Report on

Hand Gesture Controlled PowerPoint System

By

Saatwik Alle

1. Introduction

1.1 Background

Presentations, being a cornerstone of communication in various professional, academic, and public settings, demand innovative solutions for seamless control and engagement. Traditional input devices often fall short in meeting the dynamic needs of presenters, prompting the exploration of novel methods. The proposed hand gesture-controlled PowerPoint system emerges from this necessity, aiming to redefine the presenter's experience by introducing an interactive and intuitive control mechanism.

1.2 Objectives

The core objectives of the system encompass not only enhancing the presenter's control over their slides but also fostering a more inclusive and accessible presentation environment. By removing the reliance on physical input devices, the system empowers presenters to deliver their content more freely. Additionally, the focus on personalization and accessibility ensures that individuals with mobility impairments can participate in the presentation process with greater ease and independence.

2. Abstract

2.1 System Overview

The hand gesture-controlled PowerPoint system employs a multifaceted approach, integrating advancements in gesture-based human-computer interaction. At its core, the system utilizes machine learning algorithms to decipher and map hand movements accurately. This not only facilitates smoother slide control but also opens avenues for innovative gestures, enhancing the overall presentation experience

2.2 Technologies Utilized

The technological backbone of the system is comprised of various components. Machine learning algorithms, including deep learning models, contribute to robust hand gesture recognition. Libraries such as OpenCV and MediaPipe provide essential tools for implementing computer vision techniques. Furthermore, APIs from popular presentation software, such as Microsoft PowerPoint, serve as conduits for seamless integration.

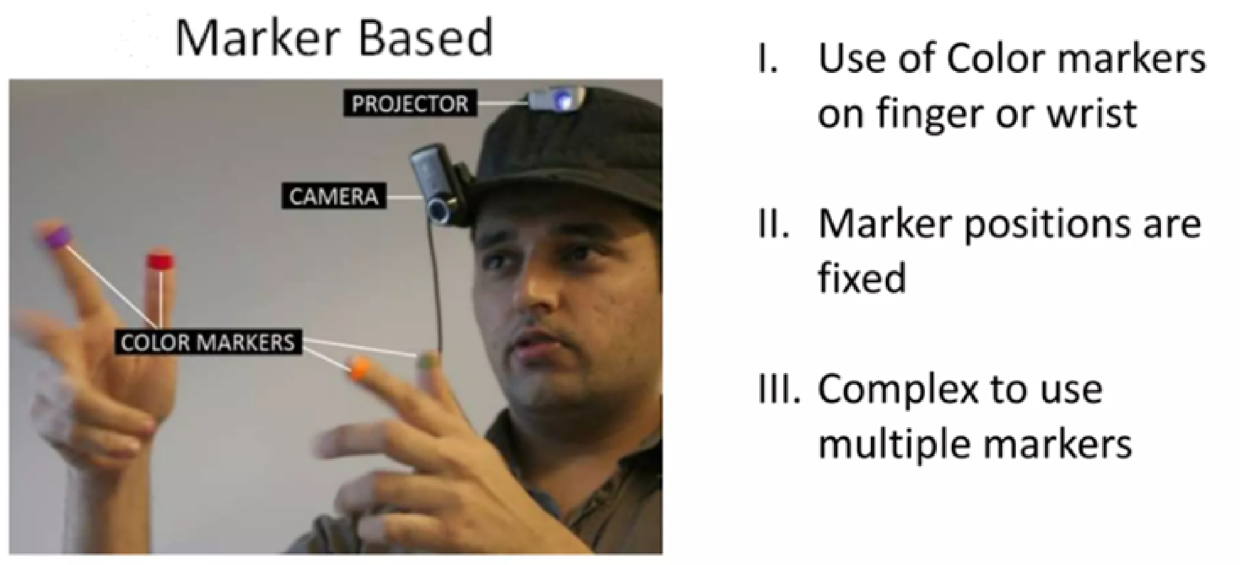
3. Existing System

3.1 Limitations of Current Systems

The limitations inherent in existing systems are multifaceted. Specific hardware requirements, such as depth cameras or motion sensors, can pose logistical challenges. Moreover, the reliance on a predefined set of gestures limits the spectrum of user interaction, hindering the potential for a truly intuitive control mechanism.

3.2 Gesture Recognition Challenges

Gesture recognition in current systems faces challenges related to predefined patterns and potential hardware dependencies. These challenges underscore the necessity for a more sophisticated solution capable of recognizing a broader array of gestures with a higher degree of accuracy.


PROJECTOR
CAMERA
COLOR MARKERS

Marker Based

I. Use of Color markers on finger or wrist

II. Marker positions are fixed

III. Complex to use multiple markers

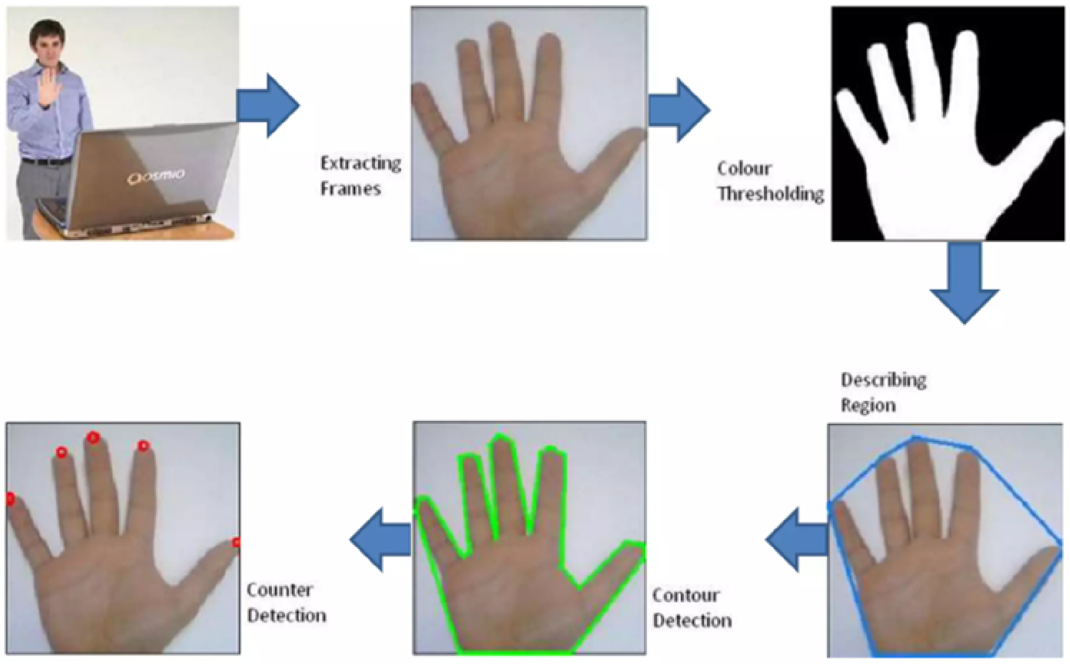
4. Proposed System

4.1 Enhanced Gesture Recognition

The proposed system introduces advancements in gesture recognition through the application of advanced computer vision techniques and machine learning algorithms. By leveraging deep learning models, the system endeavors to accurately recognize a diverse range of hand gestures, allowing presenters to express their intentions with precision.

4.2 Improved Accuracy and Responsiveness

Recognizing the critical importance of real-time responsiveness, the proposed system places a strong emphasis on refining accuracy. By mitigating false positives and negatives, the system aims to create a responsive and reliable interaction platform for presenters.

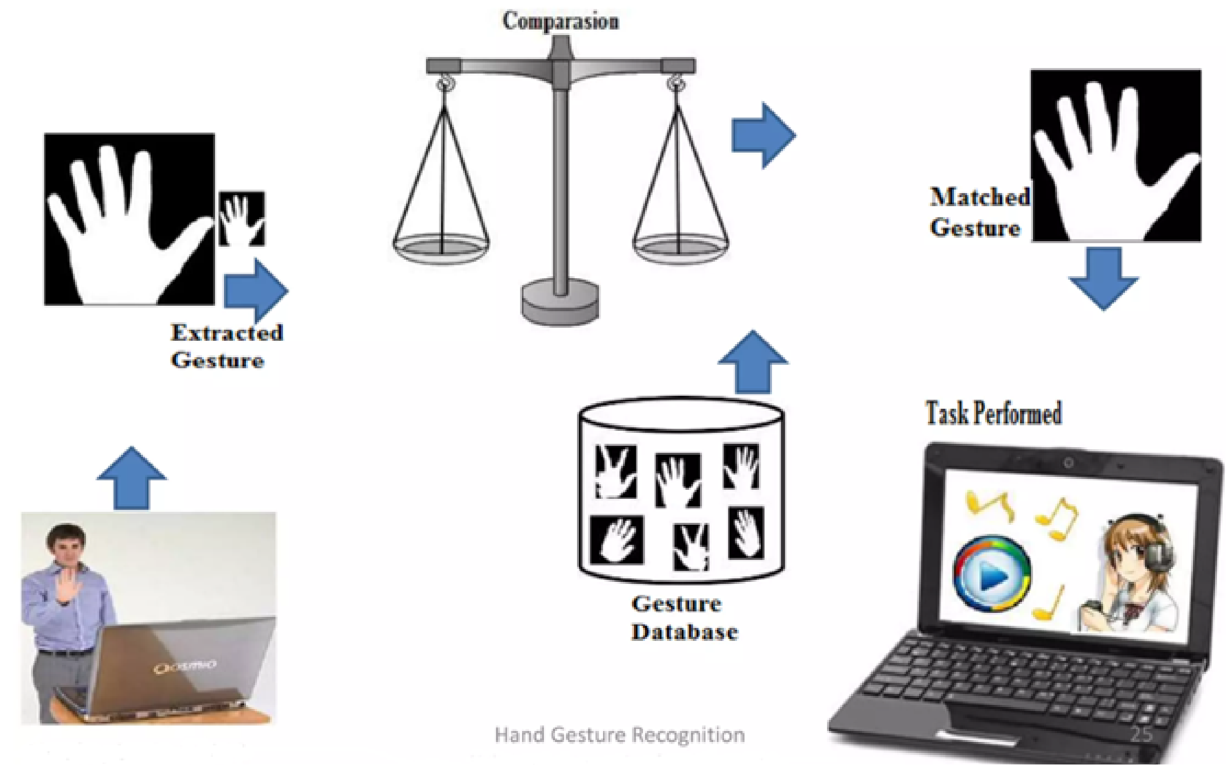


4.3 Integration with Multiple Platforms

Recognizing the diversity of presentation software, the proposed system is designed to be platform-agnostic. It seeks to integrate seamlessly not only with Microsoft PowerPoint but also with other popular platforms such as Google Slides and Apple Keynote, ensuring widespread applicability.

4.4 Additional Functionalities

Moving beyond basic slide navigation, the proposed system introduces a spectrum of additional functionalities. From dynamic zooming to real-time content highlighting, the system expands the presenter's toolkit, enabling a more engaging and interactive presentation experience.



5. Implementation Details

5.1 Technology Stack

The technology stack comprises a sophisticated amalgamation of machine learning frameworks (TensorFlow, PyTorch), computer vision libraries (OpenCV, MediaPipe), and presentation software APIs. This comprehensive stack forms the backbone for a robust and adaptable system architecture.

5.2 System Architecture

The system architecture is intricately designed to accommodate the complexities of gesture recognition and presentation control. Modules for hand gesture detection, machine learning model interfaces, and seamless communication with presentation software contribute to a cohesive and scalable architecture.

5.3 User Interface Design

User interface design is a pivotal aspect of the system's usability. The interfaces are crafted with a focus on user-friendliness, providing presenters with intuitive controls for customizing gestures, viewing real-time feedback on recognition, and effortlessly navigating through their presentations.

6. Code

```
#Importing Modules

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

print(pathImages) # ['Slide1.PNG', 'Slide2.PNG', 'Slide3.PNG', 'Slide4.PNG', 'Slide5.PNG', 'Slide6.PNG', 'Slide7.PNG', 'Slide8.PNG', 'Slide9.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
#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=False is kept such that if right hand is on the
webcam then right must be displayed on the webcam not left after flipping is used
    # 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

        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

                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([])
            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
    # 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

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

    cv2.imshow("window",img)

    cv2.imshow("Slides", imgCurrent)

    key=cv2.waitKey(1)

    if key==ord('q'):

      Break
```

7. Results and Evaluation

7.1 Performance Metrics

The evaluation of the system's performance involves a meticulous examination of various key metrics to ensure its effectiveness and reliability.

7.1.1 Gesture Recognition Accuracy:

A fundamental aspect of the system's success lies in its ability to accurately recognize and interpret hand gestures. To measure accuracy, a comprehensive dataset comprising a diverse range of gestures is utilized. The system is evaluated based on its ability to correctly identify and respond to predefined gestures, ensuring a high level of precision in slide control and additional functionalities.

7.1.2 Responsiveness:

The responsiveness of the system is a critical factor in creating a seamless and natural interaction experience. Metrics for responsiveness include the time taken between a recognized gesture and the corresponding system action. A low-latency response is crucial for maintaining the presenter's flow and engagement with the audience.

7.2 User Feedback

User testing involves a diverse group of presenters, ranging from novice users to experienced speakers. Participants are tasked with navigating through presentations, utilizing gestures for various commands, and providing feedback on the system's overall usability. Observations are made regarding the intuitiveness of gesture controls, ease of customization, and the overall user experience.

8. Conclusion

The hand gesture-controlled PowerPoint system represents a groundbreaking innovation in the realm of presentations, successfully addressing the limitations of traditional control methods. By leveraging advanced technologies such as machine learning, deep learning, and computer vision, the system offers presenters an intuitive and dynamic means of interacting with their content. The focus on inclusivity and accessibility ensures that individuals with mobility impairments can participate more actively in the presentation process, fostering a more equitable environment. The proposed system's versatility is underscored by its platform-agnostic design, seamlessly integrating with various presentation software beyond Microsoft PowerPoint. The inclusion of advanced functionalities, such as customizable gestures and real-time content highlighting, enriches the presenter's toolkit, contributing to a more engaging and interactive presentation experience. Rigorous testing, both quantitatively and qualitatively, validates the system's performance and user-friendliness. While overcoming challenges encountered during development, the system paves the way for future enhancements, emphasizing continuous innovation. In achieving its objectives, the hand gesture-controlled PowerPoint system emerges not only as a transformative tool for presenters but also as a catalyst for shaping the future landscape of human-computer interaction and interactive communication.