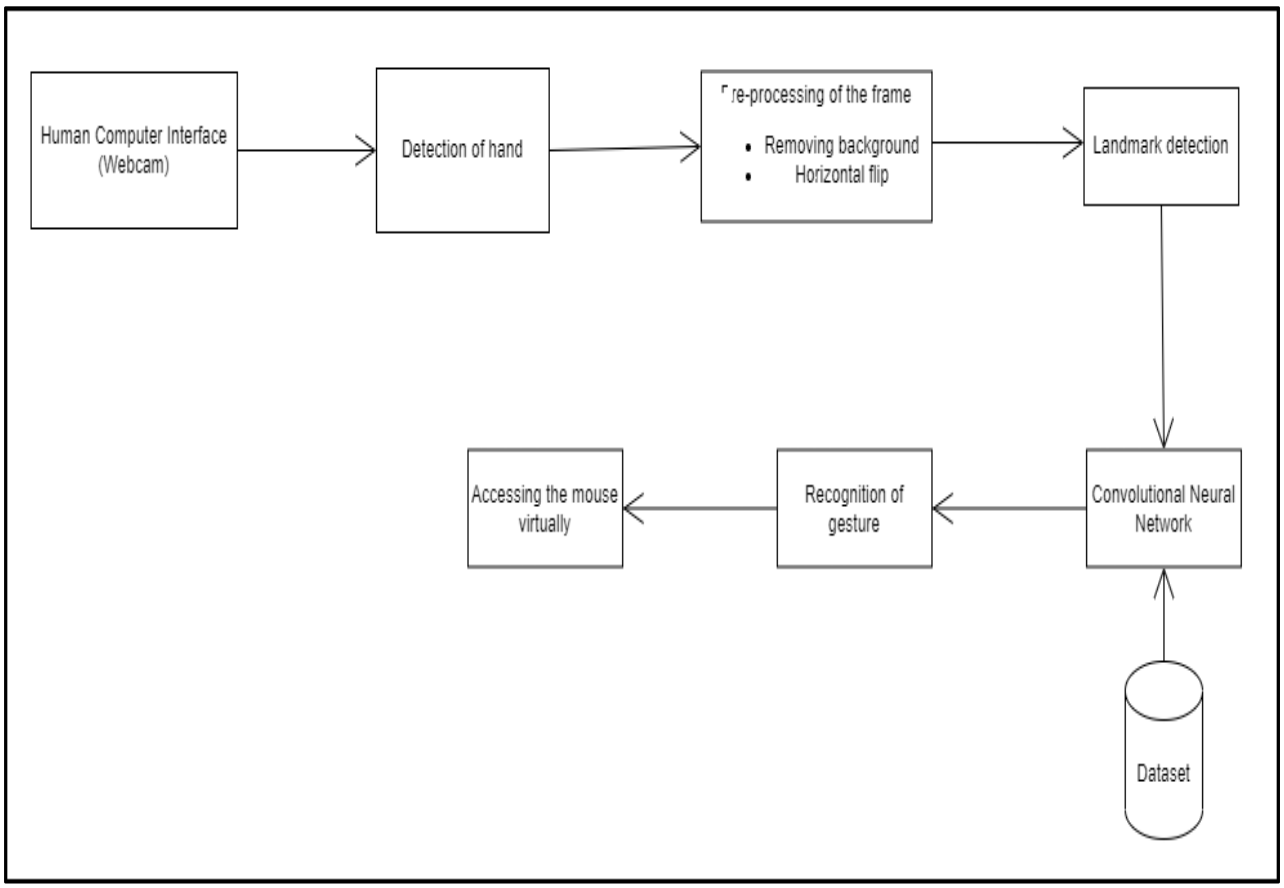
➢ **Content:** - This step involves creating the content that will be used in the product, such as graphics, user interface design, and documentation.

➢ **Design:** In this step, the team would create a detailed design for the product, including wireframes, prototypes, and user flows. It is an early phase of the project lifecycle where ideas, processes, resources, and deliverables are planned out. A project design comes before a project plan as it's a broad overview whereas a project plan includes more detailed information.

➢ **Programming:** - The actual development of the product would take place in this step, with the team writing the code to implement the design. Programming allows the project team to take the information acquired during Project Initiation and documented in the Owner's Project Requirements to the next level of detail.

➢**Testing:** - Once the product is developed, it would need to be thoroughly tested to ensure that it works as expected and meets the requirements that were identified in the analysis phase. Overall, this methodology is a standard approach to software development and product promotion. However, the specific details of each step may vary depending on the nature of the product and the needs of the project team.

ARCHITECTURE DIAGRAM



CHAPTER 3.1: CAPTURING LIVE FEED

The live feed is captured from the webcam attached to the system. It constantly extracts image from the live video. The extracted image is then flipped horizontally. Once hand is detected, it extracts image from the live feed. The size of the input is 1280 x 720. Once the image is captured, an empty array of the size of the shape of image is created and it is filled with the numeric equivalent of white i.e., 255.

The captured image is then converted from BGR to RGB and tested to see whether hand is detected in the frame or not. If hand is detected, then the subsequent elements of the array are marked for the 21 co-ordinates using Landmark algorithm from **Media Pipe** framework.

Lines are also marked joining these co-ordinates. The live feed which is constantly displayed on the screen for reference is also marked with the same co-ordinates and lines joining the same. Only one hand is detected at a time to prevent confusion and inability to control the mouse. The co-ordinates are normalized and also stored separately for further use.

The array is then resized and converted to image. It is then passed to the **CNN** for identifying the gesture. The result from CNN is displayed on the screen as well for the user. The 10th co ordinate i.e., the co-ordinate at the base of the middle finger is used to move mouse pointer on the screen accordingly. It is the reference point which determines the position of the mouse pointer.

CHAPTER 3.2: DATASET

The dataset used in training the CNN model which is used for virtual mouse in this paper is self-made. It consists of total 9000 images. As 6 different operations are included in this mouse, 6 different gestures are required. So, for each class there are 1500 images.

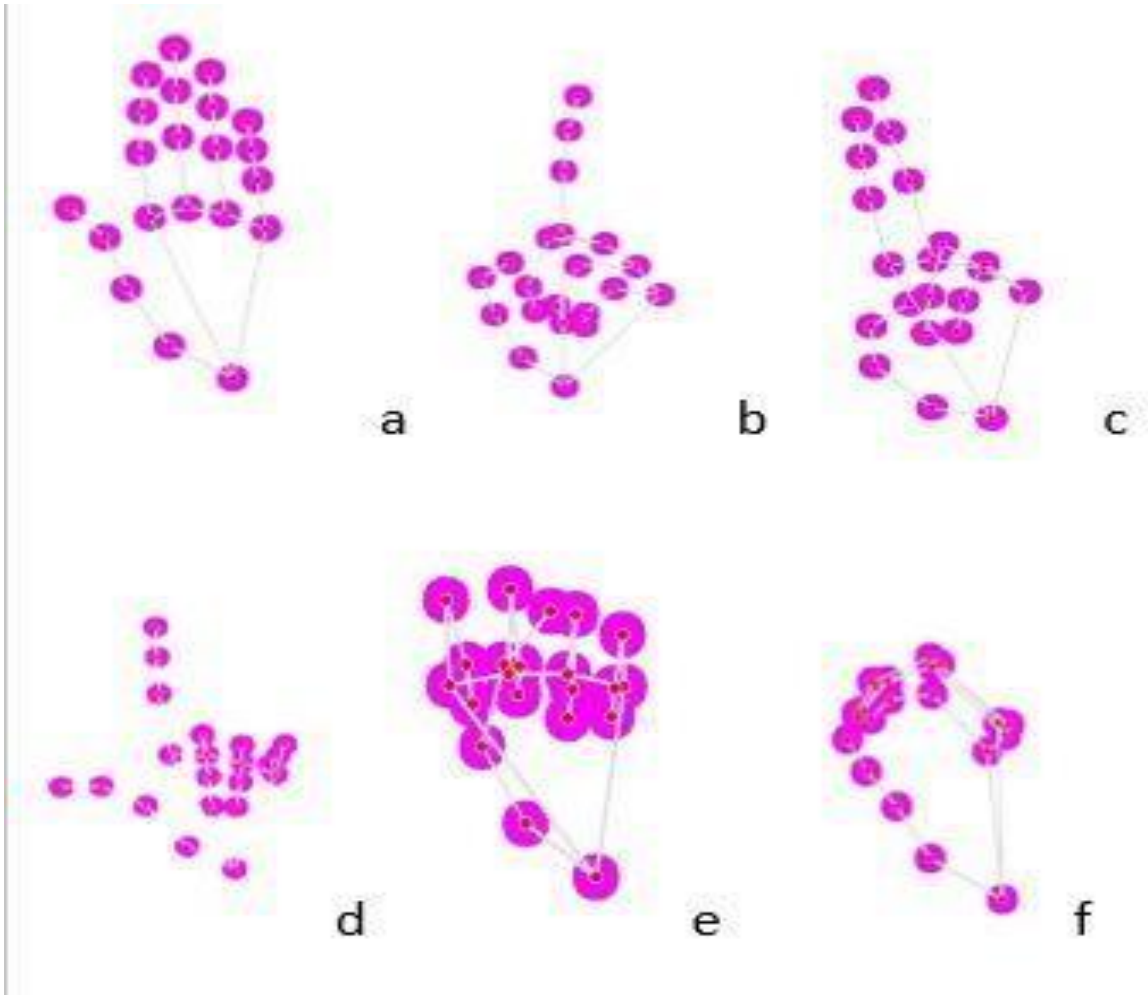
The dataset is divided in the ratio of 80-20, as a result, 1200 images for each gesture are used for training and 300 images are used for testing for each gesture. These images are pre processed similar to the images that will be passed to CNN while using the system. The images have a white background and the 21 co-ordinates on the hand are marked with circles and these are joined by lines.

While collecting dataset, care has been taken so that variations of the gestures are included to increase the scope. This means the gestures in some images are titled, have different positions so that the diversity is maintained.

As the dataset contains images of right hand, at this point, the virtual mouse works can be controlled accurately using right hand only. The images below are example of each of the gestures.

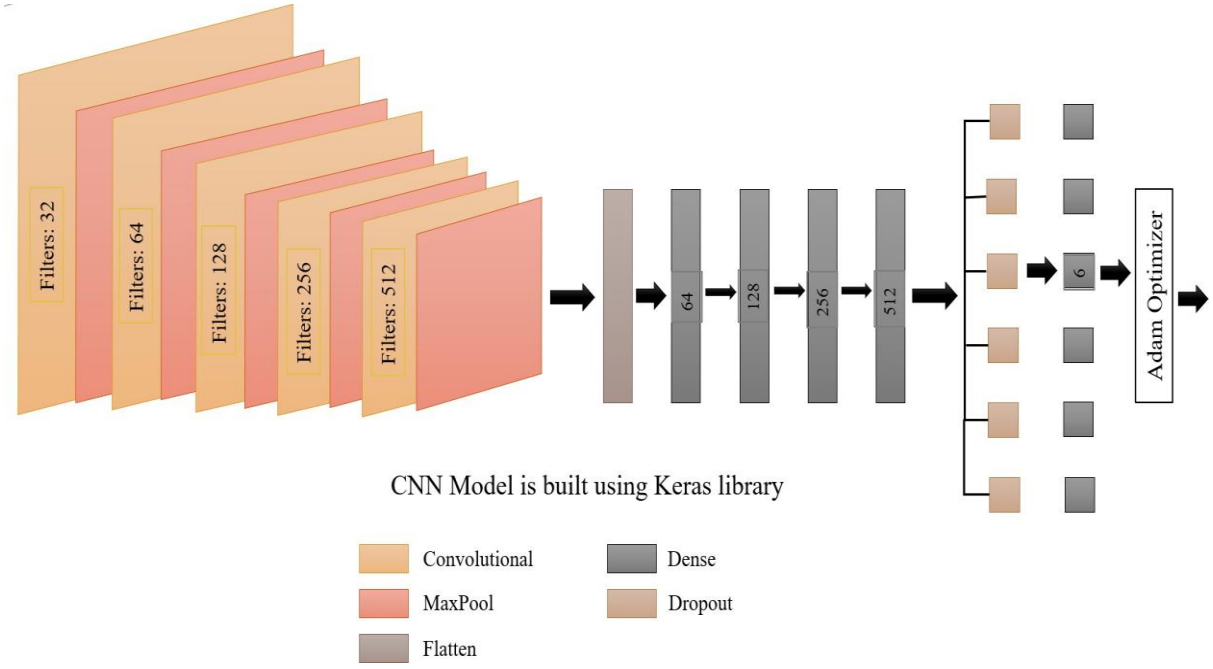
Dataset a.navigation b.left click c.right click d.scroll up e.scroll down

f.screenshot.



CHAPTER 3.3: CNN MODEL

The CNN model which is trained is of sequential type. That means the model is built layer by layer. Training and testing batches are made and data augmentation is done with the help of Image DataGenerator. The batch size is kept as 32 and it is shuffled. While pre-processing, the images are converted into arrays. The images are resized to 64 x64. The class mode is set to categorical and not binary as the images will classified into different classes.



The model consists of 4 Conv2D layers of filters 32, 64, 128, 256 respectively. The kernel sizes specified are (3, 3) and strides are 2. The pool size is set to (2,2). As the first layer takes the input, input size is also to be specified. As previously seen that the images are of size 64 x 64, input size is set to (64, 64 ,3). The activation function used for each of these layers is ReLU i.e., Rectified Linear Activation. Dense layers are also added. But before the dense layers, flatten layer is added to connect the convolutional layers and dense layers. After flatten layer, 6 dense layers are added. The units in the dense layer are 64, 128, 256, 512 respectively. All these layers use ReLU activation function.

After these layers, a dropout layer is added. Dropout layer prevents the model from overfitting. Dropout randomly sets the outgoing edges of hidden units to 0 at each update of the training phase. Dropout layer is passed a value of 0.2 i.e., every hidden unit i.e., neuron is set to 0 with a probability of 0.2. After dropout layer another dense layer is added.

As this is the last layer, this can also be said as output layer. Since we have 6 different gestures to classify, thus there is possibility is 6 different outputs. As a result, the number of units in this layer is set to 6. The activation function used for this layer is SoftMax. It switches the result over completely to 1 so it very well may be deciphered as probabilities.

The model will then, at that point, make its forecast in light of which choice has the highest probability. The model is compiled where Adam optimizer is used with a learning rate of 0.001 and categorical_crossentropy as loss function with metrics set to accuracy. The Reduce learning rate function monitors val_loss with a factor of 0.2. The early stopping function monitors val_accuracy with patience set to 2. The number of epochs is set to 10. The accuracy of the model is obtained to be 96.875% for training and 87.5% for testing.

CHAPTER 3.4 : Accessing mouse virtually

If hand is detected, then it is passed to CNN to identify gesture. If result from CNN is left, then left click is performed, if it is right then right click is performed. If the result is scroll_up then vertical scroll up is done for 15 clicks and is it is scroll_down the vertical scroll down is performed for 15 clicks. If the result is screenshot, then a screenshot is taken of current screen and saved with a .png extension.

If it is palm, then cursor is moved to the normalized position of the base of the middle finger on the screen. A sleep time of about 0.5 seconds is introduced after left and right clicks so that the clicks are not overlapped and the user gets sufficient time to change the gesture. All these operations are done using the PyAutoGUI library of python.

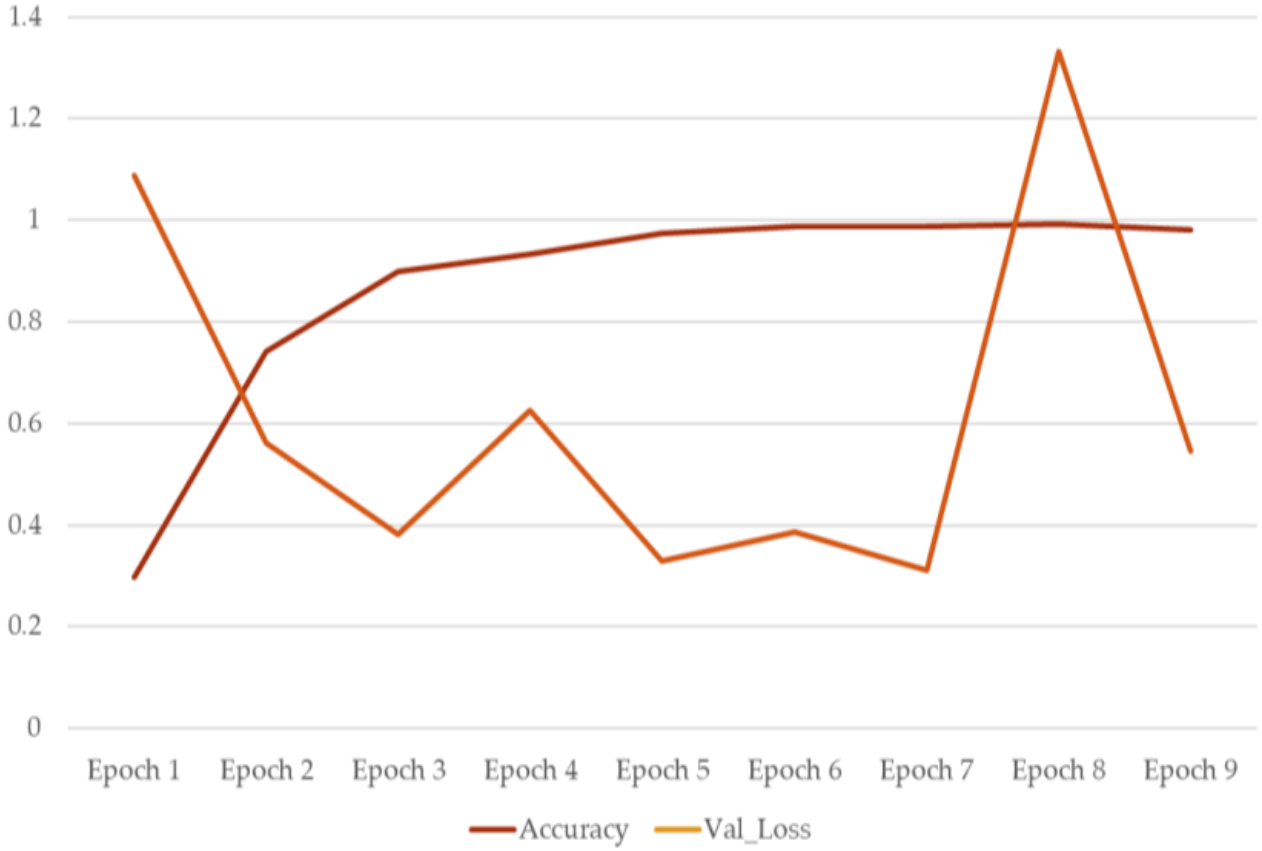
It is quite easy to use and performs all the operations accurately. There are many other operations which can be added like horizontal scroll, but the working of PyAutoGUI function for horizontal scroll function i.e. hscroll() is limited. It is supported only by OS X and Linux platforms. Thus, this limits the number of operations that can be introduced.

CHAPTER 3.5 RESULTS

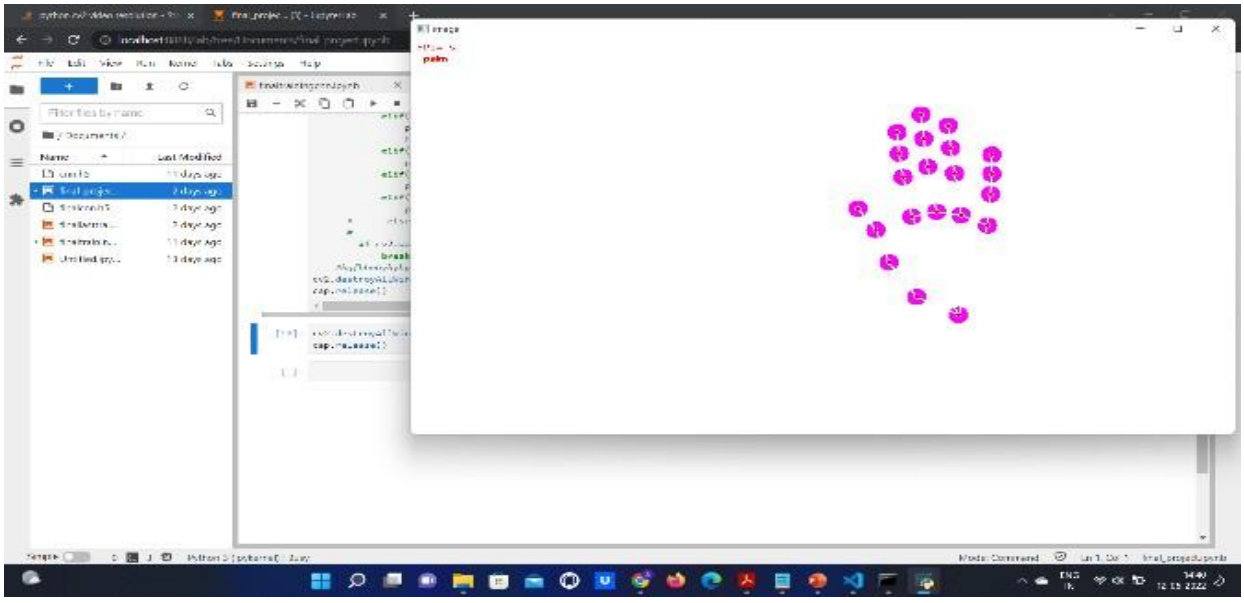
This project has successfully implemented virtual mouse using convolutional neural network based on hand gesture recognition After training, the CNN model has obtained the training accuracy of 96.875% and testing accuracy of 87.5%.

TABLE

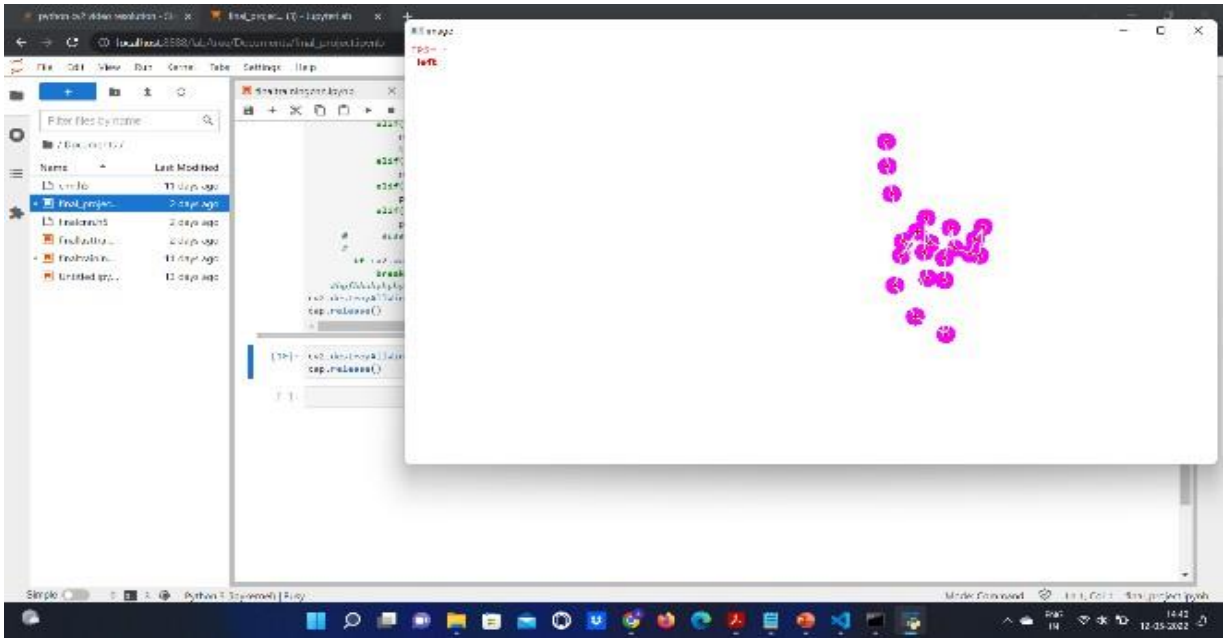
Result	Training	Testing
Accuracy	96.875%	87.5%
Val. Loss	0.2690	0.3697



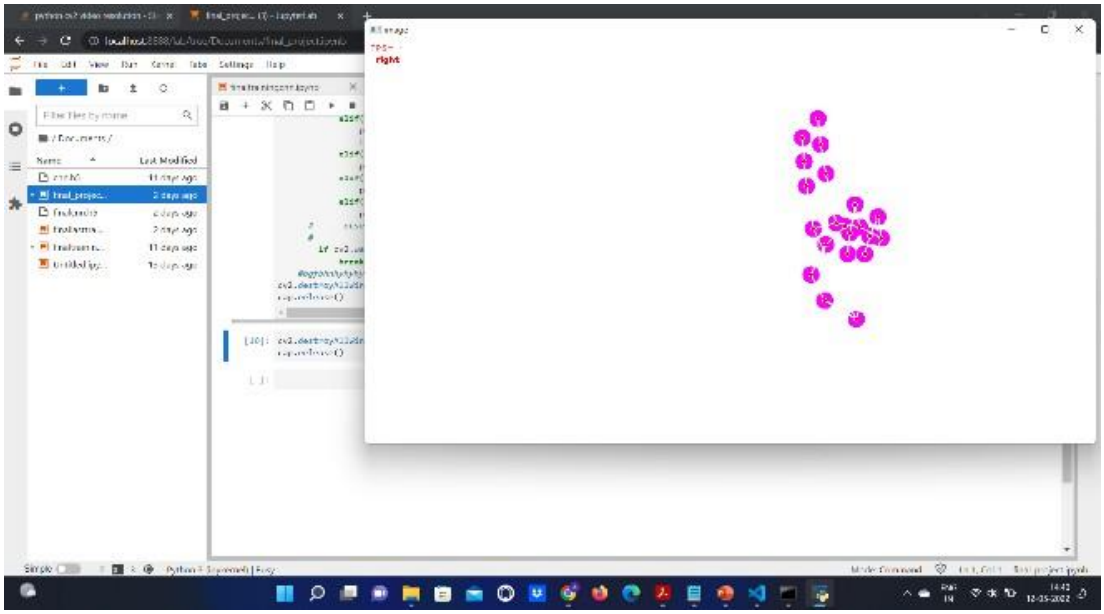
MODEL OUTPUT



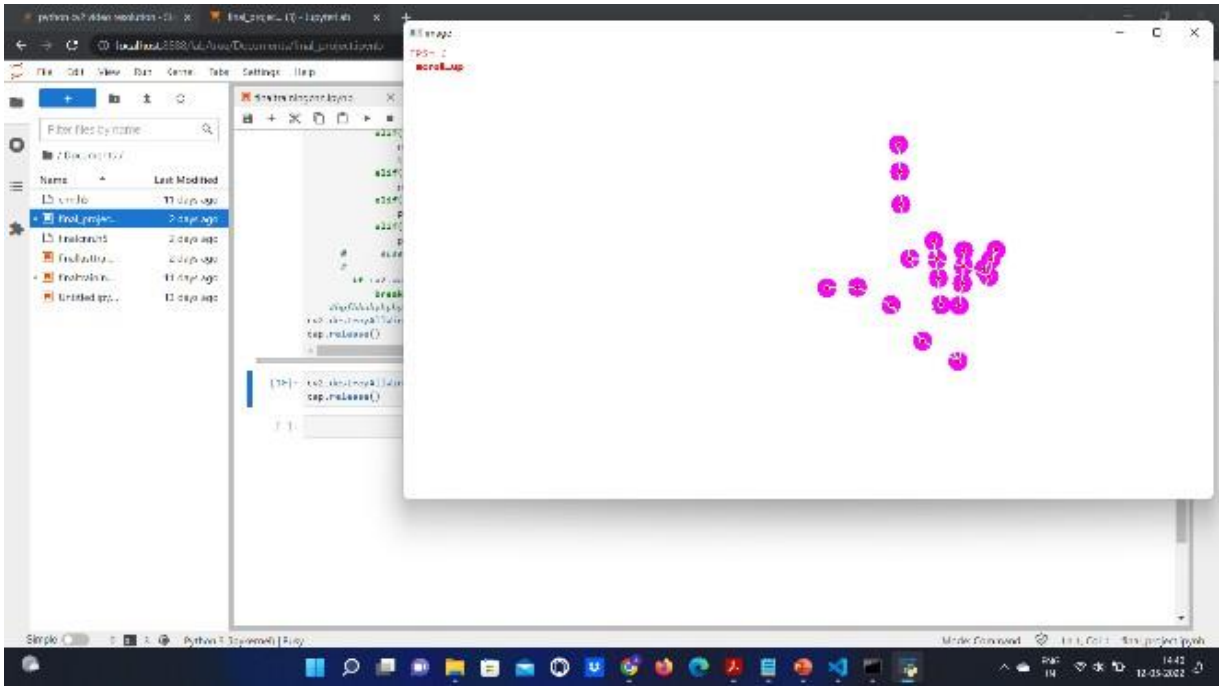
Recognition of Navigation Gesture



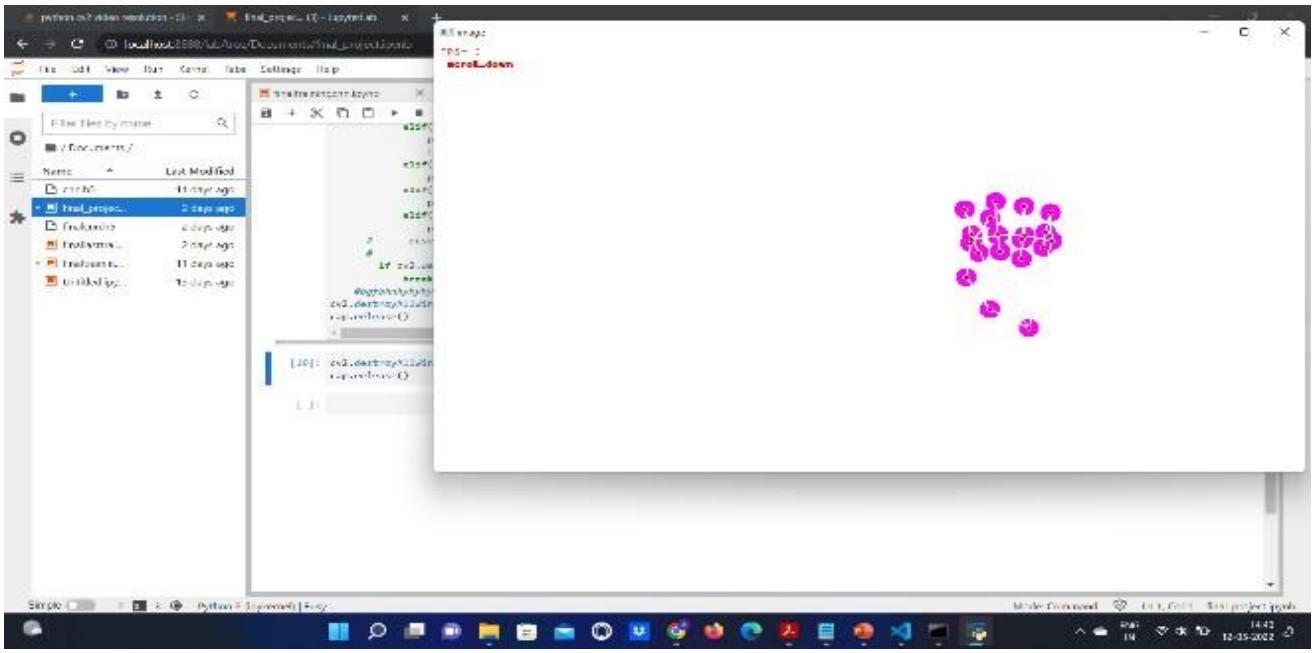
Recognition of Left Click Gesture



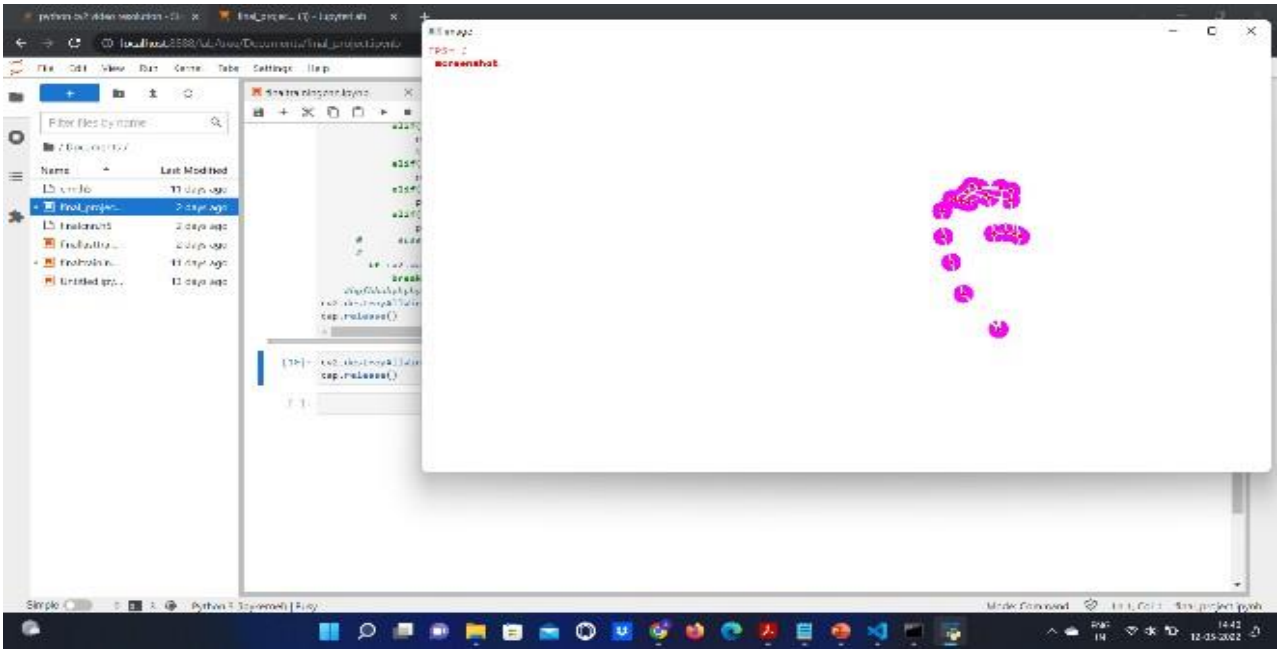
Recognition of Right Click Gesture



Recognition of Scroll-Up Gesture



Recognition of Scroll Down Gesture



Recognition of Screenshot Gesture

4. COMPARISON WITH PREVIOUS PAPERS

Characteristics	Previous papers	This Paper
Pre-processing	Background subtraction method of OpenCV is used, weighted background estimation is used.	Array form of image is created where background is taken as white and 21 co-ordinates are marked with circles and joined with lines.
Algorithm used	Color identification and gesture recognition. Classified using area ratio calculated by convex hull.	Gesture Recognition. Classified using CNN.
Hand Gesture	Gesture is based on the number of fingers.	Natural gestures are used.
Accuracy	90%	87.5%
Additional hardware	Webcam Colored caps	Webcam

The accuracy of the model is obtained to be 96.875% for training and 87.5% for testing.

Thus, the total parameters of the model are 630,022. The accuracy increases consistently for the first 4epochs and then remains steady.

4.1 APPLICATIONS/ FUTURE WORKS

From future perspective, we can incorporate more mouse operations thus widening the scope of the project. By using different algorithms, we can also strive to increase the accuracy of the results. Apart from the virtual mouse system mentioned in the paper, this model can be combined with other technologies to provide support to differently-abled people.

For e.g., sign language and communication. It can also be used in the gaming field. Further eye movements can be used to control the mouse which will be much more beneficial.

4.2 Scope of the Project

In computer science and language technology, gesture recognition is an important topic which interpret human gesture through computer vision algorithms. There are various bodily motion which can originate gesture but the common form of gesture origination comes from the face and hands. The entire procedure of tracking gesture to their representation and converting them to some purposeful command is known as gesture recognition1.

Various technologies has been used for the design and implementation of such kind of devices, but contact based and vision based technologies are two main types of technologies used for robust, accurate and reliable hand gesture recognition systems. Contact based devices like accelerometers 7, multi-touch screen, data glove9 etc. based on physical interaction of user who will be required to learn their usages.

Whereas vision based devices like cameras has to deal with the prominent variety of gestures. Gesture recognition involves to handle degrees of

freedom4, 10 (DOF), variable 2D appearances, different silhouette scales (i.e. spatial resolution) and temporal dimension (i.e. gesture speed variability). Vision based gesture recognition further classified into two main categories, which are 3D model based methods and appearance based methods1.

3D based hand models4 describes the hand shapes and are the main choice of hand. gesture modelling in which volumetric analysis is done. In appearance based models4, the appearance of the arm and hand movements are directly linked from visual images to specific gestures. A large number of models belong to this group. We have followed one of these models i.e. silhouette geometry based models to recognize the gesture in our project. A fast, simple and effective gesture recognition algorithm for robot application has been presented which automatically recognizes a limited set of gestures.

However, the segmentation process should be robust and required to be deal with temporal tracking, occlusion and 3D modelling of hand. The author of7 has used multi-stream Hidden Markov Models (HMMs) consisting of EMG sensors and 3D accelerometer (ACC) to provide user friendly environment for HCI. However, there are some problems or limitations in ACC-based techniques and EMG measurement. In11, a method has been proposed which firstly store the human hand gesture into the disk, convert them into binary image by extracting frame from each video one by one and then creates 3D Euclidian space for binary image, for recognizing vision-based hand gesture.

They have used back propagation algorithm and supervised feed-forward neural network based training for classification However it is suitable for only simple kind of gesture against the simple background. In12, a method for detecting finger from the detected hand, can be used as a non- 10 contact mouse, has been proposed. They have used skin color technique for segmentation and contour as the feature to locate the fingertip in hand. The authors in13 have used bag-of-features and multiclass SVM to detect and

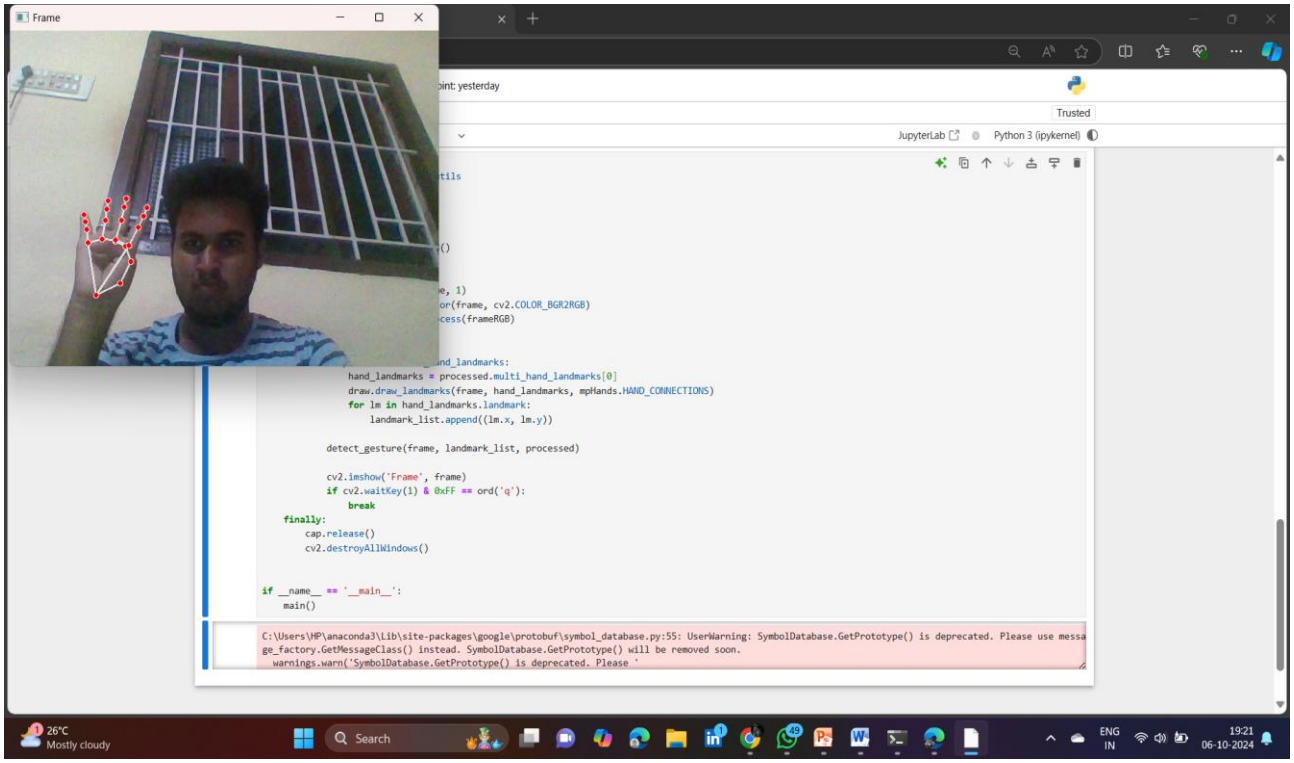
track bare hand, and to control an application using command generated by a grammar in a complex background, via skin detection and contour comparison algorithm.

They have also used K-means clustering algorithm and scale invariance feature transform (SIFT) to extract the main features from the trained images. However, the segmentation and localization method is unclear for the system and there is no rigorous geometric information of the object components. In14, the author has used Lucas KanadePyramidical Optical Flow algorithm to detect moving hand and K- means algorithm to find center of moving hand. Here Principal Component Analysis (PCA) was used to extract features and then the extracted features were matched using K-nearest neighborh od.

However, PCA made whole system slower and required more memory. In15, a comparative analysis of different segmentation techniques and how to select an appropriate segmentation method for the system have been presented. It has also described Gaussian Model Classifier along with some other classification techniques.

4.3 VISUALIZATION

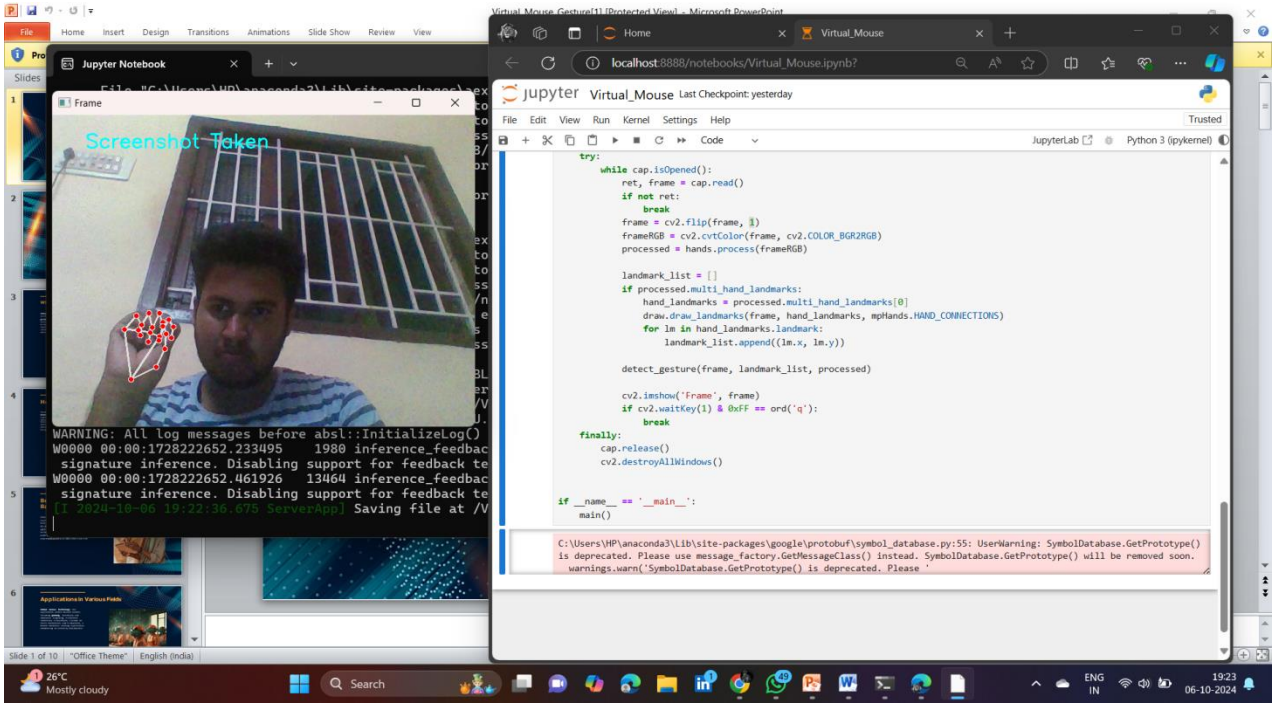
Accomplishes accurate crucial point clustering of 21 main points with only a 3D touch coordinates that is done within the identified hand areas and immediately generates the coordinates predictor that is a representation within Media Pipe of hand landmarks.



The pink circle in the image corresponds to the index finger, serving as a marker for mouse movement. To determine the optimal location for mouse control, we employ a method to pinpoint the canter of two detected red objects.

This involves calculating the average of the maximum and minimum points of their respective bounding boxes. By obtaining two coordinates representing the centres of these pink objects, we then determine the midpoint between them.

This midpoint, denoted by the pink point, serves as the reference for controlling the mouse cursor, specifically through the movement of the index finger.



Incorporating five fingers for taking a screenshot involves a straightforward process: when the system detects the spread of all five fingers on the screen simultaneously, it triggers the screenshot function.

This intuitive gesture provides users with a convenient and effortless way to capture their screen contents with just a simple hand movement.

Modules:

- OpenCV
- Media Pipe
- PyAutoGUI
- Math

OpenCV:

- OpenCV, a renowned computer vision library, offers a plethora of techniques for image processing and object detection. Its Python package enables the development of real-time computer vision applications, empowering users to analyze data from images and videos, including tasks like face and object recognition.

This open-source software, widely known for its versatility, serves as a robust tool for both computer vision and machine learning applications. By providing a standardized infrastructure, OpenCV accelerates the integration of artificial intelligence into various products. Notably, its Apache 2 license facilitates easy adoption and modification of the codebase, making it a preferred choice for businesses seeking customizable solutions in the realm of computer vision.

MediaPipe:

- MediaPipe, an open-source framework developed by Google, serves as a versatile tool within machine learning pipelines. Leveraging time series data, MediaPipe facilitates cross-platform programming, offering support for various audio and video formats due to its multimodal architecture.

Developers utilize the MediaPipe framework to design and analyze systems through graph-based approaches, tailoring them to application-specific requirements.

At the core of MediaPipe lies the pipeline configuration, orchestrating actions within the system. This flexible pipeline enables scalability across desktop and mobile platforms. The framework comprises three key

components: performance evaluation, sensor data access, and a collection of reusable modules known as calculators.

A MediaPipe pipeline is essentially a graph composed of interconnected calculators, forming pathways through which data packets flow. Developers have the freedom to customize the pipeline by adding, removing, or redefining calculators to suit their application needs. This flexibility empowers developers to create bespoke solutions while harnessing the power of MediaPipe framework.

PyAutoGUI:

- PyAutoGUI stands as a cross-platform Python software compatible with Windows, MacOS X, and Linux, facilitating the emulation of keyboard inputs, mouse movements, and clicks.

This Python package caters to GUI automation needs across various operating systems, empowering users to automate tasks such as clicking, dragging, scrolling, and precise cursor positioning.

With PyAutoGUI, users can automate keyboard and mouse controls seamlessly, abstracting away the intricate technicalities involved in programmatically manipulating these input devices across different operating systems.

Rather than grappling with complex and cryptic methods for controlling the mouse and keyboard individually in Windows, macOS, and Linux, PyAutoGUI simplifies the process through a user friendly API, streamlining GUI automation tasks for developers and users alike.

Math:

- Python's math module stands as a cornerstone for handling mathematical operations seamlessly within Python programs. This module, integral to the default Python installation, provides a comprehensive range of mathematical functions, simplifying complex calculations.

Behind the scenes, many of these functions serve as thin wrappers for their C platform equivalents, ensuring efficiency and adherence to CPython standards.

A notable feature of the Python math module lies in its provision of predefined constants, offering users a repertoire of commonly used mathematical values. Leveraging these constants not only saves time by obviating the need for manual inclusion in code but also ensures consistency across the entire program.

By leveraging the Python math module, developers can execute a myriad of mathematical operations effortlessly, tapping into a robust library of functions and constants to enhance the efficiency and accuracy of their programs.

The system we have proposed and designed for vision-based hand gesture recognition system contained various stages which we have explained through an algorithm. The working flowchart of gesture recognition system has also shown above.

5. SOURCE CODE

```
import cv2
import mediapipe as mp
import pyautogui
import random
import util
from pynput.mouse import Button, Controller
mouse = Controller()



screen_width, screen_height = pyautogui.size()


mpHands = mp.solutions.hands
hands = mpHands.Hands(
    static_image_mode=False,
    model_complexity=1,
    min_detection_confidence=0.7,
    min_tracking_confidence=0.7,
    max_num_hands=1
)
```

```
def find_finger_tip(processed):
    if processed.multi_hand_landmarks:
        hand_landmarks = processed.multi_hand_landmarks[0]  # Assuming only one hand is detected
        index_finger_tip = hand_landmarks.landmark[mpHands.HandLandmark.INDEX_FINGER_TIP]
        return index_finger_tip
    return None, None


def move_mouse(index_finger_tip):
    if index_finger_tip is not None:
        x = int(index_finger_tip.x * screen_width)
        y = int(index_finger_tip.y / 2 * screen_height)
        pyautogui.moveTo(x, y)


def is_left_click(landmark_list, thumb_index_dist):
    return (
        util.get_angle(landmark_list[5], landmark_list[6], landmark_list[8]) < 50 and
        util.get_angle(landmark_list[9], landmark_list[10], landmark_list[12]) > 90 and
```

```
        thumb_index_dist > 50
    )


def is_right_click(landmark_list, thumb_index_dist):
    return (
        util.get_angle(landmark_list[9], landmark_list[10], landmark_list[12]) < 50 and
        util.get_angle(landmark_list[5], landmark_list[6], landmark_list[8]) > 90 and
        thumb_index_dist > 50
    )


def is_double_click(landmark_list, thumb_index_dist):
    return (
        util.get_angle(landmark_list[5], landmark_list[6], landmark_list[8]) < 50 and
        util.get_angle(landmark_list[9], landmark_list[10], landmark_list[12]) < 50 and
        thumb_index_dist > 50
    )
```

```
def is_screenshot(landmark_list, thumb_index_dist):
    return (
        util.get_angle(landmark_list[5], landmark_list[6], landmark_list[8]) < 50
    and
        util.get_angle(landmark_list[9], landmark_list[10], landmark_list[12]) <
    50 and
        thumb_index_dist < 50
    )



def detect_gesture(frame, landmark_list, processed):
    if len(landmark_list) >= 21:

        index_finger_tip = find_finger_tip(processed)
        thumb_index_dist = util.get_distance([landmark_list[4], landmark_list[5]])

        if util.get_distance([landmark_list[4], landmark_list[5]]) < 50 and
    util.get_angle(landmark_list[5], landmark_list[6], landmark_list[8]) > 90:
            move_mouse(index_finger_tip)
        elif is_left_click(landmark_list, thumb_index_dist):
            mouse.press(Button.left)
            mouse.release(Button.left)
```

```
            cv2.putText(frame, "Left Click", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
        elif is_right_click(landmark_list, thumb_index_dist):
            mouse.press(Button.right)
            mouse.release(Button.right)
            cv2.putText(frame, "Right Click", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
        elif is_double_click(landmark_list, thumb_index_dist):
            pyautogui.doubleClick()
            cv2.putText(frame, "Double Click", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 0), 2)
        elif is_screenshot(landmark_list,thumb_index_dist ):
            im1 = pyautogui.screenshot()
            label = random.randint(1, 1000)
            im1.save(f'my_screenshot_{label}.png')
            cv2.putText(frame, "Screenshot Taken", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 0), 2)


def main():
    draw = mp.solutions.drawing_utils
    cap = cv2.VideoCapture(0)
```

```
try:
    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break
        frame = cv2.flip(frame, 1)
        frameRGB = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        processed = hands.process(frameRGB)

        landmark_list = []
        if processed.multi_hand_landmarks:
            hand_landmarks = processed.multi_hand_landmarks[0]  # Assuming only one hand is detected
            draw.draw_landmarks(frame, hand_landmarks, mpHands.HAND_CONNECTIONS)
            for lm in hand_landmarks.landmark:
                landmark_list.append((lm.x, lm.y))

        detect_gesture(frame, landmark_list, processed)

        cv2.imshow('Frame', frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
```

```
    finally:
        cap.release()
        cv2.destroyAllWindows()


if _name_ == '_main_':
    main()


# This is the code execute the output successfully…….
```

UTIL.PY

```
import numpy as np


def get_angle(a, b, c):
    radians = np.arctan2(c[1] - b[1], c[0] - b[0]) - np.arctan2(a[1] - b[1], a[0] - b[0])
    angle = np.abs(np.degrees(radians))
    return angle
```

```
def get_distance(landmark_ist):
    if len(landmark_ist) < 2:
        return
    (x1, y1), (x2, y2) = landmark_ist[0], landmark_ist[1]
    L = np.hypot(x2 - x1, y2 - y1)
    return np.interp(L, [0, 1], [0, 1000])
```

This is the code execute the output successfully…….

6. CONCLUSION

With technology advancing every day, improvements are made in each and every aspect. In this paper, we have developed a virtual mouse which is based on CNN. The user's hand is detected and the gestures of hand are identified by CNN and according to the gestures different operations are carried out.

The detection of hand is not affected by the lighting conditions and background thus, overcoming the limitations of the previous systems. The developed system has quite high accuracy and is suitable for real-time use.

This project showcases a programme that enables hand gestures as a practical and simple method of software control. A gesture-based presentation controller doesn't need any special markers, and it can be used in real life on basic PCs with inexpensive cameras since it doesn't need particularly high quality cameras to recognise or record the hand movements. The method keeps track of the locations of each hand's index finger and counter tips.

This kind of system's primary goal is to essentially automate system components so that they are easy to control. As a result, we have employed this method to make the system simpler to control with the aid of these applications in order to make it realistic. In this modern world, where technologies is at the peak, there are many facilities available for offering input to any applications running on the computer systems, some of the inputs can be offered using physical touch and some of them without using physical touch (like speech, hand gestures, head gestures etc. Using hand gestures many users can handle applications from distance without even touching it.

But there are many applications which cannot be controlled using hand gestures as an input. This technique can be very helpful for physically challenged people because they can define the gesture according to their need. The present system which we have implemented although seems to be user friendly as compared to modern device or command based system but it is less robust in detection and recognition as we have seen in the previous step.

We need to improve our system and try to build more robust algorithm for both recognition and detection even in the cluttered background and a normal lighting condition. We also need to extend the system for some more class of gestures as we have implemented it for only 6 classes. However we can use this system to control applications like power point presentation, games, media player, windows picture manager etc…

7.REFERENCE

[1] N. Shaker and M. A. Zliekha, "Real-time Finger Tracking for Interaction," 2007 5th International Symposium on Image and Signal Processing and Analysis, Istanbul, 2007.

[2] R. M. Prakash, T. Deepa, T. Gunasundari and N. Kasthuri, "Gesture recognition and fingertip detection for human computer interaction," 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), Coimbatore 2017.

[3] A. Dekate, A. Kamal and K. S. Surekha, "Magic Glove – wireless hand gesture hardware controller," 2014 International Conference on Electronics and Communication Systems (ICECS), Coimbatore, 2014.

[4] J. Suh, M. Amjadi, I. Park and H. Yoo, "Finger motion detection glove toward human- machine interface," 2015 IEEE SENSORS, Busan, 2015.

[5] R. S. Batu, B. Yeilkaya, M. Unay and A. Akan, "Virtual Mouse Control by Webcam for the Disabled," 2018 Medical Technologies National Congress (TIPTEKNO), Magusa, 2018.

[6] A. Mhetar, B. K. Sriroop, A. G. S. Kavya, R. Nayak, R. Javali and K. V. Suma, "Virtual mouse", International Conference on Circuits, Communication, Control and Computing, Bangalore, 2014.

[7] S. K. Kang, M. Y. Nam and P. K. Rhee, "Color Based Hand and Finger Detection Technology for User Interaction," 2008 International Conference on Convergence and Hybrid Information Technology, Daejeon, 2008.

[8] Y. Fang, K. Wang, J. Cheng and H. Lu, "A Real-Time Hand Gesture Recognition Method," in 2007 International Conference on Multimedia & Expo, Beijing, 2007 .

[9] M. Han, J. Chen, L. Li and Y. Chang, "Visual hand gesture recognition with convolutionneural network," in 2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), Shanghai, China, 2016 .

[10] T. D. Grove, K. D. Baker and T. N. Tan, "color-based object tracking," Proceedings. Fourteenth International Conference on Pattern Recognition (Cat. No.98EX170), Brisbane, Queensland, Australia, 1998.

[11] N. Vo, Q. Tran, T. B. Dinh, T. B. Dinh and Q. M. Nguyen, "An Efficient Human- Computer Interaction Framework Using Skin Color Tracking and Gesture Recognition," 2010 IEEE RIVF International Conference on Computing & Communication Technologies, Research, Innovation, and Vision for the Future (RIVF), Hanoi, 2010 pp.