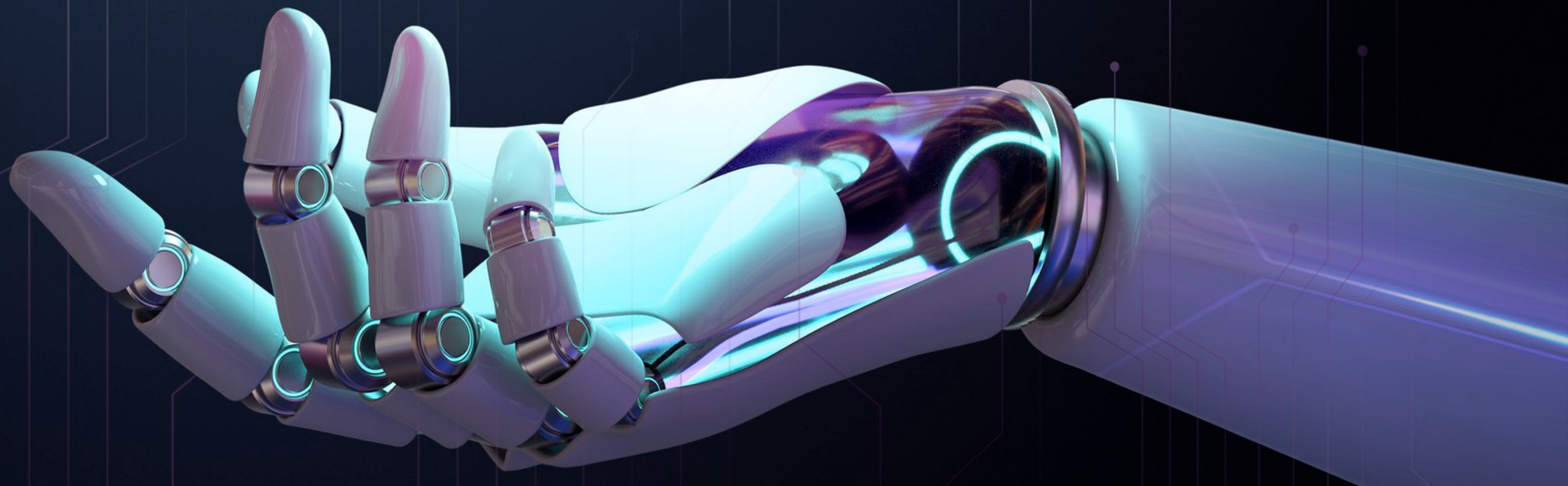
# HAND GESTURE CONTROLLED PRESENTATION




Team 5

Anihant Gadi

Navyasree R

Saatwik Alle

# INTRODUCTION

Hand gesture control presentation project using MediaPipe and OpenCV, we aim to revolutionize the dynamics of presentations by implementing an interactive interface based on hand gestures. By integrating MediaPipe and OpenCV technologies, the system interprets and responds to a diverse range of natural hand movements in real-time. Unlike conventional slide management methods, this project focuses on providing users with a seamless and user-friendly approach to control presentations. Through machine learning and computer vision, it enables dynamic slide navigation, content emphasis, and animation triggers, offering a novel and engaging experience. The utilization of advanced gesture recognition libraries ensures not only efficient slide control but also allows users to personalize interactions by creating custom gestures for specific actions. This project is not just about functionality; it strives to redefine audience engagement, fostering an interactive and immersive presentation environment.

# ABSTRACT

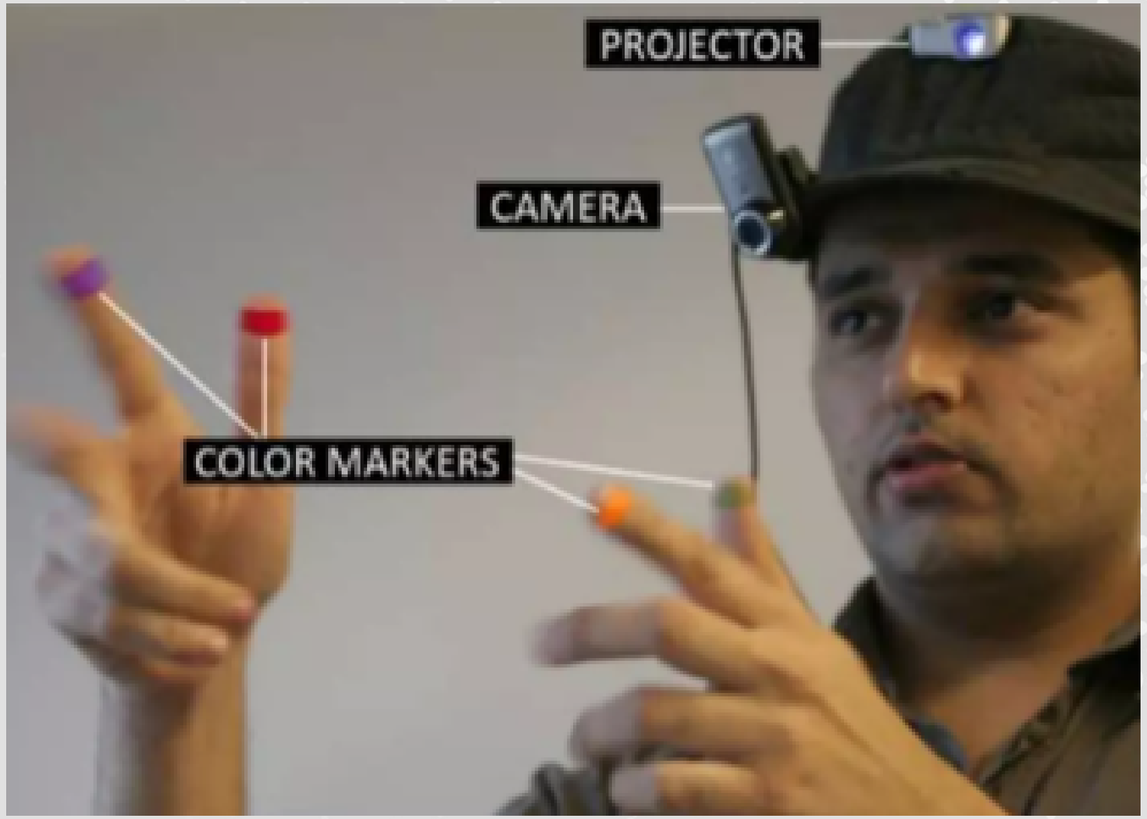
The Hand Gesture-Controlled PPT project redefines presentation dynamics by employing innovative gesture-based control, transcending traditional slide management. It represents a fusion of cutting-edge technologies like machine learning and computer vision, decoding an extensive range of hand movements. Liberating presenters from conventional input devices, it aims to elevate the presentation experience through effortless slide navigation, dynamic content emphasis, and intuitive animation triggers using natural hand gestures. This project embodies inclusivity and accessibility, empowering individuals with disabilities or limited mobility to independently deliver impactful presentations. Leveraging advanced gesture recognition libraries such as OpenCV and MediaPipe, it not only transforms slide control but also encourages personalized interactions, enabling users to create custom gestures for specific actions. Beyond functionality, this innovation redefines audience engagement, fostering an interactive and immersive presentation atmosphere that captivates diverse audiences.

# EXISTING SYSTEM

1.Hardware Requirements: The existing system may require specific hardware components, such as a depth camera or a motion sensor, to track hand gestures accurately.

2.Limited Gesture Set: The existing system might have a limited set of recognized hand gestures for controlling the PowerPoint presentation, restricting the user's options for interaction.

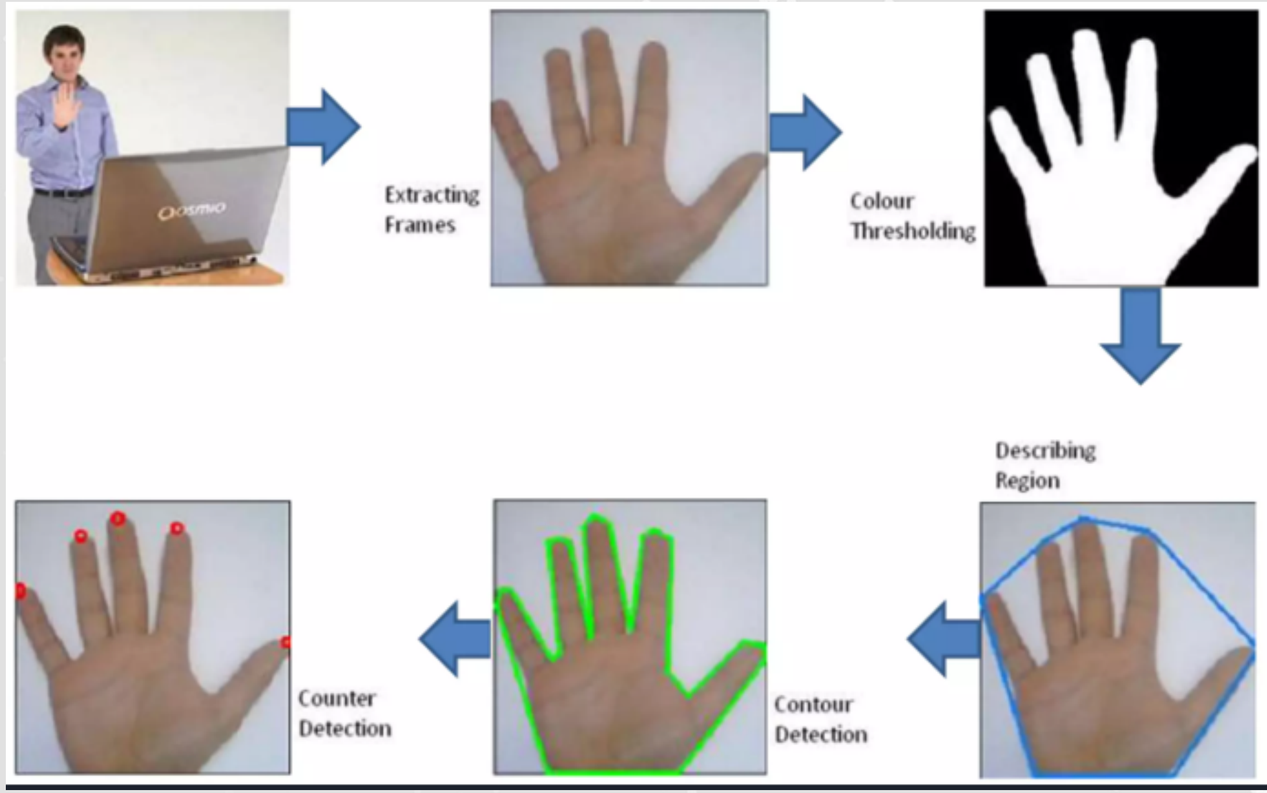
3.Glove based: Requires use of hand gloves and variable glove size for users.



# PROPOSED SYSTEM

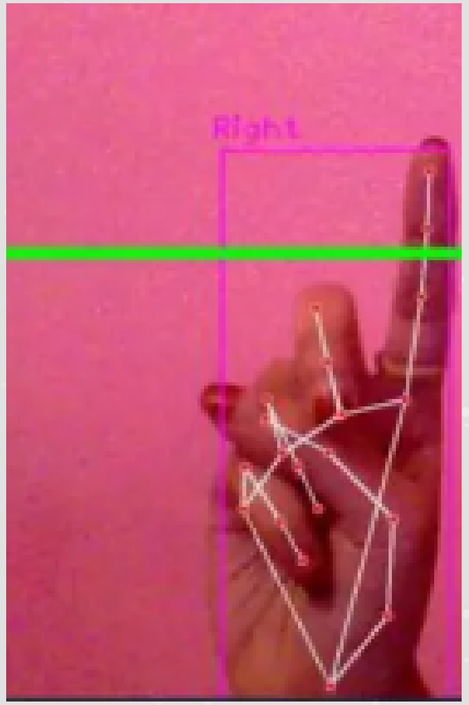
1.Enhanced Gesture Recognition: The proposed system could employ advanced computer vision techniques or machine learning algorithms to accurately recognize a broader range of hand gestures, providing more precise control over the PowerPoint presentation.

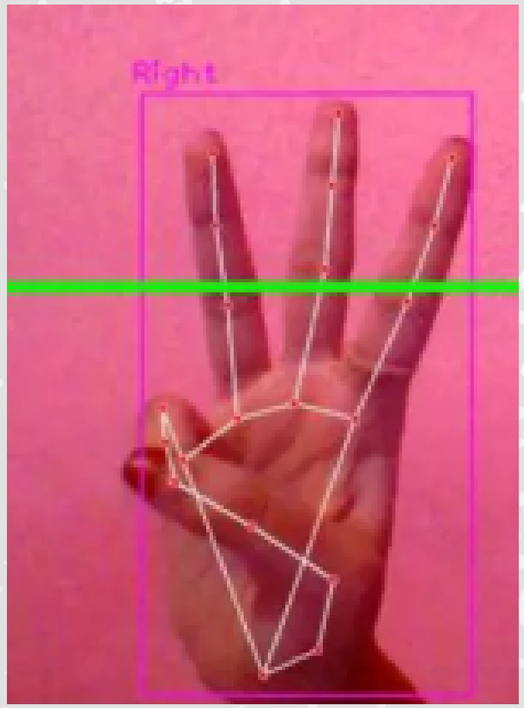
2.Customizable Gesture Set: The proposed system might allow users to define and customize their own hand gestures for specific actions, offering a more personalized and adaptable interaction



# PROPOSED SYSTEM

3.Additional Functionality: The proposed system may introduce additional functionalities beyond basic slide navigation, such as zooming in or out on slides, highlighting specific content, or activating embedded media elements.



# CODE

```
import os
import cv2
from cvzone.HandTrackingModule import HandDetector
import numpy as np
#variables
width,height=1280,720
folderPath="PPT SLIDES"
#Camera Setup
cap=cv2.VideoCapture(0,cv2.CAP_DSHOW)
cap.set(cv2.CAP_PROP_FRAME_WIDTH,1280)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT,720)
cap.set(3,width)
cap.set(4,height)
# get the list of presentation images
pathImages=sorted(os.listdir(folderPath),key=len) #key=len means sorting according to length
```

# CODE

```
print(pathImages) # ['Slide1.PNG', 'Slide2.PNG', 'Slide3.PNG', 'Slide4.PNG', 'Slide5.PNG', 'Slide6.PNG', 'Slide7.PNG', 'Slide8.PNG', 'Slide9.PNG', 'Slide10.PNG']
# variables
imgNumber= 0 # for going front and back slides
heightsmall , widthsmall= int(120*1),int(213*1) # dividing width,height=1280,720 by 6 each, we get 120,213 height,width such that it is the height,width for the webcam on the slide on top write(not accurately)
gestureThreshold=300 # if values is above 300 then detect the hand
buttonPressed=False #used to slow down the movement of the slides slowly insteadly of rapidly changing
buttonCounter=0
buttonDelay=30
draw= [[]]
drawNumber=-1
drawStart=False
```

# CODE

```
#Hand Detector
detector= HandDetector(detectionCon=0.8,maxHands=1) # detectionCon=0.8 implies if 80% confidence that it is hand then work
while True:
 #import images
 success,img=cap.read()
 img=cv2.flip(img,1) # flip is used that if right hand moves towards left in real life then in webcam it must move towards left only not right , 1 -> horizontal flipping,0-> vertical flipping
 pathFullImg=os.path.join(folderPath,pathImages[imgNumber]) # pathFullImg shows us the current slide imgNumber is pointing from pathImages var which is sorted
 imgCurrent= cv2.imread(pathFullImg)
 print(drawNumber)
# Find the hand and its landmarks
    hands,img=detector.findHands(img)
 # flipType=
```

# CODE

```
# Draw Gesture Threshold line
    cv2.line(img,(0,gestureThreshold),(width,gestureThreshold),(0,255,0),5)
     if hands and buttonPressed is False: # if hand is detected in the webcam
        hand=hands[0] # 0 means only 1 hand is detected as maxHands=1
        fingers=detector.fingersUp(hand) # List of which fingers are up
        cx,cy=hand["center"]
        #print(fingers)
        lmList=hand["lmList"] # List of 21 Landmark points
        #constrain values for easier drawing
        #indexFinger=lmList[8][0],lmList[8][1] # 8-> indexfinger number ,0,1 are the two points on the
fingers
        xVal=int(np.interp(lmList[8][0],[width//2, width],[0,width])) # converting [width//2, wslide]->
[0,width] such that the pointer willl be at the right half of the webcam for easy usage
        yVal =int(np.interp(lmList[8][1],[150,height-150],[0,height]))
        indexFinger=xVal,yVal
```

# CODE

```
 if cy<=gestureThreshold: # if hand is at the height of the face or above the gesture line
            # Gesture 1 -left
            if fingers==[1,0,0,0,0]:
                print('left')
                buttonPressed = True #
                if imgNumber>0:
                    draw = [[]]   #
                    drawNumber = -1 # the draw,drawNumber,drawStart is declared again such that if we draw in current slide, then me move to next slide the drawing will get erased
                    drawStart = False
                    imgNumber = imgNumber-1
           # Gesture 2 -right
            if fingers == [0, 0, 0, 0, 1]:
                print('right')
                buttonPressed = True
```

# CODE

```
    if imgNumber <len(pathImages)-1:
            draw = [[]]
            drawNumber = -1
            drawStart = False
            imgNumber = imgNumber +1
# Gesture 3 - Show Pointer
    if fingers==[0,1,1,0,0]:
        cv2.circle(imgCurrent,indexFinger,12,(0,0,255),cv2.FILLED)

# Gesture 4 - Drawing Pointer
    if fingers==[0,1,0,0,0]:
        if drawStart is False:
            drawStart=True
            drawNumber=drawNumber+1
            draw.append([])
```

# CODE

```
        print(drawNumber)
        draw[drawNumber].append(indexFinger)
        cv2.circle(imgCurrent, indexFinger, 12, (0, 0, 255), cv2.FILLED)
    else:
        drawStart=False
    # Gesture 5 - UNDO
    if fingers == [0, 1, 1, 1, 0]:
        if draw:
                draw.pop(-1)
                drawNumber = drawNumber - 1
                buttonPressed=True
else: # if no hand
    drawStart=False
```

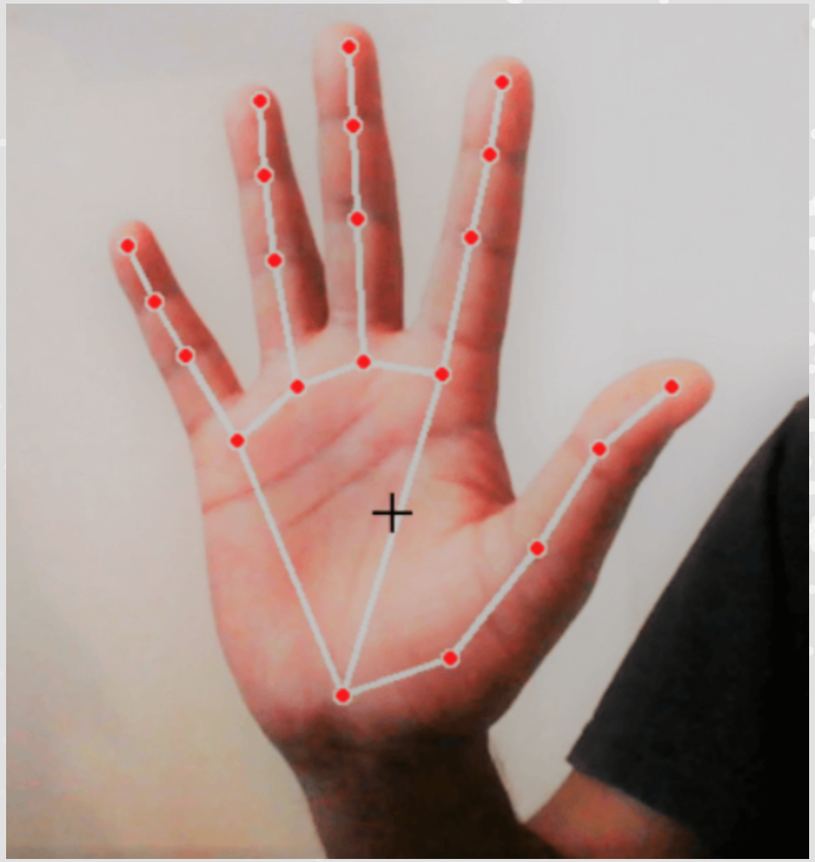
# CODE

```
 # Button Pressed Iterations
  if buttonPressed:
     buttonCounter=buttonCounter+1
     if buttonCounter>buttonDelay:
        buttonCounter=0
        buttonPressed=False
  for i in range(len(draw)):
     for j in range(len(draw[i])):
        if j!=0:
           cv2.line(imgCurrent,draw[i][j-1],draw[i][j],(0,0,200),12,) # from draw[i-1]-> to draw[i] the
line must be drawn
  #Adding Webcam image on the slides
  imgSmall=cv2.resize(img,(widthsmall,heightsmall))
  hslide,wslide,channel=imgCurrent.shape # gives height,width of slides
```

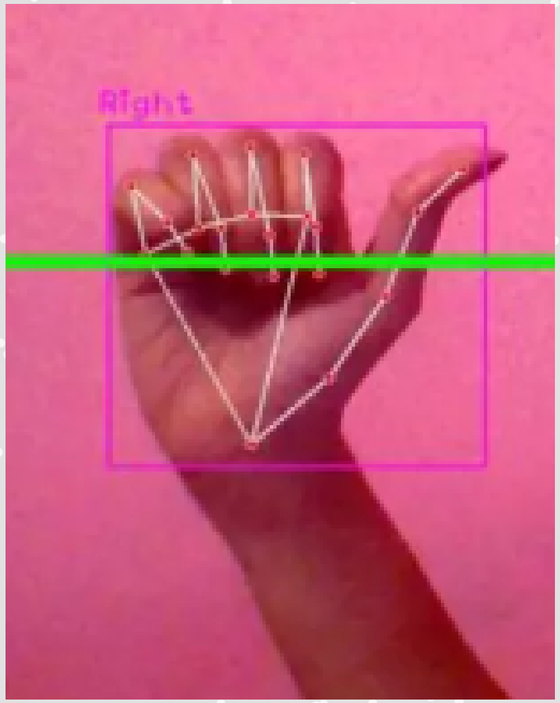
# CODE

imgCurrent[0:heightsmall,wslide-widthsmall:wslide]=imgSmall # putting the webcam on the slide in the top right conner

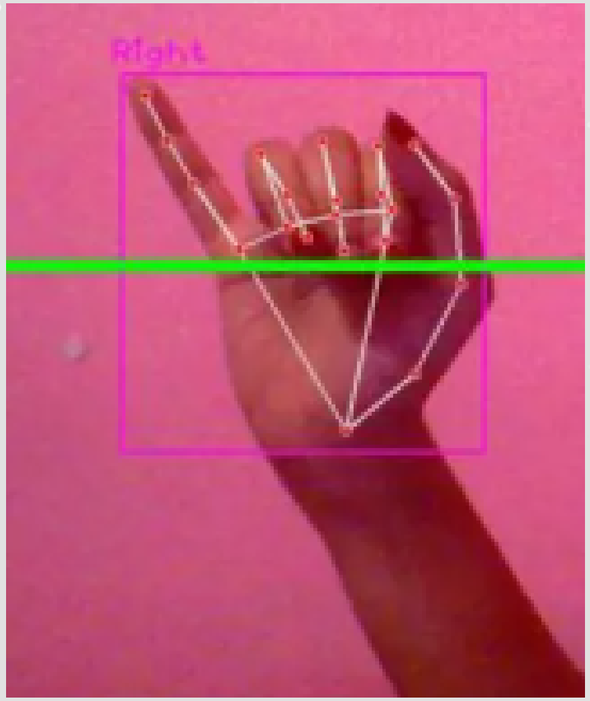
```
cv2.imshow("window",img)
cv2.imshow("Slides", imgCurrent)

key=cv2.waitKey(1)
if key==ord('q'):
  break
```



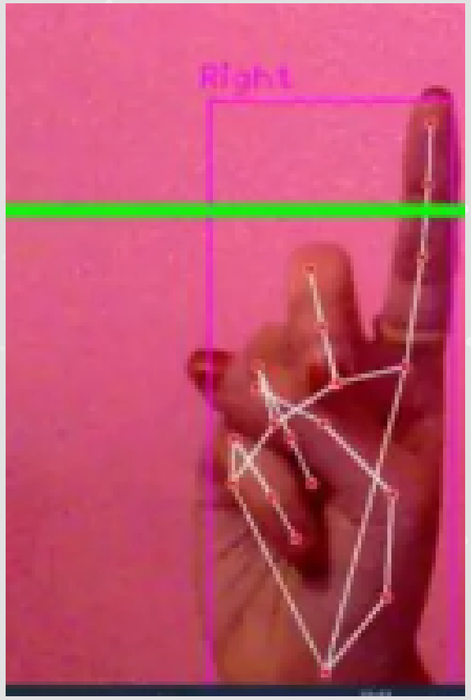
# EXECUTION



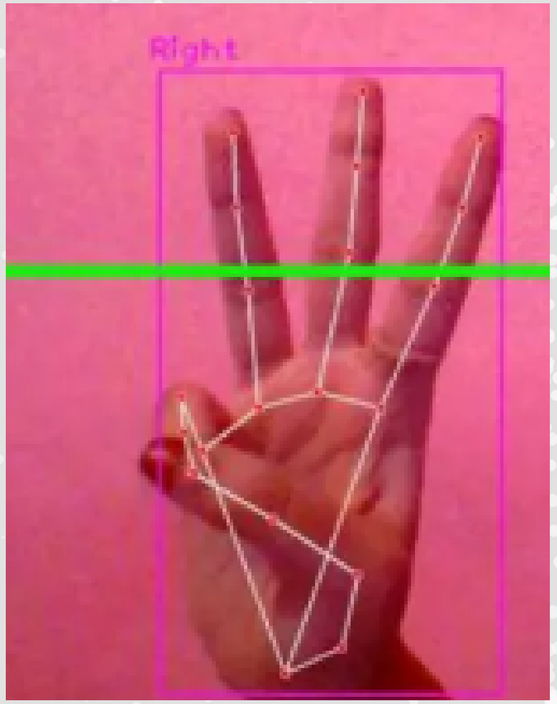

Previous Slide




Next Slide




Draw using pointer




Erase the drawing

# LITERATURE SURVEY

| S.no | Title of the paper | Journal name with year of publication | Authors | Methodology or algorithm used | Advantages | Drawbacks |
|---|---|---|---|---|---|---|
| 1 | Power point control using hand gesture recognition based on hog feature extraction and k-nn classification | ICCMC, 2017 | Tejashree P D Bharkad | The system takes the input data from the portable webcam consisting of four hand gestures. The processed image is then compared with the database of gesture images. Image is compared and recognized using K-nearest neighbor algorithm. | Enhanced accuracy. | Dependency on light in the background. |

# LITERATURE SURVEY

| S.no | Title of the paper | Journal name with year of publication | Authors | Methodology or algorithm used | Advantages | Drawbacks |
|---|---|---|---|---|---|---|
| 2 | Controlling PowerPoint Presentation using Hand Gestures in Real-Time | ICOEI, 2022 | Sai Charan Meenakshi Bhavani Kashyap | This research is to control the power point using Hand Gestures and as well as convert the presenter's speech signal into text. Python interface is used with the Microsoft power point that gives the presenter a lot of flexibility in controlling the slides of power point with hand gestures. | Fast translation in real time. | Limited gestures |

# CONCLUSION

The Hand Gesture-Controlled PPT project stands as a testament to technological ingenuity, reshaping the landscape of presentations with an elegant fusion of cutting-edge innovation and inclusivity. Its revolutionary approach, utilizing sophisticated machine learning and computer vision, transcends the bounds of traditional slide management, offering a seamless, intuitive interface driven by natural hand gestures. By liberating presenters from the constraints of conventional input devices, this project not only empowers individuals with disabilities or limited mobility but also creates an immersive, interactive experience that captivates diverse audiences. It symbolizes the harmonious marriage of technology and accessibility, redefining the very essence of presentation dynamics into a captivating and inclusive art form.

# REFERENCES


https://www.slideshare.net/irjetjournal/smart-presentation-control-by-hand-gestures-using-computer-vision-and-googles-mediapipe

https://ieeexplore.ieee.org/document/8282654

https://www.slideserve.com/edwardmoore/hand-gesture-recognition-powerpoint-ppt-presentation

https://ieeexplore.ieee.org/document/10125869

https://learningmsp430.wordpress.com/2014/11/16/gesture-control-for-powerpoint-presentation/

# THANK YOU

Anihant Gadi 21071A7207
R Navyasree 21071A7259
Saatwik Alle 21071A7261