PROJECT REPORT

(Project Term Feb -April 2022)

(*Virtual white board using Google pipeline and open cv*)

Submitted by

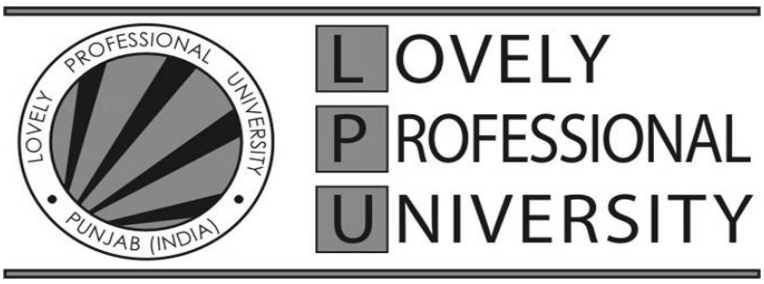
Ankit Mehta **Registration Number :11903842**

Course Code: INT247 – Machine Learning Foundation

Under the Guidance of

(Dr. Sagar Pande Sir)

School of Computer Science and Engineering



DECLARATION

We hereby declare that the project work entitled ("Virtual white board using Google pipeline and open cv") is an authentic record of our own work carried out as requirements of Project for the award of B. Tech degree in _Computer Science and Engineering_ from Lovely Professional University, Phagwara, under the guidance of (Dr. Sagar Pande Sir), during January to April 2022. All the information furnished in this project report is based on our own intensive work and is genuine.

Name of Student: Ankit Mehta
Registration Number: 11903842
Roll no - 30

Ankit Mehta
(Signature of Student)
Date: 27-March-2022

CERTIFICATE

This is to certify that the declaration statement made by this group of students is correct to the best of my knowledge and belief. They have completed this Project under my guidance and supervision. The present work is the result of their original investigation, effort and study. No part of the work has ever been submitted for any other degree at any University. The Project is fit for the submission and partial fulfillment of the conditions for the award of B. Tech degree in Computer Science and Engineering from Lovely Professional University, Phagwara.

Dr. Sagar Pande Sir

Signature and Name of the Mentor

School of Computer Science and Engineering,
Lovely Professional University,
Phagwara, Punjab.

Date: 27-03-2022

ACKNOWLEDGEMENT

I would like to express my gratitude towards my university as well as Dr. Sagar Pande Sir providing me the golden opportunity to do this wonderful Project on **Virtual white board using Google pipeline and open cv which** also helped me in doing a lot of homework and learning. As a result, I came to know about so many new things. So, I am really thanking full to them.

Moreover, I would like to thank my friends who helped me a lot whenever I got stuck in some problem related to my course. I am thankful to have such a good support of them as they always have my back whenever I need.

Also, I would like to mention the support system and consideration of my parents who have always been there in my life to make me choose right thing and oppose the wrong. Without them I could never **had learned and became a person who I am now.**

>→→→→→→→→→→→

TABLE OF CONTENTS

5. **Software Requirement Analysis (Software Tools)**

- Introduction
- Open CV
- Python
- Mediapipe

6. Source Code (wherever applicable) or System Snapshots

- Overview
- HandTrackingModule.py
- VirtualPainter.py
- Output

7. Result

8. Conclusion

- User Manual: A complete document (Help Guide) of the software developed**.**
- **Project Legacy**

9. Future scope

10. References

11. Appendices

12. Publication of paper on project Work (attach abstract and certificate)

13. Biodata of Projectees

Abstract

In the current pandemic situation, we have seen one of the reasons for spreading of corona virus is surface touching, and to avoid this sort of spread in the virus we have come up with a project which is titled as “Drawing on Air”, As the title itself indicates, we are not going to touch the surface but we are going to draw on air and what we have drawn on air will be displayed on the screen / monitor. This will help us reduce the spread of the virus in the pandemic situations. In this pandemic we have seen maximum of the transactions are been made online but for deposit of money or for transfer of money in person we see people use ATMs which is very risky and there are chances of spreading of the virus at such point of times we can use this as a prevention method to a great extent. Not only in ATMs but also in offices for biometrics or in cyber cafes and many more areas we can use this project.

Virtual whiteboard is where we can draw by just capturing the motion of a colored marker with a camera. One colored object at the tip of the finger is mainly used as the marker. We are here now, using the techniques of computer vision in open cv to build this project. The required language for this project is python due to its more exhaustive libraries and easy to make use of the syntax and but understanding the basics as well as it can be implemented in any open cv supported languages The colour tracking and detection processes are used to achieve the goal of this project. The color marker here used is detected and mask is produced. The next steps of morphological operations on the mask produced those are Erosion and Dilation. Erosion makes the impurities present in the mask to get reduced and Dilation further regains the eroded main mask.

1. INTRODUCTION

Machine learning is a branch of Artificial Intelligence which is used to analyze the data more smartly. It automates the process using certain algorithms to minimize human intervention in the process.

Machine learning is the practice of building systems, known as models, that can be trained using data to find patterns which can then be used to make predictions on new data.

Unlike a lot of other programming, a machine learning model is *not* a rules-based system where a series of 'if/then' statements are used to determine outcomes (e.g., 'If a student miss more than 50% of classes then automatically fail them').

Instead, machine learning model examines the statistical relationships between data points in a data set with defined outcomes, and then applies what it has learned about those relationships to analyze and predict outcomes for a new data set.

In this machine learning project, we are going to make a Virtual board using google pipeline and open cv.

This Project is implemented with the aim of building and developing software that can replace the using of interactive multimedia board. The Software is called a Virtual Whiteboard, that used by using a USB camera as video sensor, infrared pointer as a pointer coordinates, and a projector or monitor as a media for interaction between user and computer. This Desktop-based software runs at Microsoft's Windows operating system environments using image-processing technology from OpenCV and Google pipeline programming language. Virtual Whiteboard software is expected to facilitate and contribute new in the world of technology, especially image processing and the interaction between humans and computers

.OpenCV and Jupyter Notebook with the help of object detection. This project can be a very helpful project for the society for the pandemic situations. It was a simple illustration of image processing capabilities of OpenCV.

In This Project I made this virtual whiteboard where you can write in thin air using only your fingers and a live camera feed. In this project , we have create a virtual white board using google pipeline and open cv . We will first track our hand and get its landmarks and then use the points to draw on the screen. We will use two fingers for selection and one finger for drawing. And the best part is that all of this will be done in real-time.

I used Google's mediapipe library and its pretrained hand pose estimation model to detect landmarks on a hand. I then collected a small dataset of different hand postures corresponding to different actions, such as write and erase. Then I trained a simple feed forward neural network to identify those postures based on the landmarks extracted by the pretrained hand pose estimation model.

It works pretty good overall, however there are some inaccuracies in detecting the correct pose made by a hand.

Libraries used

PyTorch , Mediapipe , OpenCV

Fig 1. Drawing on a virtual canvas using hand pose estimation

2. Literature Review / Related works =>

An application has been developed that allows you to trace a pencil, pen, or paintbrush, that is, an oblong object that looks like a pen. Tracking allows you to draw or write in the air without touching the screen, that is, without physically affecting anything. Tracking is done using the Leap Motion controller. The application was developed using JavaScript as the main front end and Python as the back end. Interaction with the application is shown in Figure 2. The user writes some information, then the server processes the request in real-time using ajax technology and the jQuery library, and then displays the data on the computer screen.

i. Existing System ->

Tradition ways of education is on the offline mode but corona virus pandemic realize us about the concept of online classes. Offline Education means a student needs to go in a school, in a classroom, and attend a class face to face with a teacher. In an offline setting, students listen to long lectures and take notes, sometimes easily get bored and might doze off a little. offline learning, it is typically carried out between office hours and doesn't offer as much flexibility to the learner or the trainer.

Offline classes are the traditional mode of taking classes which we can say is going out to a school or an institute and attending classes with an instructor. Well here is for the readers that they should not be confused between online and offline learning because they are both different modes serving the same purpose.

Online Education is a very flexible learning system that allows students to study solely via the internet on their own computer at home, or wherever they see fit. With online classes, and new technology teaching techniques such as interactive whiteboards and videos for example, students engage with the class and activities given by the teacher.

An online whiteboard (also known as a digital whiteboard or virtual whiteboard) is used the same way as a classroom whiteboard would be used offline. An online whiteboard, as the name suggests, is a board with a white background and that you can write things on with various colored markers.

The board can be used by different people and interactively and, because it has a shiny background, it is easy to see the different colours on it. Thus, the drawings and notes made on the whiteboard are easy to see.

ii. What's new in the system to be developed

With the advent of digitalization in the education industry, online classes have become the need of the hour for students across the globe. Online classes offer flexibility and an ample variety of academic opportunities, such as studying from any location with affordable fees, saving time and money. The advantages of online classes are immense. Students can enjoy the benefits of online classes which they might not be able to while attending traditional classes due to several reasons.

Digital learning has made learning interesting and fun for students. Earlier, students had two options – either they would travel daily to and from the coaching classes or they would compromise with a local tutor who may not be that great in teaching. But not anymore! Now, you can learn from the best online classes under the guidance of the highest-rated teachers.

In a virtual classroom, an online whiteboard is a wonderful tool that you can use as a teacher to make the classes more interesting and to draw and write things onto a screen that everyone can see in real-time.

these video-conferencing platforms (that you can use for Online teaching) all have a whiteboard feature:

Zoom, Adobe connect, Webex, GoToWebinar, GotoTraining , GoToMeeting.

Fig 2

iii. Virtual whiteboard

A virtual whiteboard is a digital application that functions like a traditional whiteboard, but is hosted virtually. Digital whiteboards can integrate with other video conferencing and screen sharing platforms to allow for collaboration even when you are not physically in the same room. A virtual whiteboard has multiple colors, shapes and templates to choose from and allows whiteboards to be saved in shareable files for easy access in the future.

A digital whiteboard can enhance a virtual or in person meeting by encouraging collaboration. A whiteboard can help people visualize a process. Whiteboards increase collaboration by allowing participants to easily add ideas to the whiteboard with sticky notes or colored markers.

A whiteboard can help increase productivity as it allows users to easily draw processes with different shapes and colors. A virtual whiteboard further increases productivity as a user can easily copy or remove parts of the whiteboard and even save the file for future use.

Drawing on a whiteboard can be intimidating - especially with others watching you. A virtual whiteboard makes drawing easy. Many virtual whiteboards allow you to type text and select from a variety of shapes to make your digital whiteboard experience easier.

If you are a skilled writer or artist, you can freehand draw with a variety of colors and pen thickness with your computer mouse, trackpad or touchscreen. If your computer has the option to be touchscreen, that is the best simulation of writing on a physical whiteboard

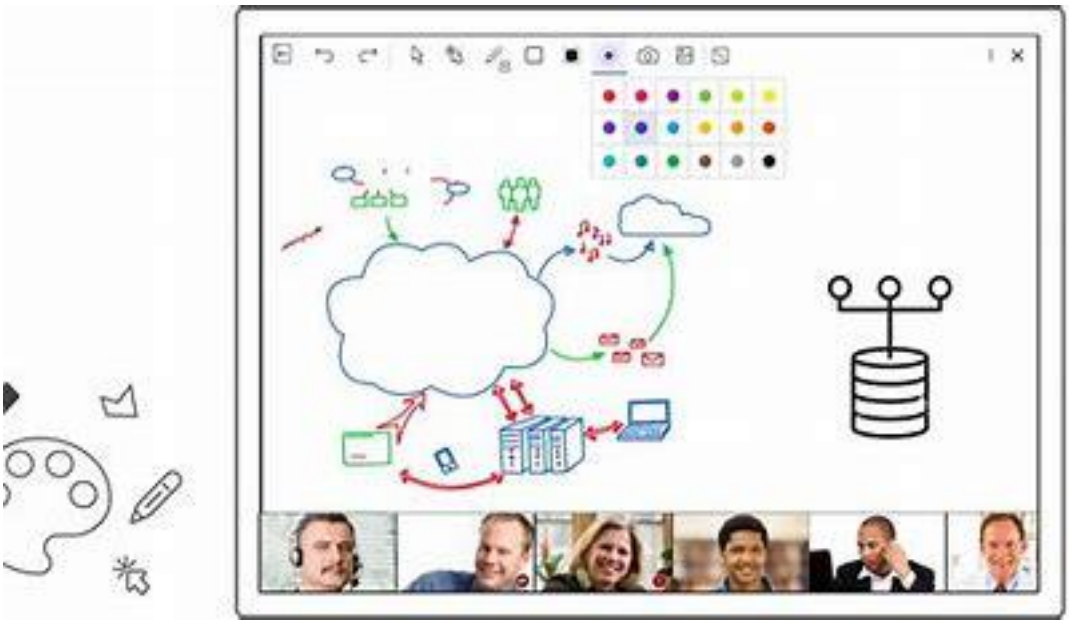


Fig 3

iv. Profile of the Problem. Rationale/Scope of the Project

Virtual white board project explains about a concept of developing a virtual class room with teach and student interaction which can be accessed from all over the world. This system works as client server model where client is a student and server acts as teacher.

Teacher will be located at fixed area and students from different countries can interact with teacher and share knowledge. Teacher can clear doughts , take video classes and audio classes and also use chatting feature and file sharing features for providing best education services.

In existing system manual methods like attending classes is old trend in present trend online is playing important role in education. As the improvement of video and audio chatting with high speed internet connection is increased online education had became a upcoming trend in education.

Already there are many online education website which provide paid services for students. This application is implemented in four modules client module, server module, network module and the file transfer module.

Advantages of this project are students from various part of world can interact with teacher and it is a web application, there is no need of any client side software installation, multiple users can interact with teacher at a time.

Virtual reality technologies represent another step towards discoveries and applications. The global pandemic has influenced another surge of interest in VR. In connection with the forced transition to a remote mode of operation, the question of using tools for online learning arose sharply. Today the leading platforms are Microsoft Teams, Zoom, Skype, etc. These platforms

have great functionality for organizing distance learning. However, the inclusion of VR technology allows you to make learning more lively and closer to reality. The development of a virtual reality application that allows you to use a personal computer as a whiteboard .

Vi. **Problem Analysis**

Ever wanted to draw your imagination by just waiving your finger in air. In this post we will learn to build an Air Canvas which can draw anything on it by just capturing the motion of a coloured marker with camera. Here a coloured object at tip of finger is used as the marker.

We will be using the computer vision techniques of OpenCV to build this project. The preffered language is python due to its exhaustive libraries and easy to use syntax but understanding the basics it can be implemented in any OpenCV supported language.

Here Colour Detection and tracking is used in order to achieve the objective. The colour marker in detected and a mask is produced. It includes the further steps of morphological operations on the mask produced which are Erosion and Dilation. Erosion reduces the impurities present in the mask and dilation further restores the eroded main mask.

vii. Product definition

The Virtual WhiteBoard is a project I made using the OpenCV and Mediapipe Python libraries. Using your index and middle finger you can select colors or the eraser in the palette above. Using the index finger you can proceed to either draw with the color or erase.

Libraries used

- PyTorch
- Mediapipe
- OpenCV

viii. **(**Project Plan) **Algorithm**

Start reading the frames and convert the captured frames to HSV colour space.(Easy for colour detection)

Prepare the canvas frame and put the respective ink buttons on it. 3.. Adjust the trackbar values for finding the mask of coloured marker.

Preprocess the mask with morphological operations.(Erotion and dilation)

Detect the contours, find the center coordinates of largest contour and keep storing them in the array for successive frames .(Arrays for drawing points on canvas)

Finally draw the points stored in array on the frames and canvas .

Requirements: python3 , numpy , opencv installed on your system.

Ix. Feasibility Analysis – .

We will train our laptop or our screen to read whatever the user will be writing Infront of the screen. We have used Open CV [2] for object detection that is with which the user will be writing on air and we have used Python language for coding since

Start -> Image collection -> Image processing -> Feature extraction -> Image detection -> output

All we need in this project is a laptop with a web camera installed in it. We will train our laptop or our screen to read whatever the user will be writing Infront of the screen. We have used Open CV [2] for object detection that is with which the user will be writing on air and we have used Python language for coding since Python is an easy language and easily understandable and also, we have used Jupyter Notebook latest version since it produces a native and managed code. Firstly, after code is executed, we get a white screen displayed in the monitor and whatever we write on air Infront of camera it tracks the object whose properties we have declared in the code. After the tracked points are connected and projected on the screen.

3. Overview of the Project - >

This Project is implemented with the aim of building and developing software that can replace the using of interactive multimedia board. The Software is called a Virtual Whiteboard, that used by using a USB camera as video sensor, infrared pointer as a pointer coordinate, and a projector or monitor as a media for interaction between user and computer.

First and foremost, we must find an appropriate color range for our target-colored object, this range will be used in cv2.inrange() function to filter out our object. We will also save our range array as a .npy file in our disk so we can access it later.

Since we are trying to go for color detection, we will convert our RGB (or BGR in OpenCV) format image to HSV (Hue, Saturation, Value) color format as it's much easier to manipulate colors in that model.

This Desktop-based software runs at Microsoft's Windows operating system environments using image-processing technology from OpenCV and python programming language. Virtual Whiteboard software is expected to facilitate and contribute new in the world of technology, especially image processing and the interaction between humans and computers.

- This entire application is built fundamentally on contour detection. It can be thought of as something like closed color curves on compromises that have the same color or intensity, it's like a blob. In this project we use color masking to get the binary mask of our target color pen, then we use the counter detection to find the location of this pen and the contour to find it.

When we work with it then it is a matter of literally connecting the dots, we just have to draw a line using the x, y dots of the previous position of the pen with new x, y dots and that's it. , We have a virtual pen.

There are 3 main points in the code

- Apply morphological operations.
- Detect and track the colored object.
- Find the object's x,y coordinates

4. Circuit Description -

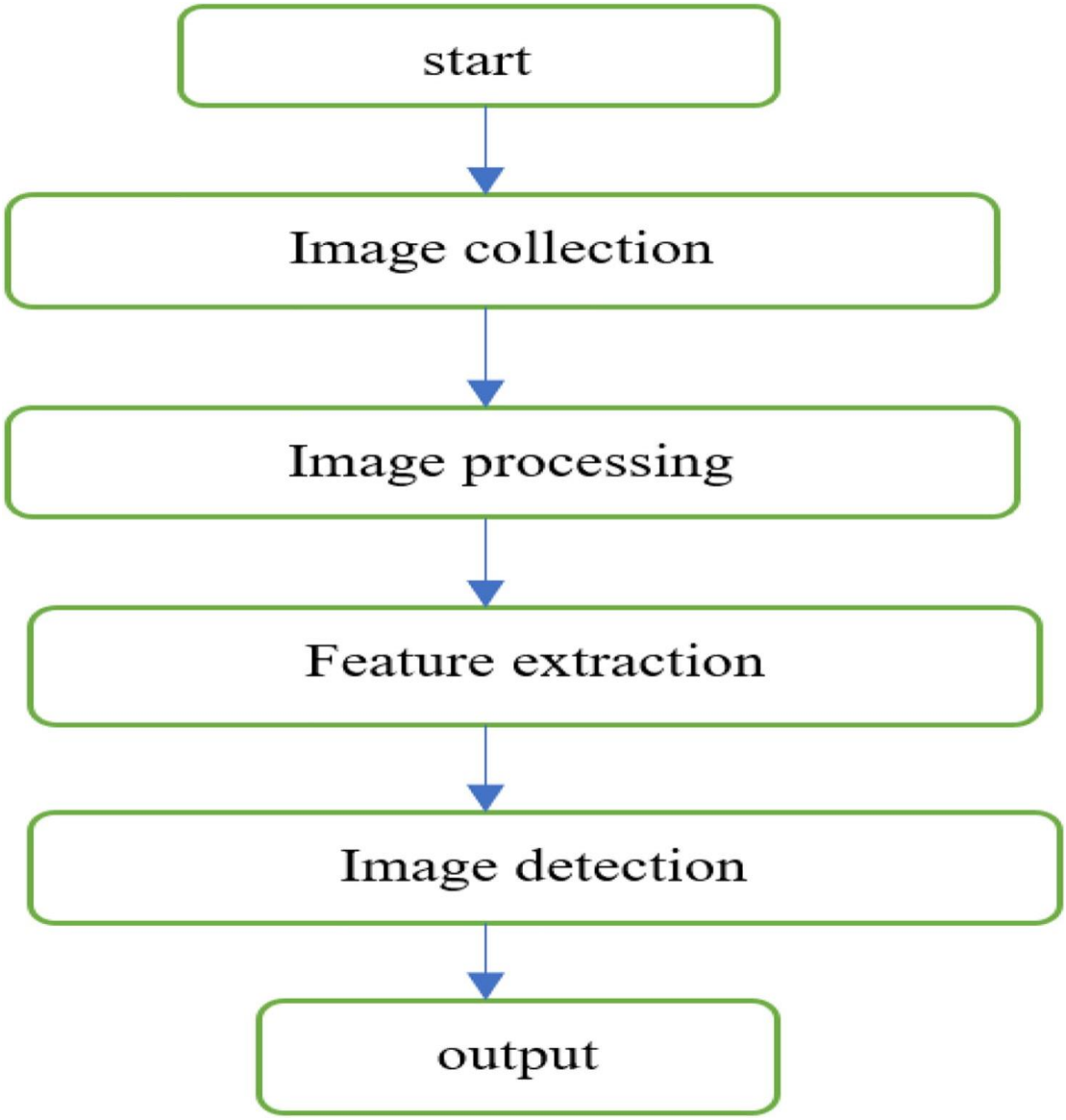


Fig. 4. Work flow of proposed methodology.

- Machine Learning Model -

There are Seven Steps of Machine Learning

1. Gathering Data - Now, we download the data and look at it. Determine which features are available and which aren't. The Quality and quantity of our data dictate how accurate our model is and this data we will use for training.

2. Preparing that data - We need to dig deeper. This step helps us understand the nature of variables (skewed, missing, zero variance feature) so that they can be treated properly. It involves creating charts, graphs (univariate and bivariate analysis), and cross-tables to understand the behavior of features.

3. Choosing a model - Using a suitable algorithm, we train the model on the given data set.

4. Training - Finally, we test the model on the unseen data (test data) set. The goal of training is to predict correctly as often as possible.

5. Evaluation - sees some matric or combination of matrices to measure objective performance of model. Test the model against previously unseen data.

Selecting a performance measure: A typical performance measure for regression problem is the Root Mean Square Error (RMSE)

6. Hyperparameter Tuning - Hyperparameter tuning is the problem if choosing a set of optimal parameters for a learning algorithm.

7. Prediction - For predict the data first we need to save our model and launch that model.

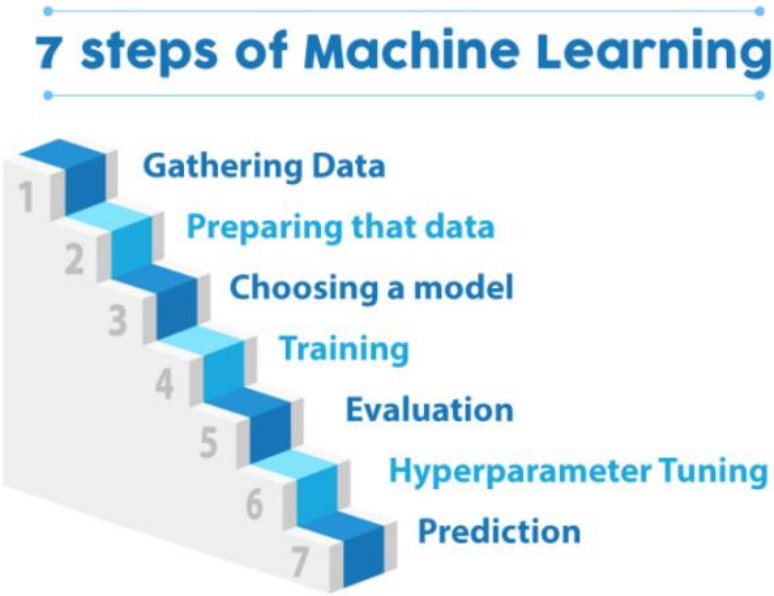


Fig 5. & steps of Machine Learning

- **Hue setup in HSV adjustment taskbar.**

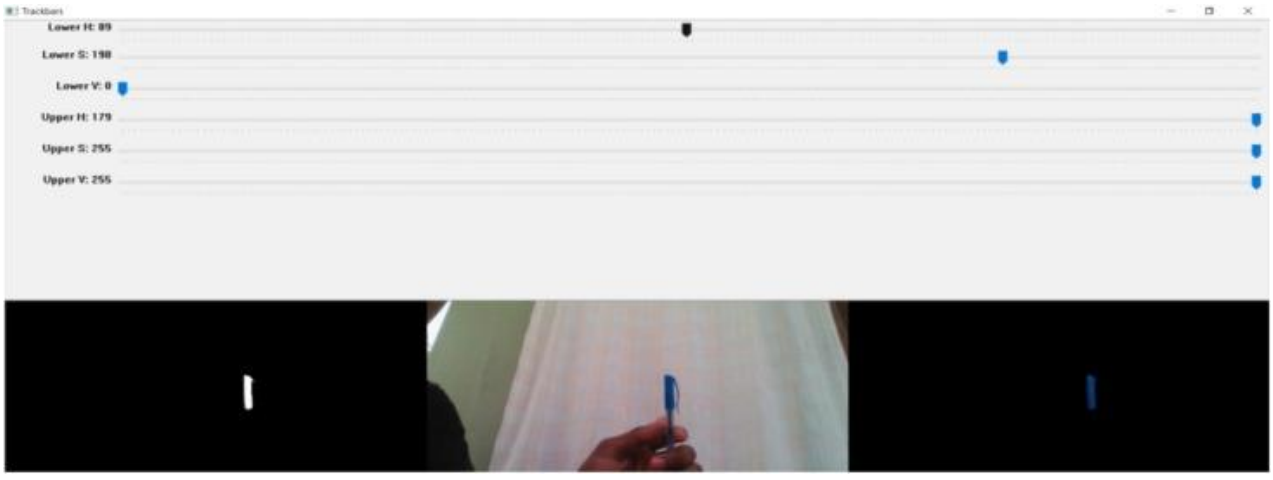


Fig .6

- **Saturation setup in HSV ADJUSTMENT taskbar.**

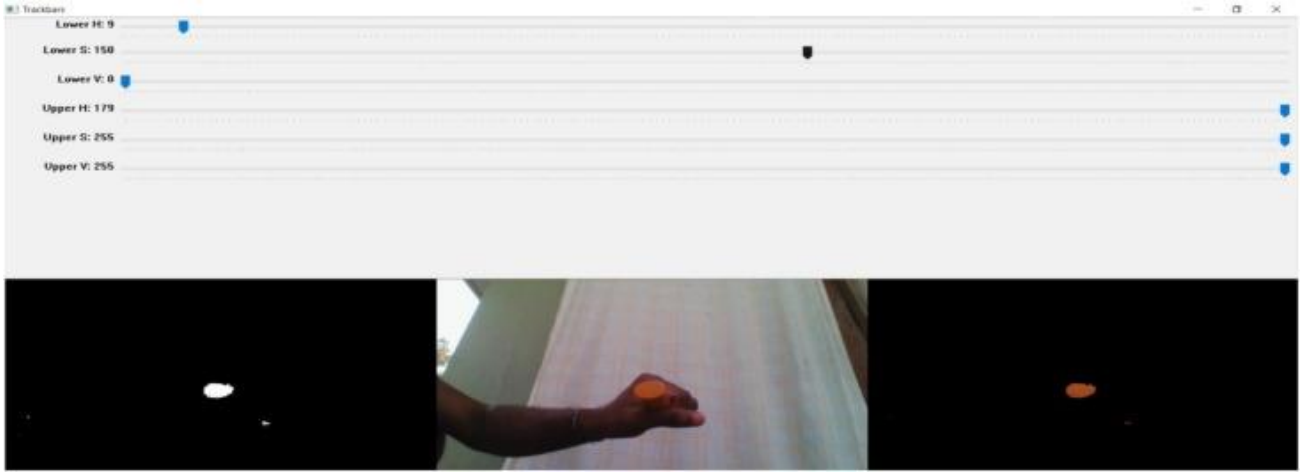


Fig 7

- **Value setup in HSV adjustment Taskbar.**

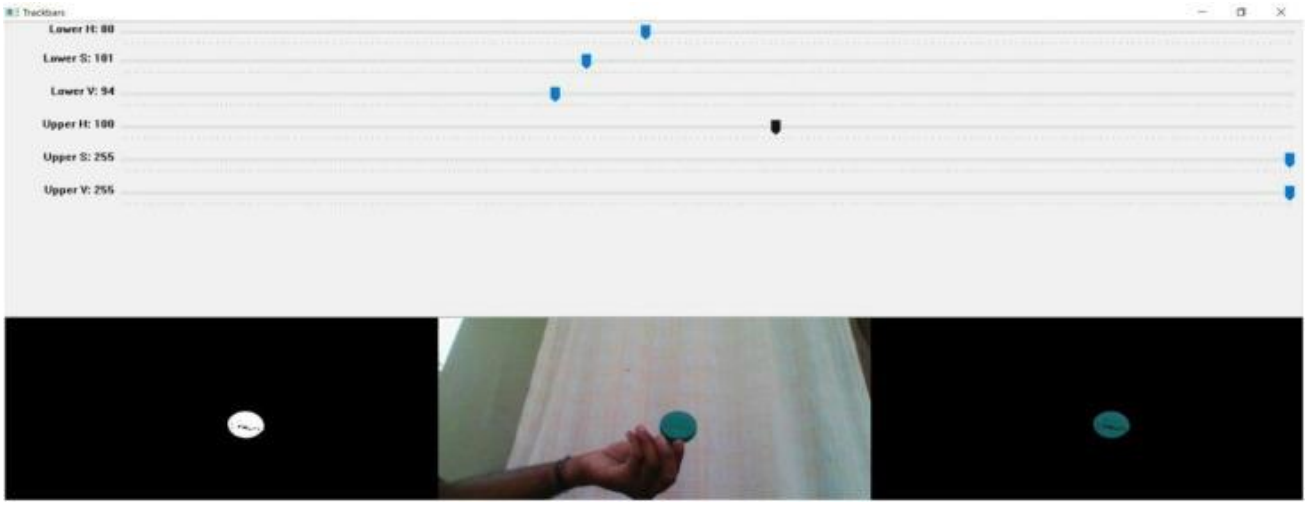


Fig 8

- **Output window showing the drawn text ‘smile’ with emoj**

Fig 9

- **Output window showing the drawn text ‘Hello’.**

Fig 10

Output window showing the drawn text ‘Hi’

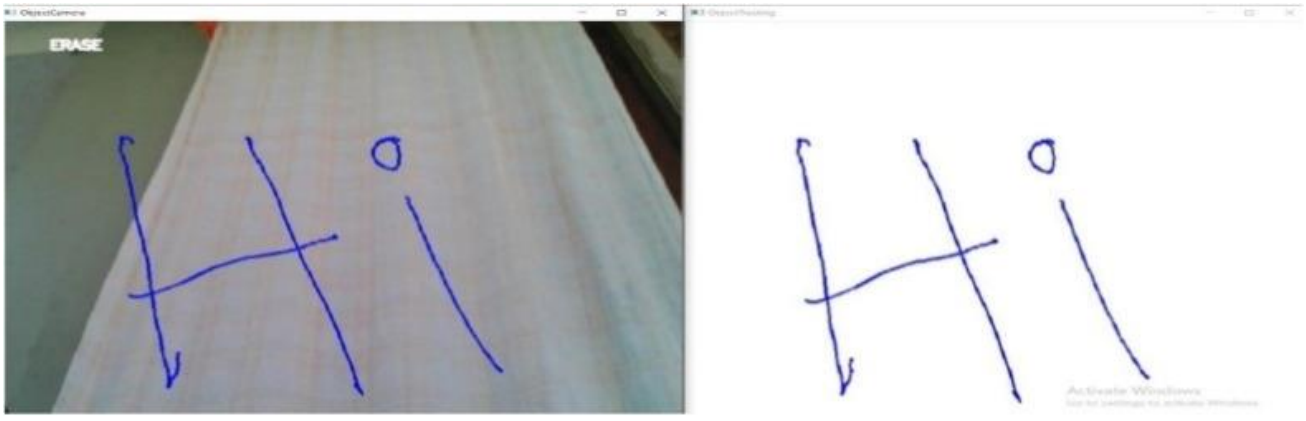


Fig 11

- **Open CV**

OpenCV is an abbreviation of the Open Computer Vision, which is open source libraries which specialize to perform image processing. The Objective is that computers have the ability similar to the way the human visual processing. This library is made for the language C / C ++ as a realtime optimization applications, has API (Application Programming Interface) for high -level and low-level, there fungsi2 are ready for loading, saving, acquisition of images / videos.

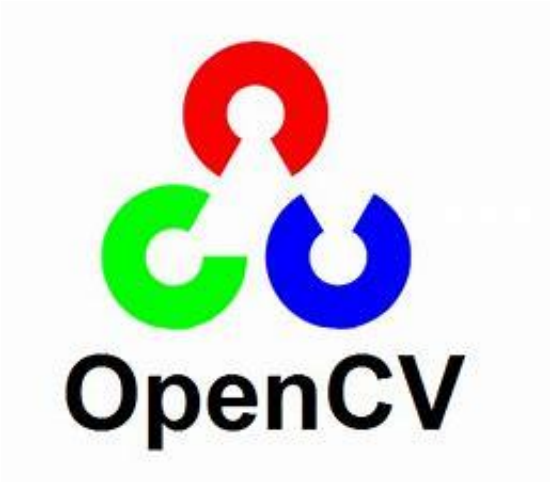


Fig 12

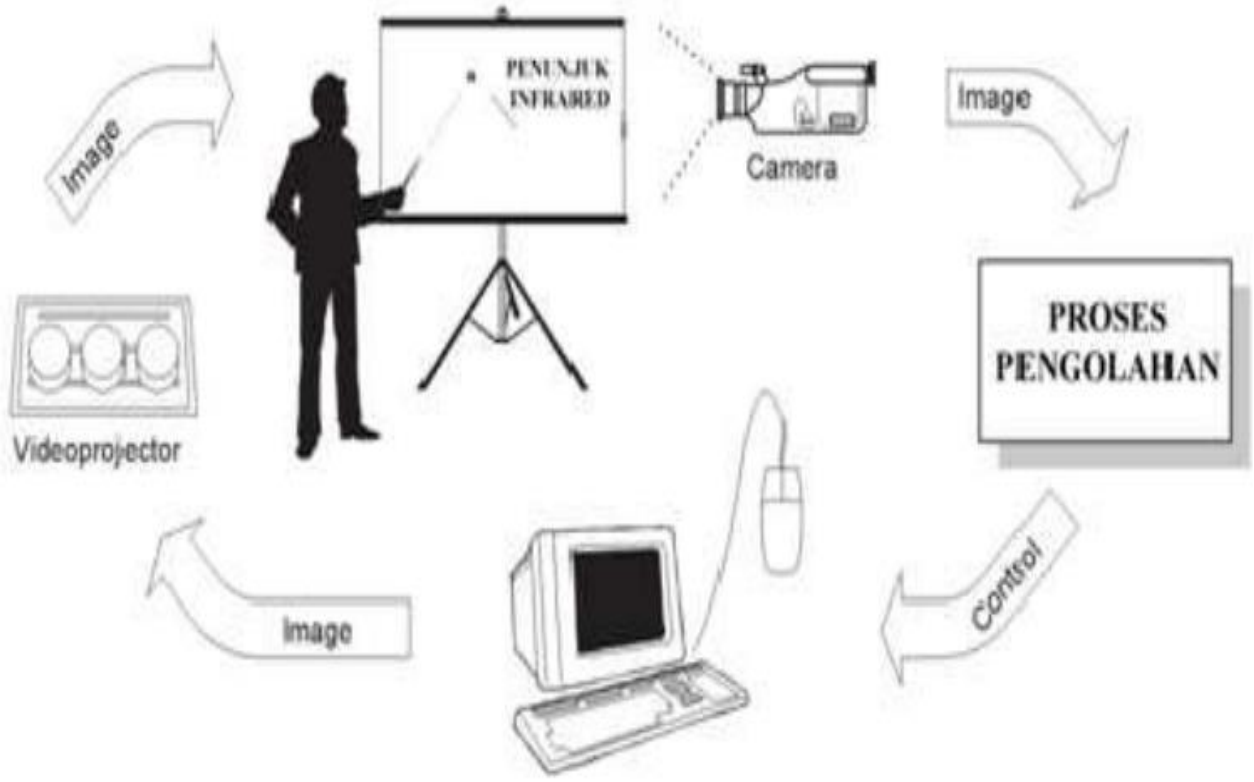


Fig 13

- **Optimum Variables**

From the results of running the program obtained the best effectiveness of the program can be achieved under the following conditions

No	Variabel	Value
1	Distance between Camera and Screen	
	a. Monitor Screen	2-3 meter
	b. Projector Screen	1,5 meter
2	Lighting	lighting with low sun light
3	Treshold Value	50-100
4	Frame per Second	25-30 fps

Fig 14

Open project from PyCharm or suitable IDE and run the VirtualPainter.py file or Open Command Prompt and Run python VirtualPainter.py :

- **Steps for this implementation:-**

Install Required Packages and Libraries

- pip upgrade --user pip
- pip install opencv-python
- pip install numpy
- pip install mediapipe

1. Importing libraries, assigning BGR values for color.

imports

- import cv2
- import numpy as np
- import time

2. Creating a window that provides various options.
3. Capturing a video using a webcam and grabbing frames.
4. Using different morphing techniques
5. Displaying output
6. So, now let's begin with this interesting implementation.

- ## Related work

To implement this work, one should first learn object tracking [1] and one should be aware of OpenCV [2] and Jupyter Notebook. Using Object tracking we detect the HSV

code [3] that is similar like RGB color code [4]. HSV is Hue, Saturation and Value respectively.

Typically tracking algorithms [5] are quicker than detection algorithms. We know a lot about the objects appearance when we are monitoring an object which was observed in the previous frame, we already know the position in the next frame [6], we can use all this data to determine the position of object in the next frame, and to precisely find the object with a limited scan around the estimated location of the object. In the other hand, a successful monitoring algorithm can manage any amount of obstruction. The motion model estimates the objects precise location. To have a more precise assessment based on appearance, the appearance model makes improvements on this assessment.

- **Work flow**

Methodology

So basically, the screen detects the color whose properties we declare in the code. H, S, V are Hue, Saturation and Value respectively. This color code is similar to the RGB color code. You will have to run the object Tracking code e beforehand in order to get the HSV value. All you have to do is slide the taskbars until the only remaining white parts of the “Thresholded Image” windows are correspondent to object [7], [8] that you want to detect in real life.
For example, if you want to detect the yellow sharpie, you will have to slide the trackbars so that the white parts of the “Thresholded Image” windows are actually the threshold image of the yellow sharpie [11], [12]. After you get HSV value you want, save it for near future references. Then run the webcam code [9], [10] remember to change the HSV values into whatever HSV value you found previously. Then hit the “debug” button to draw.

Whiteboard images generally contain less contrast and low brightness as they would be captured in mobile under normal room light conditions. Enhancing whiteboard images makes text readable and gives an image with high contrast and brightness.

We will apply different image-processing techniques to enhance whiteboard images using OpenCV in Python. From this whiteboard-cleaner gist that enhances whiteboard images using ImageMagick, we will implement those ImageMagick methods in Python.

We will apply series of image-processing methods and effects to enhance whiteboard images in the following order

- Difference of Gaussian (DoG)
- Negative effect

- Contrast Stretching
- Gaussian blur
- Gamma correction
- Color balance

• Results and discussions

After debug the webcam automatically turns on and we get to draw Infront of the webcam and then the output screen appears and we get the output with the help of the HSV values which we have already declared in code. The HSV adjustment plays major role in this implementation, based on its value, the text that the person drawing on air will be reflected and visible clearly at the output.

Fib 15. Implementation of Project

5. Software Requirement Analysis(Software Tools)

- **Open CV**

Open CV (Open-Source Computer Library) is a programming feature library specifically targeted at computer vision [2] in real time. This is an open-source platform and free for use. Its primary interface is in c++, it still preserves an older c interface that is less detailed but vast. In the Python interface, all the latest advancements and algorithms appear. It is major open-source computer vision, machine learning and image processing library and now it plays an important part in real time activity in today's systems. We will use this to process photographs and videos to recognize human beings, faces or even handwriting. Python is able to process the OpenCV array structure for review as it is combined with different libraries such as NumPy. We use vector space to recognize the image pattern and its different characteristics and perform arithmetic computations on these traits. It is accessible on Windows, Linux, iOS etc., with Python, C++, C and Java as interfaces.

OpenCV Functionality

- Image/video I/O, processing, display (core, imgproc, highgui)
- Object/feature detection (objdetect, features2d, nonfree)
- Geometry-based monocular or stereo computer vision (calib3d, stitching, videostab)
- Computational photography (photo, video, superres)
- Machine learning & clustering (ml, flann)
- CUDA acceleration (gpu)

Image-Processing

Image processing is a method to perform some operations on an image, in order to get an enhanced image and or to extract some useful information from it.

If we talk about the basic definition of image processing then "Image processing is the analysis and manipulation of a digitized image, especially in order to improve its quality".

Digital-Image :

An image may be defined as a two-dimensional function f(x, y), where x and y are spatial(plane) coordinates, and the amplitude of fat any pair of coordinates (x, y) is called the intensity or grey level of the image at that point.

In another word An image is nothing more than a two-dimensional matrix (3-D in case of coloured images) which is defined by the mathematical function f(x, y) at any point is giving the

pixel value at that point of an image, the pixel value describes how bright that pixel is, and what colour it should be.

Image processing is basically signal processing in which input is an image and output is image or characteristics according to requirement associated with that image.

Image processing basically includes the following three steps:

- Importing the image
- Analysing and manipulating the image
- Output in which result can be altered image or report that is based on image analysis.

Reading an image

```
# Importing the OpenCV library
import cv2
# Reading the image using imread() function
image = cv2.imread('image.png')

# Extracting the height and width of an image
h, w = image.shape[:2]
# Displaying the height and width
print("Height = {},  Width = {}".format(h, w))
```

Extracting the RGB values of a pixel

```
# Extracting RGB values.
# Here we have randomly chosen a pixel
# by passing in 100, 100 for height and width.
(B, G, R) = image[100, 100]

# Displaying the pixel values
print("R = {}, G = {}, B = {}".format(R, G, B))

# We can also pass the channel to extract
# the value for a specific channel
B = image[100, 100, 0]
print("B = {}".format(B)
# We will calculate the region of interest
# by slicing the pixels of the image
roi = image[100 : 500, 200 : 700]
```

Resizing the Image

```
# resize() function takes 2 parameters,
```

```
# the image and the dimensions
resize = cv2.resize(image, (800, 800))
```

```
# Calculating the ratio
ratio = 800 / w

# Creating a tuple containing width and height
dim = (800, int(h * ratio))

# Resizing the image
resize_aspect = cv2.resize(image, dim)
```

Rotating the Image

```
# Calculating the center of the image
center = (w // 2, h // 2)

# Generating a rotation matrix
matrix = cv2.getRotationMatrix2D(center, -45, 1.0)

# Performing the affine transformation
rotated = cv2.warpAffine(image, matrix, (w, h))
```

Displaying text

```
# Copying the original image
output = image.copy()

# Adding the text using putText() function
text = cv2.putText(output, 'OpenCV Demo', (500, 550),
                   cv2.FONT_HERSHEY_SIMPLEX, 4, (255, 0, 0), 2)
```

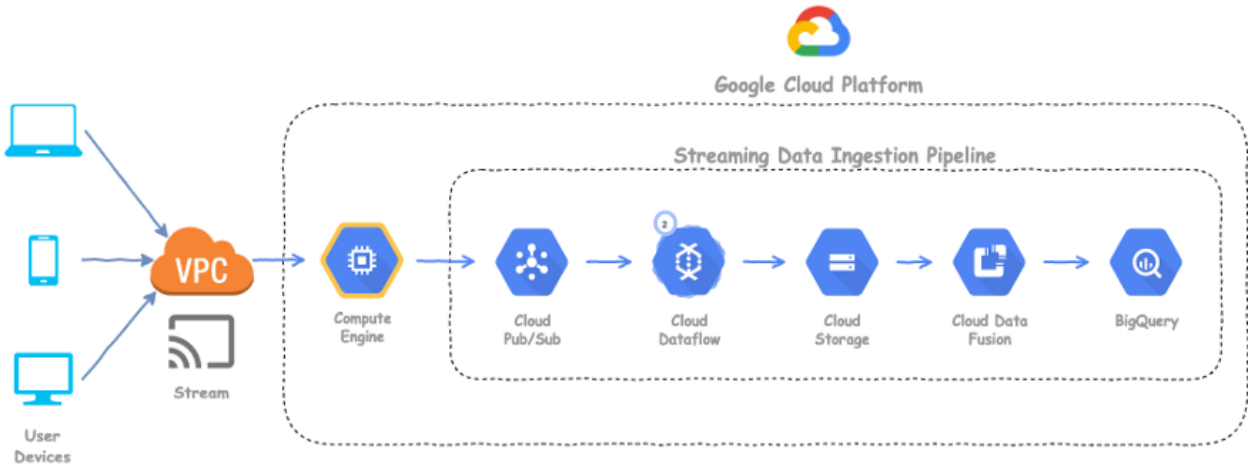
Python

Python is a high-level programming language. It is mainly famous for code reusability and simplicity. Even though it is slower it has an important characteristic of python that it can be

easily extended within c. Because of this characteristic we can write computationally intrinsic codes in C++/C. Python supports different forms of programming patterns such as procedural programming, object-oriented programming, functional programming etc. It consists of NumPy library. NumPy is highly powerful and optimized for mathematical operations. In order to take advantages of multi-core computing, all items are written in optimized C or C++.

Google ML pipeline

Machine learning (ML) workflows include steps to prepare and analyze data, train and evaluate models, deploy trained models to production, track ML artifacts and understand their dependencies, etc. Managing these steps in an ad-hoc manner can be difficult and time-consuming.



Fib 16

Mediapipe –

Object Detection is one of the leading and most popular use cases in the domain of computer vision. Several object detection models are used worldwide for their particular use case applications. Many of these models have been used as an independent solution to a single computer vision task with its own fixed application. Combining several of these tasks into a single end-to-end solution, in real-time, is exactly what MediaPipe does.

MediaPipe is an open-source, cross-platform Machine Learning framework used for building complex and multimodal applied machine learning pipelines. It can be used to make cutting-edge Machine Learning Models like face detection, multi-hand tracking, object detection, and tracking, and many more. MediaPipe basically acts as a mediator for handling the implementation of models for systems running on any platform which helps the developer focus more on experimenting with models, than on the system.

Possibilities with MediaPipe:

- Human Pose Detection and Tracking High-fidelity human body pose tracking, inferring a minimum of 25 2D upper-body landmarks from RGB video frames
- Face Mesh 468 face landmarks in 3D with multi-face support
- Hand Tracking 21 landmarks in 3D with multi-hand support, based on high-performance palm detection and hand landmark model
- Holistic Tracking Simultaneous and semantically consistent tracking of 33 pose, 21 per-hand, and 468 facial landmarks
- Hair Segmentation Super realistic real-time hair recoloring
- Object Detection and Tracking Detection and tracking of objects in the video in a single pipeline
- Face Detection Ultra-lightweight face detector with 6 landmarks and multi-face support
- Iris Tracking and Depth Estimation Accurate human iris tracking and metric depth estimation without specialized hardware. Tracks iris, pupil, and eye contour landmarks.
- 3D Object Detection Detection and 3D pose estimation of everyday objects like shoes and chairs

MediaPipe Holistic:

Mediapipe Holistic is one of the pipelines which contains optimized face, hands, and pose components which allows for holistic tracking, thus enabling the model to simultaneously detect hand and body poses along with face landmarks. one of the main usages of MediaPipe holistic is to detect face and hands and extract key points to pass on to a computer vision model.

Detect face and hands using Holistic and extract key points

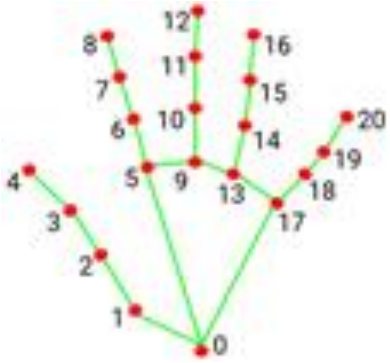
The following code snippet is a function to access image input from system web camera using OpenCV framework, detect hand and facial landmarks and extract key points.

```
Install dependencies
pip install opencv-python
pip install mediapipe
'''
# Import packages
import cv2
```

```
import mediapipe as mp

#Build Keypoints using MP Holistic
mp_holistic = mp.solutions.holistic # Holistic model
mp_drawing = mp.solutions.drawing_utils # Drawing utilities

def mediapipe_detection(image, model):
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) # COLOR CONVERSION BGR 2 RGB
    image.flags.writable = False                  # Image is no longer writable
    results = model.process(image)                 # Make prediction
    image.flags.writable = True                   # Image is now writable
    image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR) # COLOR CONVERSION RGB 2 BGR
    return image, results

def draw_landmarks(image, results):
    mp_drawing.draw_landmarks(
      image, results.face_landmarks, mp_holistic.FACE_CONNECTIONS) # Draw face connections
    mp_drawing.draw_landmarks(
      image, results.pose_landmarks, mp_holistic.POSE_CONNECTIONS) # Draw pose connections
    mp_drawing.draw_landmarks(
      image, results.left_hand_landmarks, mp_holistic.HAND_CONNECTIONS) # Draw left hand
connections
    mp_drawing.draw_landmarks(
      image, results.right_hand_landmarks, mp_holistic.HAND_CONNECTIONS) # Draw right hand
connections

def draw_styled_landmarks(image, results):
    # Draw face connections
    mp_drawing.draw_landmarks(
      image, results.face_landmarks, mp_holistic.FACE_CONNECTIONS,
      mp_drawing.DrawingSpec(color=(80,110,10), thickness=1, circle_radius=1),
      mp_drawing.DrawingSpec(color=(80,256,121), thickness=1, circle_radius=1))
    # Draw pose connections
    mp_drawing.draw_landmarks(image, results.pose_landmarks, mp_holistic.POSE_CONNECTIONS,
                        mp_drawing.DrawingSpec(color=(80,22,10), thickness=2, circle_radius=4),
                        mp_drawing.DrawingSpec(color=(80,44,121), thickness=2, circle_radius=2)
                        )
    # Draw left hand connections
    mp_drawing.draw_landmarks(image, results.left_hand_landmarks,
mp_holistic.HAND_CONNECTIONS,
                        mp_drawing.DrawingSpec(color=(121,22,76), thickness=2, circle_radius=4),
                        mp_drawing.DrawingSpec(color=(121,44,250), thickness=2, circle_radius=2)
                        )
    # Draw right hand connections
```

```
    mp_drawing.draw_landmarks(image, results.right_hand_landmarks,
mp_holistic.HAND_CONNECTIONS,
                              mp_drawing.DrawingSpec(color=(245,117,66), thickness=2, circle_radius=4),
                              mp_drawing.DrawingSpec(color=(245,66,230), thickness=2, circle_radius=2)
                              )
#Main function
cap = cv2.VideoCapture(0)
# Set mediapipe model
with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) as holistic:
    while cap.isOpened():

        # Read feed
        ret, frame = cap.read()

        # Make detections
        image, results = mediapipe_detection(frame, holistic)
        print(results)

        # Draw landmarks
        draw_styled_landmarks(image, results)

        # Show to screen
        cv2.imshow('OpenCV Feed', image)

        # Break gracefully
        if cv2.waitKey(10) & 0xFF == ord('q'):
            break
    cap.release()
    cv2.destroyAllWindows()
```

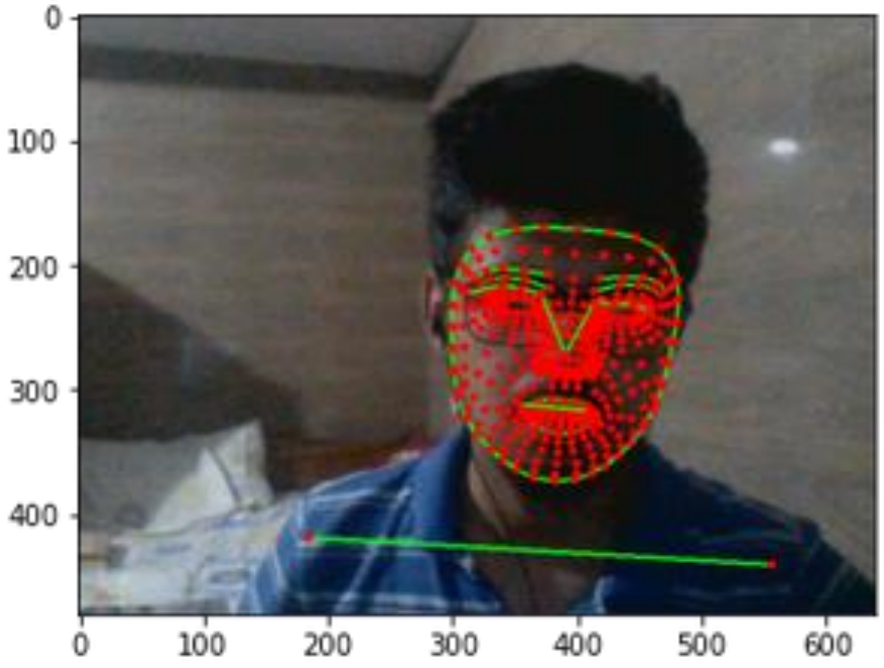
- Landmarks that can be detected using Mediapipe Holistic - >



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

Hand Landmarks

Face Landmarks



Fib 17

Fig 18

6. Result –

➢ Source Code

- **overview**

In this project, we are going to create a virtual whiteboard using ML. We will first track our hand and get its landmarks and then use the points to draw on the screen. We will use two fingers for selection and one finger for drawing. And the best part is that all of this will be done in real-time.

- **HandTrackingModule.py**

```
"""
Hand Tracking Module
By: Murtaza Hassan
Youtube: http://www.youtube.com/c/MurtazasWorkshopRoboticsandAI
Website: https://www.computervision.zone
"""

import cv2
import mediapipe as mp
import time
import math
import numpy as np

class handDetector():
    def __init__(self, mode=False, maxHands=2, detectionCon=0.5, trackCon=0.5):
        self.mode = mode
        self.maxHands = maxHands
        self.detectionCon = detectionCon
        self.trackCon = trackCon

        self.mpHands = mp.solutions.hands
        self.hands = self.mpHands.Hands(self.mode, self.maxHands,
        self.detectionCon, self.trackCon)
```

```
        self.mpDraw = mp.solutions.drawing_utils
        self.tipIds = [4, 8, 12, 16, 20]

def findHands(self, img, draw=True):
    imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    self.results = self.hands.process(imgRGB)
    # print(results.multi_hand_landmarks)

    if self.results.multi_hand_landmarks:
        for handLms in self.results.multi_hand_landmarks:
            if draw:
                self.mpDraw.draw_landmarks(img, handLms,
                self.mpHands.HAND_CONNECTIONS)

    return img

def findPosition(self, img, handNo=0, draw=True):
    xList = []
    yList = []
    bbox = []
    self.lmList = []
    if self.results.multi_hand_landmarks:
        myHand = self.results.multi_hand_landmarks[handNo]
        for id, lm in enumerate(myHand.landmark):
            # print(id, lm)
            h, w, c = img.shape
            cx, cy = int(lm.x * w), int(lm.y * h)
            xList.append(cx)
            yList.append(cy)
            # print(id, cx, cy)
            self.lmList.append([id, cx, cy])
            if draw:
                cv2.circle(img, (cx, cy), 5, (255, 0, 255), cv2.FILLED)

    xmin, xmax = min(xList), max(xList)
    ymin, ymax = min(yList), max(yList)
    bbox = xmin, ymin, xmax, ymax

    if draw:
        cv2.rectangle(img, (xmin - 20, ymin - 20), (xmax + 20, ymax + 20),
        (0, 255, 0), 2)

    return self.lmList, bbox

def fingersUp(self):
    fingers = []
    # Thumb
    if self.lmList[self.tipIds[0]][1] > self.lmList[self.tipIds[0] - 1][1]:
        fingers.append(1)
    else:
        fingers.append(0)
```

```
84
85      # Fingers
86      for id in range(1, 5):
87          if self.lmList[self.tipIds[id]][2] < self.lmList[self.tipIds[id] - 2][2]:
88              fingers.append(1)
89          else:
90              fingers.append(0)
91
92          # totalFingers = fingers.count(1)
93
94      return fingers
95
96  def findDistance(self, p1, p2, img, draw=True,r=15, t=3):
97      x1, y1 = self.lmList[p1][1:]
98      x2, y2 = self.lmList[p2][1:]
99      cx, cy = (x1 + x2) // 2, (y1 + y2) // 2
100
101     if draw:
102         cv2.line(img, (x1, y1), (x2, y2), (255, 0, 255), t)
103         cv2.circle(img, (x1, y1), r, (255, 0, 255), cv2.FILLED)
104         cv2.circle(img, (x2, y2), r, (255, 0, 255), cv2.FILLED)
105         cv2.circle(img, (cx, cy), r, (0, 0, 255), cv2.FILLED)
106         length = math.hypot(x2 - x1, y2 - y1)
107
108     return length, img, [x1, y1, x2, y2, cx, cy]
109
110 def main():
111     pTime = 0
112     cTime = 0
113     cap = cv2.VideoCapture(1)
114     detector = handDetector()
115     while True:
116         success, img = cap.read()
117         img = detector.findHands(img)
118         lmList, bbox = detector.findPosition(img)
119         if len(lmList) != 0:
120             print(lmList[4])
121
122         cTime = time.time()
123         fps = 1 / (cTime - pTime)
124         pTime = cTime

            cv2.putText(img, str(int(fps)), (10, 70), cv2.FONT_HERSHEY_PLAIN, 3,
            (255, 0, 255), 3)

            cv2.imshow("Image", img)
            cv2.waitKey(1)

    if __name__ == "__main__":
        main()
```

- **VirtualPainter.py ->**

```
import numpy as np
import time
import os
import HandTrackingModule as htm

########################
brushThickness = 25
eraserThickness = 100
########################


folderPath = "Header"
myList = os.listdir(folderPath)
print(myList)
overlayList = []
for imPath in myList:
    image = cv2.imread(f'{folderPath}/{imPath}')
    overlayList.append(image)
print(len(overlayList))
header = overlayList[0]
drawColor = (255, 0, 255)

cap = cv2.VideoCapture(1)
cap.set(3, 1280)
cap.set(4, 720)

detector = htm.handDetector(detectionCon=0.65,maxHands=1)
xp, yp = 0, 0
imgCanvas = np.zeros((720, 1280, 3), np.uint8)

while True:

    # 1. Import image
    success, img = cap.read()
    img = cv2.flip(img, 1)

    # 2. Find Hand Landmarks
    img = detector.findHands(img)
    lmList = detector.findPosition(img, draw=False)

    if len(lmList) != 0:
```

```
        # print(lmList)

        # tip of index and middle fingers
        x1, y1 = lmList[8][1:]
        x2, y2 = lmList[12][1:]

        # 3. Check which fingers are up
        fingers = detector.fingersUp()
        # print(fingers)

        # 4. If Selection Mode - Two finger are up
        if fingers[1] and fingers[2]:
            # xp, yp = 0, 0
            print("Selection Mode")
            # # Checking for the click
            if y1 < 125:
                if 250 < x1 < 450:
                    header = overlayList[0]
                    drawColor = (255, 0, 255)
                elif 550 < x1 < 750:
                    header = overlayList[1]
                    drawColor = (255, 0, 0)
                elif 800 < x1 < 950:
                    header = overlayList[2]
                    drawColor = (0, 255, 0)
                elif 1050 < x1 < 1200:
                    header = overlayList[3]
                    drawColor = (0, 0, 0)
            cv2.rectangle(img, (x1, y1 - 25), (x2, y2 + 25), drawColor, cv2.FILLED)

        # 5. If Drawing Mode - Index finger is up
        if fingers[1] and fingers[2] == False:
            cv2.circle(img, (x1, y1), 15, drawColor, cv2.FILLED)
            print("Drawing Mode")
            if xp == 0 and yp == 0:
                xp, yp = x1, y1

            cv2.line(img, (xp, yp), (x1, y1), drawColor, brushThickness)

            # if drawColor == (0, 0, 0):
            #     cv2.line(img, (xp, yp), (x1, y1), drawColor, eraserThickness)
            #     cv2.line(imgCanvas, (xp, yp), (x1, y1), drawColor, eraserThickness)
            #
            # else:
            #     cv2.line(img, (xp, yp), (x1, y1), drawColor, brushThickness)
            #     cv2.line(imgCanvas, (xp, yp), (x1, y1), drawColor, brushThickness)

            xp, yp = x1, y1
```

```
        # # Clear Canvas when all fingers are up
        # if all (x >= 1 for x in fingers):
        #     imgCanvas = np.zeros((720, 1280, 3), np.uint8)

    imgGray = cv2.cvtColor(imgCanvas, cv2.COLOR_BGR2GRAY)
    _, imgInv = cv2.threshold(imgGray, 50, 255, cv2.THRESH_BINARY_INV)
    imgInv = cv2.cvtColor(imgInv,cv2.COLOR_GRAY2BGR)
    img = cv2.bitwise_and(img,imgInv)
    img = cv2.bitwise_or(img,imgCanvas)


    # Setting the header image
    img[0:125, 0:1280] = header
    # img = cv2.addWeighted(img,0.5,imgCanvas,0.5,0)
    cv2.imshow("Image", img)
    cv2.imshow("Canvas", imgCanvas)
    cv2.imshow("Inv", imgInv)
    cv2.waitKey(1)
```

- **Output-**

To test the whiteboard, run main.py .

It uses the different functions of OpenCV available in Python. It takes an image from the camera and detects the contour in the image. Starts drawing on the canvas as contour moves.

1. It turns on the camera and starts detecting the contour in the camera.

2. It draws a circle at the tip of the contour.

3. The color of the circle is the same as the color of your contour or object.

4. It stores the points in which you move your object.

5. Now, it starts drawing a circle to all the points which are stored.

6. You can add the different minimum and maximum HSV values for different colors.

7. Also, we have to add corresponding BGR color values.

8. Then you will be able to draw with the different color objects.

8. You have to take care of the background.

A. Clenching a fist with your index finger sticking out allows you to draw.

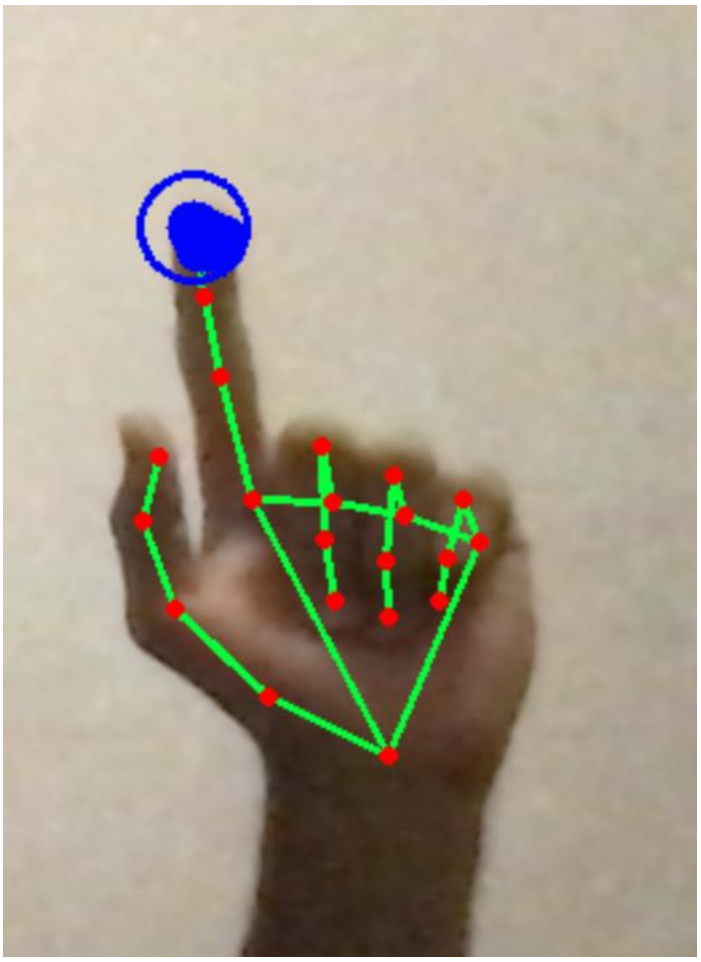


Fig 19

B. Closing your hand into a fist allows you to erase, where the red box represents the bounds of the eraser.

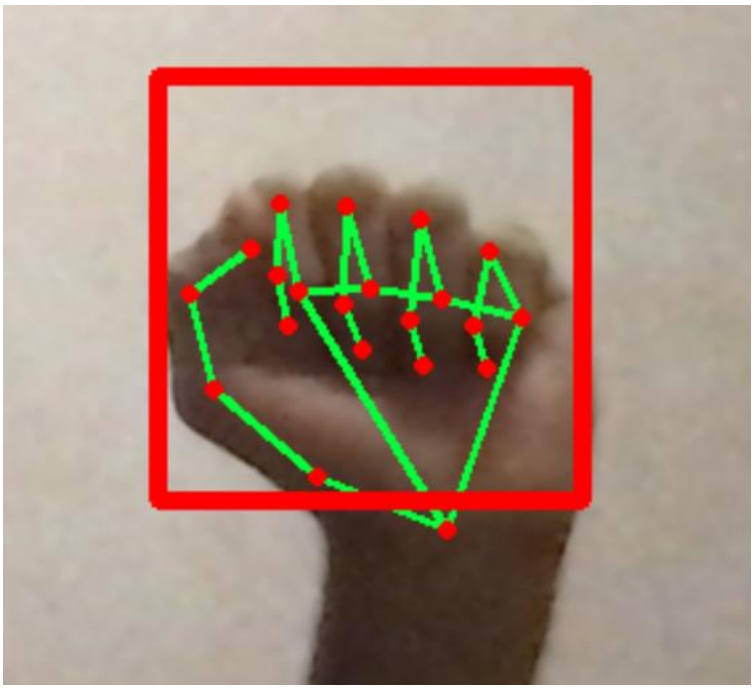


Fig 20

C. Holding your hand with all of your fingers open does nothing.

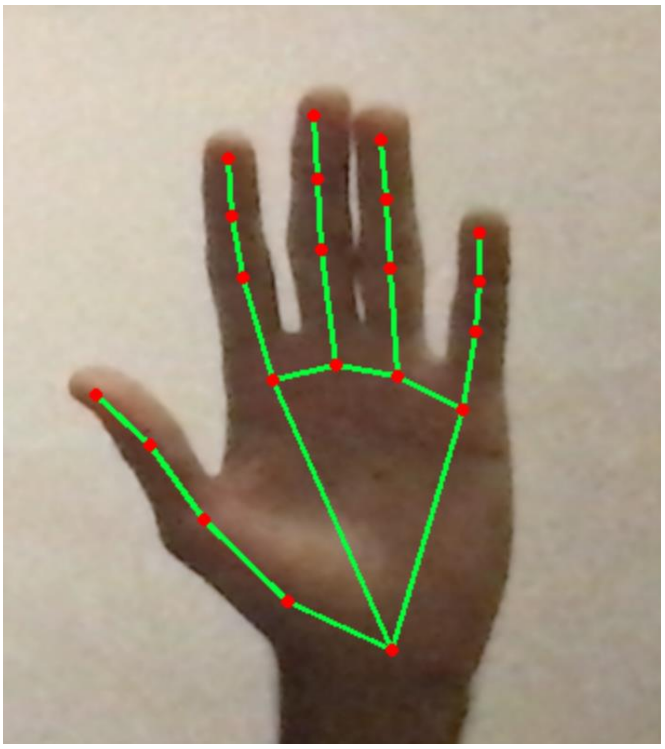


Fig 21

Finally, this implementation has come up with an idea for this project, how one can reduce the chances of spread of contagious virus or diseases that is beneficial for the society with the help of open-source networks i.e., OpenCV and Jupyter Notebook with the help of object detection. This project can be a very helpful project for the society for the pandemic situations. It was a simple illustration of image processing capabilities of OpenCV.

Figure shows a screenshot of the developed application. It should be noted that the writing of information takes place in the air without physical contact with the computer screen. The application interface is a white screen, a simple HTML page that makes testing easier. This work allows us to conclude that the future of technologies is in their combination. For example, human-computer interaction can be improved by a combination of machine learning techniques, VR devices, and web services. This makes the computer more responsive to user needs for new tasks which was demonstrated in the application using the Leap Motion virtual reality device

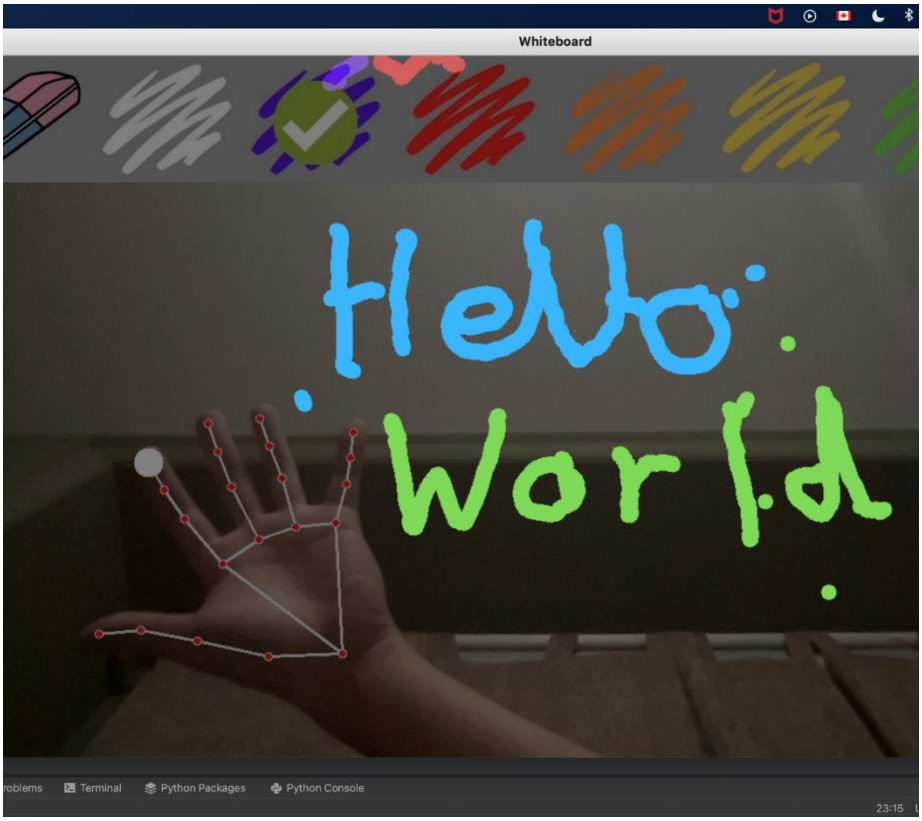


Fig 22

7. Conclusion-

This work allows us to conclude that the future of technologies is in their combination. For example, human-computer interaction can be improved by a combination of machine learning techniques, VR devices, and web services. This makes the computer more responsive to user needs for new tasks which was demonstrated in the application using the Leap Motion virtual reality device.

The article is devoted to the development of a virtual reality application that allows you to use a personal computer as a whiteboard. As a VR device, a Leap Motion motion controller is used, the principle of which is based on capturing hand movements for humancomputer interaction. Leap Motion is limited to a 3D userspace [11]. The scope of the device is determined by IR diodes emitting light to the workspace.

When the hand is in this area, the reflected light rays enter the controller's IR cameras, after which the captured stereographic image is transmitted to the controller's software processor. After the algorithms of computer vision, the contours of a person's hands are distinguished, thereby determining the positions and coordinates of fingers, bones, and wrists. experience to understand complex topics and work with real installations [9].

Very often VR is used in the entertainment industry, especially in video games and movies. The first virtual reality headsets available to a wide range of users were released by video game companies in the early to mid1990s. But since the 2010s, Oculus (Rift), HTC (Vive), and Sony (PlayStation VR) have released headsets even more affordable, starting a new wave of app development.

- User Manual: A complete document (Help Guide) of the software developed**.**

Machine learning is a branch of Artificial Intelligence which is used to analyze the data more smartly. It automates the process using certain algorithms to minimize human intervention in the process.

To implement this work, one should first learn object tracking [1] and one should be aware of OpenCV [2] and Jupyter Notebook. Using Object tracking we detect the HSV code [3] that is similar like RGB color code [4]. HSV is Hue, Saturation and Value respectively.

Typically tracking algorithms [5] are quicker than detection algorithms. We know a lot about the objects appearance when we are monitoring an object which was observed in the previous frame, we already know the position in the next frame [6], we can use all this data to determine the position of object in the next frame, and to precisely find the object with a limited scan around the estimated location of the object. In the other hand, a successful monitoring algorithm can manage any amount of obstruction. The motion model estimates the objects precise location. To have a more precise assessment based on appearance, the appearance model makes improvements on this assessment.

The project has been created to help people understand the complete process of machine learning / data science modeling.

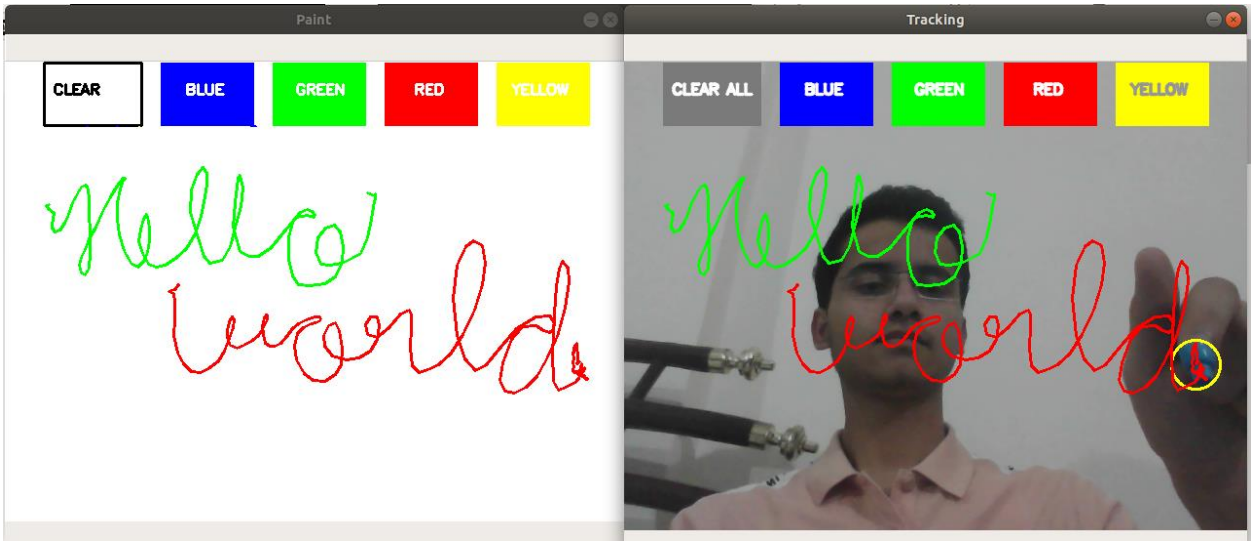


Fig 23

7. Future scope –

Virtual reality (VR) implies a simulated environment, interaction with which is carried out using special hardware and software, based on signals received by a person through the senses [1,2]. Virtual reality is characterized by the properties of interactivity and immersivity, which implies complete immersion of the user in the virtual environment and autonomy, that is, the independence of VR from the real world [2]. To date, virtual reality technology has found application in many areas of human activity. In medicine, simulated virtual reality surgical environments serve as an environment for effective teaching and surgical operations [3, 4]. The concept of computational surgery makes it possible to simulate complex structures in 3D, which facilitates the planning and execution of procedures [5]. In addition to planning operations, VRs are used directly during the operations themselves, so doctors can easily access the necessary material quickly, affordably, and safely [6]. Virtual reality is used for education in many industries where the operation of real devices and mechanisms is associated with increased risk or high costs while providing students with a safer work environment for learning skills. Methods for introducing virtual reality technology are being studied in a variety of educational institutions [7]. So, for example, the space agency NASA, as a result of several years of research, has successfully used VR for preparing astronauts [8]. VR has also proven useful for technical educators and their students. First, due to the possibility of replacing expensive equipment with its virtual simulator. One of the important elements is the interaction of students with threedimensional models, which accurately respond to input, thanks to simulation, repeating the reaction of real objects. This additional tool provides the user with the necessary.

- **Project Legacy –**

  Computer vision is a science of teaching computers to see. With the state of the art algorithms, this technology is behind many applications like self-driving cars, image recognition, medical diagnosis etc. The best part of computer vision is these techniques are used for detecting cancerous cells which helps in saving lives by adding filters to your face to entertain you. Similarly, it has a wide variety of interesting and useful applications.

In the current pandemic situation, we have seen one of the reasons for spreading of corona virus is surface touching, and to avoid this sort of spread in the virus we have come up with a project which is titled as "Drawing on Air", As the title itself indicates, we are not going to touch the surface but we are going to draw on air and what we have drawn on air will be displayed on the screen / monitor. This will help us reduce the spread of the virus in the pandemic situations. In this pandemic we have seen maximum of the transactions are been made online but for deposit of money or for transfer of money in person we see people use ATMs which is very risky and there are chances of spreading of the virus at such point of times we can use this as a prevention method to a great extent. Not only in ATMs but also in offices for biometrics or in cyber cafes and many more areas we can use this project.

Keywords: Feature extraction; Image processing; Open CV; Python; Virtual classroom; Virtual white board.

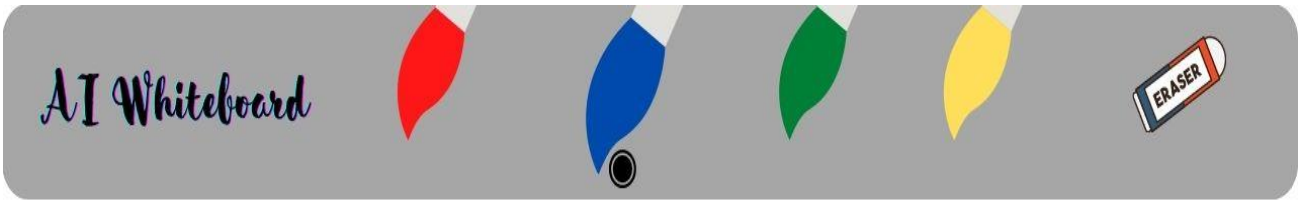


Fig 24

8. References

- **Ministry of Health, Labour and Welfare**, ‘‘The survey of difficulties of daily life (survey of nationwide home children with disabilities, and others, 2011),’’ Dept. Health Welfare Persons Disabilities, Tokyo, Japan, Tech. Rep., 2013.
  Google Scholar

- **National Library of Medicine** , “An implementation of virtual white board using open CV for virtual classes” , **An implementation of virtual white board using open CV for virtual classes - PMC (nih.gov)**

- **Murtaza Hassan,** AI Virtual Painter **,** AI Virtual Painter - Computer Vision Zone

- **T. Murata, J. Shin**
  Hand gesture and character recognition based on kinect sensor
  Int. J. Distrib. Sensor Netw., 10 (7) (2014),
  View Record in Scopus Google Scholar

- **Viraj lakshitha Bandara ,** ai-whiteboard-using-opencv **,** viraj-lakshitha/ai-whiteboard-using-opencv (github.com)

- **Santha Lakshmi Narayana ,** Whiteboard image enhancement using OpenCV , Whiteboard Image Enhancement using OpenCV- Santha Lakshmi Narayana

- **Y. Tanaka**, Training system for learning finger alphabets with feedback functions, M.S. thesis, Faculty Ind. Technol., Tsukuba Univ. Technol., Tsukuba, Japan, 2014.
  Google Scholar

- **Prudhvi varma, Virtual Paintaing App Using OpenCV,** How To Create A Virtual Painting App Using OpenCV? (analyticsindiamag.com)

- **M. Maehatake, M. Nishida, Y. Horiuchi, A. Ichikaw**a, A study on sign language recognition based on gesture components of position and movement, in Proc. Workshop Interact. Syst. Softw. (WISS), Japan, Dec. 2007, pp. 129–130.
  Google Scholar

- **Taha Anwar** (BleedAI.com), Creating a Virtual Pen And Eraser with OpenCV ,
  Creating a Virtual Pen & Eraser with OpenCV | LearnOpenCV

- **A. Sato, K. Shinoda, S. Furui**
  'Sign language recognition using time-of-flight camera (in Japanese)
  Proc. Meeting Image Recognit. Understand. (MIRU), 3 (44) (2010), pp. 1861-1868
  View Record in Scopus Google Scholar

- **Dhairya Kumar , Introduction to OpenCV ,** Introduction to OpenCV - GeeksforGeeks

- **D. Takabayashi, *et al.***
  Training system for learning finger alphabets with feedback functions ,IEICE (HIP) , Tech. Rep., 112 (483) (2013), pp. 79-84
  View Record in Scopus Google Scholar

9. Appendices –

APPENDIX

This Desktop-based software runs at Microsoft's Windows operating system environments using image-processing technology from OpenCV and python programming language. Virtual Whiteboard software is expected to facilitate and contribute new in the world of technology, especially image processing and the interaction between humans and computers. This Project is implemented with the aim of building and developing software that can replace the using of interactive multimedia board. The Software is called a Virtual Whiteboard, that used by using a USB camera as video sensor, infrared pointer as a pointer coordinates, and a projector or monitor as a media for interaction between user and computer.

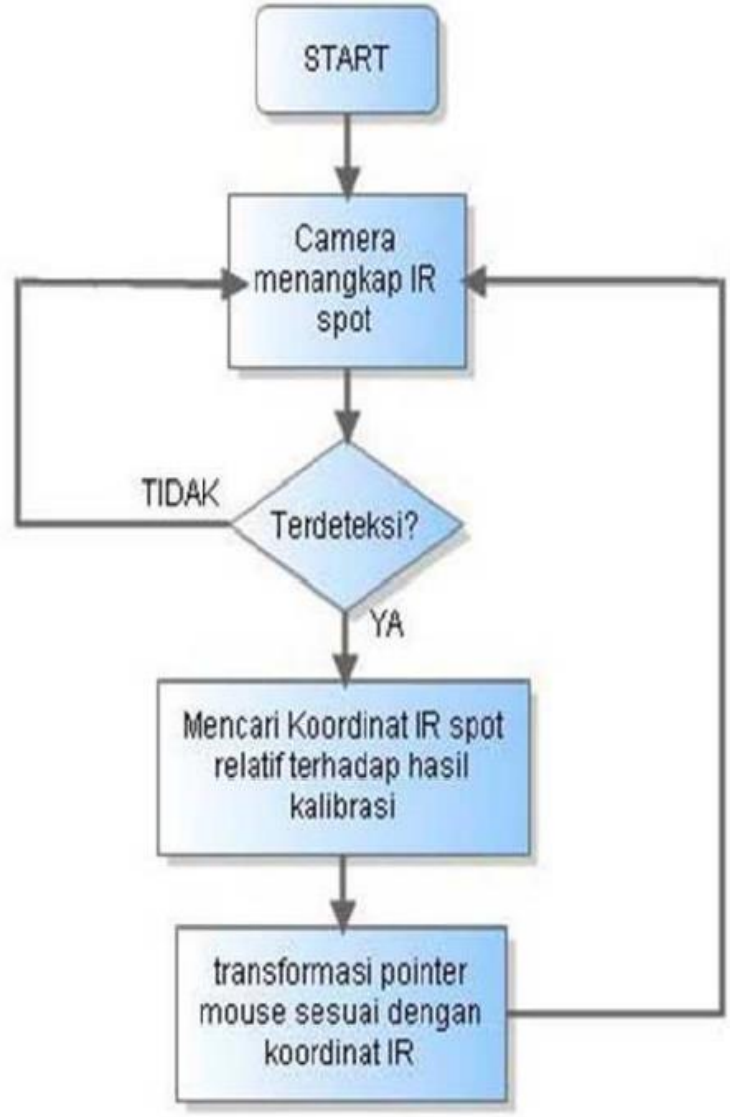


Fig 25 System Flowchart:

Fig 26. Virtual drawing board ✍️

Publication of paper on project Work (attach abstract and certificate)

10. Biodata of Projectees

Ankit Mehta, currently Pursuing Btech CSE from Lovely Professional Punjab,

Passionate about software development, really obsessed with staying organized, love competitive programming. Dynamic, energetic, and extremely motivated professional with a passion for driving tough issues to completion. As a team member will go above and beyond to help peers. An experienced folk was willing to leap into deep technical challenges at any stage with the flexibility to know when to get help as well as to when to stand out of the way.