VIRTUAL PAINTER USING HAND GESTURE

A report submitted in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

ELECTRONICS AND COMMUNICATION ENGINEERING

By

Divyansh Tak (20210)

Navneet Kumar (20222)

Parvinder Kumar (20228)

SCHOOL OF ELECTRONICS

INDIAN INSTITUTE OF INFORMATION TECHNOLOGYUNA

HIMACHAL PRADESH

November 2022

BONAFIDE CERTIFICATE

This is to certify that the project titled VIRTUAL PAINTER USING HAND GESTURE is a bonafide record of the work done by

Navneet Kumar (20222)

in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in ELECTRONIC AND COMMUNICATION ENGINEERING of the INDIAN INSTITUTE OF INFORMATION TECHNOLOGY UNA, HIMACHAL PRADESH, during the year 2020 - 2024.

Under the guidance of
DR. GURPREET KAUR

Project viva-voce held on: _____

Internal Examiner External Examiner

ORIGNALITY / NOPLAGARISMDECLARATION

We certify that this project report is our original report and no part of it is copied from any published reports, papers, books, articles, etc. We certify that all the contents in this report are based on our personal findings and research and we have cited all the relevant sources which have been required in the preparation of this project report, whether they be books, articles, reports, lecture notes, and any other kind of document. We also certify that this report has not previously been submitted partially or as whole for the award of degree in any other university in India and/or abroad.

We hereby declare that, we are fully aware of what constitutes plagiarism and understand that if it is found at a later stage to contain any instance of plagiarism, our degrees may be cancelled.

Navneet Kumar (20222)

ABSTRACT

The customary drawing framework in view of the 2-D mouse cannot understand the haptic connection among administrators and virtual climate. This isn't accessible for administrator to feel the collaboration awareness. A virtual space painting framework in light of straight forwardness haptic connection is proposed. This framework involves the permission type haptic gadget as the data collaboration interface among administrators and virtual climate. This paper presents a certain power control calculation to adjust the control of the straightforwardness haptic collaboration gadget. This framework utilize the body contact recognition, the virtual power age and perfection to construct the virtual climate. At last, a virtual work of art try is finished. The outcome demonstrates the way that this framework could work on the straightforwardness of the haptic collaboration. It implies that the genuine imaginative creation could be acknowledged on the PC.

***Keywords*:** Gesture Recognition, OpenCV, Mediapipe, python

ACKNOWLEDGEMENT

We would like to thank the following people for their support and guidance without whom the completion of this project in fruition would not be possible.

We would like to express our sincere gratitude and heartfelt thanks to [DR. GURPREET KAUR] for their unflinching support and guidance, valuable suggestions and expert advice. Their words of wisdom and expertise in subject matter were of immense help throughout the duration of this project.

We also take the opportunity to thank our Director and all the faculty of School of Electronics, IIIT Una for helping us by providing necessary knowledge base and resources.

We would also like to thank our parents and friends for their constant support.

Navneet Kumar (20222)

TABLE OF CONTENTS

LIST OF ACRONYMS

OpenCv Open Source Computer Vision Library

HCI Human-Computer Interaction

LIST OF TABLES

LIST OF FIGURES

Chapter 1
Introduction

1.1 Virtual painter using hand gesture

Sketching on Air is made possible by our cutting-edge open-source and Python technologies. The most well-known use of open cv is as an open source computer vision and machine learning programmed. The collection contains more than 2400 of the greatest algorithms, including a wide range of traditional and cutting-edge computer vision and machine learning techniques. The majority of these algorithms are used to extract 3D ones, track camera movements, identify objects, classify human actions in films, and detect and recognise faces. One of the high-level, all-purpose programming languages is Python. The major goal of the object-oriented approach is to assist programmers in producing clean, comprehensible code for both small- and large-scale projects. In this project, we are processing photos using morphological operations, which are a set of actions based on forms.

1.1.1 Hand Detection

It includes of contours, convex hull, convexity defects and finger count In Contours are the curves joining all the continuous points along the boundary, having same color or intensity. The contours are a useful tool for shape analysis and object detection and recognition. The contour is drawn along the boundary of the hand image which is found after thresholding.

In convex hull is the set of continuous points in the Euclidean space that is connected to contours. Convex hull is drawn around the contour. Contour points within the convex hull. Convex hull works as an envelope around the hand.

In convexity defect, When the convex hull is drawn around the contour of the hand, it fits set of contour points of the hand within the hull. It uses minimum points to form the hull to include all contour points inside or on the hull and maintain the property of convexity. This causes the formation of defects in the convex hull with respect to the contour drawn on hand.

1.1.2 Gesture controlling the paint application

Without touching a remote control or even your laptop's screen, gesture detection enables you to control your virtual painting programmed. A camera notices the ball in the glove and takes action. For instance, if we move our hand to the right side of our body, a rectangle will be drawn, and if we move it to the left, a circle will be formed. Accordingly, the gestures we make will cause the activity to take place.

Chapter 2

Review of Literature

2.1 Hand Gesture

Many methods are used for hand gesture recognition in real-time. Sayem Mohammad Siam, Jahidul Adnan Sakel, and Md. Hasanul Kabir has proposed a new method of HCI (Human-Computer Interaction), that uses marker detection and tracking technique. Instead of having a mouse or touchpad, two colored markers are worn on the tips of the fingers to generate eight hand movements to provide instructions to a desktop or laptop computer with a consumer-grade camera [1]. They have also used the "Template matching" algorithm for the detection of markers and Kalman Filter for tracking. In [2] the developed system uses a data glove-based approach to recognize real-time dynamic hand gestures. The data glove has ten soft sensors integrated in it that measure the joint angles of five fingers and are used to collect gesture data. Real-time gestures are recognized using techniques such as gesture spotting, gesture sequence simplification, and gesture recognition.

2.1.1 Images

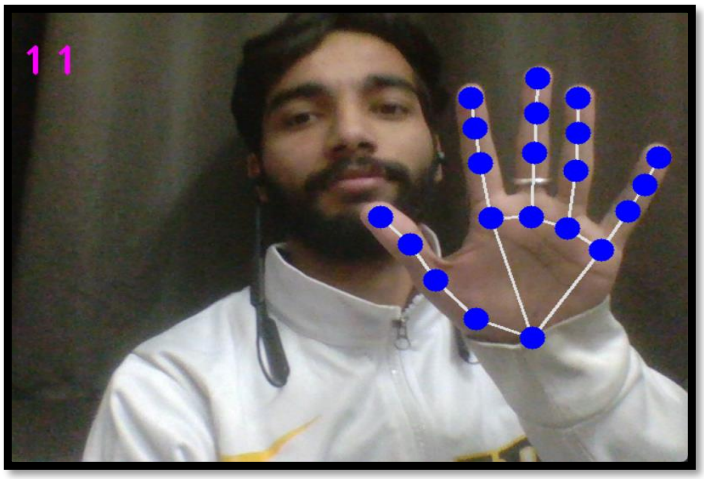

Figure 2.1: Hand Landmarks Node.

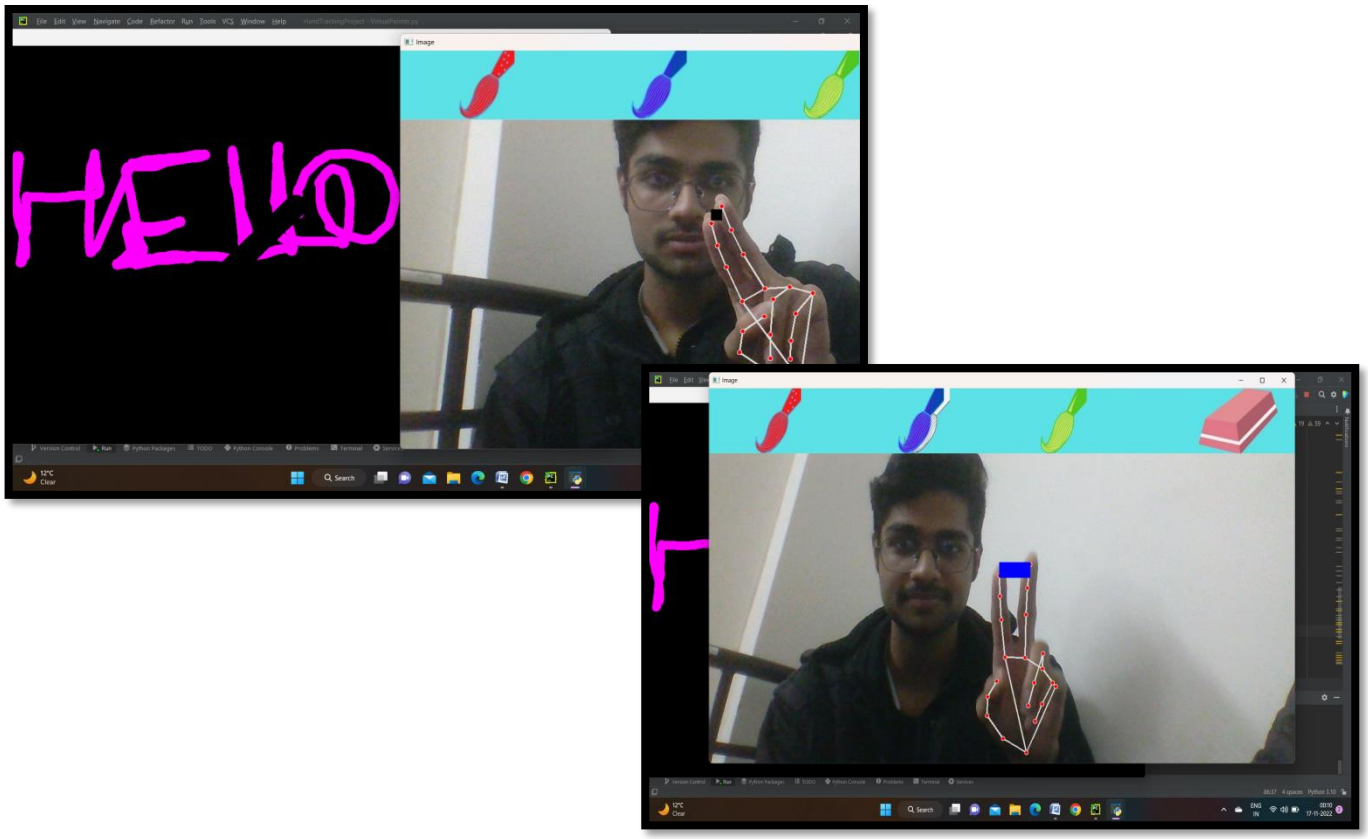

Figure 2.2: Virtual Painter

2.2 Hand Landmark Model

After detecting the palm over the whole image, our subsequent hand landmark model uses regression, or direct coordinate prediction, to accomplish precise keypoint localization of 21 3D hand-knuckle coordinates inside the detected hand regions. The model acquires a reliable internal hand posture representation and is unaffected by self-occlusions or partially visible hands.We manually added 21 3D coordinates to around 30K real-world photos to obtain ground truth data, as shown below (we take Z-value from image depth map, if it exists per corresponding coordinate). We additionally render a high-quality synthetic hand model over a variety of backgrounds and map it to the associated 3D coordinates in order to better cover the range of possible hand poses and provide additional supervision on the nature of hand geometry.

2.2.2 Images

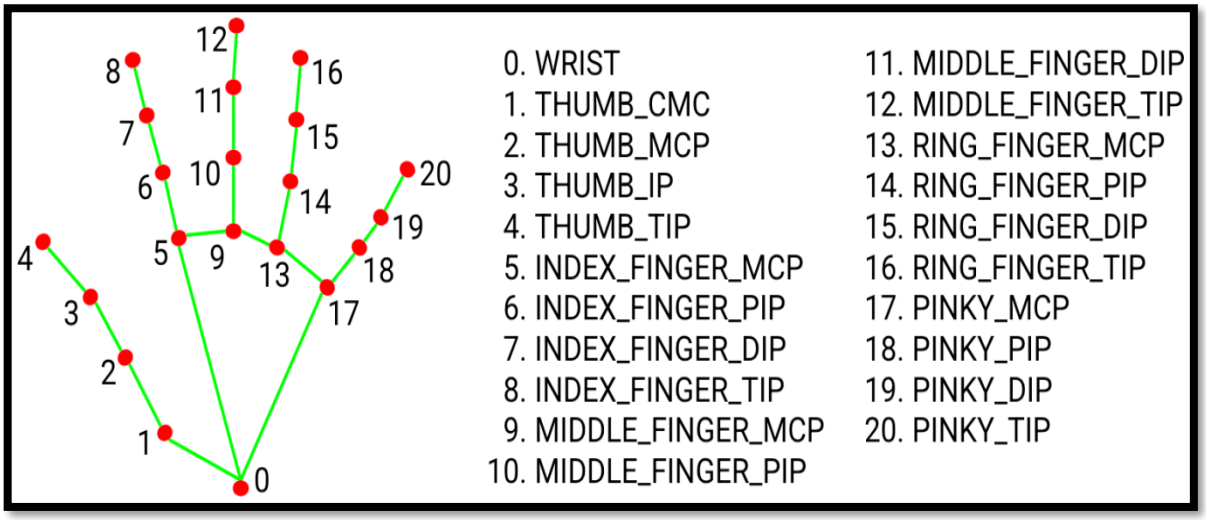

Figure 2.3: Hand Landmarks

1. Ghotkar, A. S., Khatal, R., Khupase, S., Asati, S., & Hadap, M. (2017)

Hand gesture recognition for Indian Sign Language. 2017 International Conference on Computer Communication and Informatics. In this paper, we introduce a hand gesture recognition system to recognize the alphabets of Indian Sign Language. In our proposed system there are 4 modules: real time hand tracking, hand segmentation, feature extraction and gesture recognition

2. Islam, M. R., Mitu, U. K., Bhuiyan, R. A., & Shin, J. (2018). Hand Gesture Feature Extraction Using Deep Convolution Neural Network for Recognizing American Sign Language.

Interacting with computers, human Hand Gesture Recognition (HGR) is the most significant way and the major part of HCI. Extracting features and detecting hand gesture from inputted color videos is more challenging because of the huge variation in the hands. For resolving this issue, this paper introduces an effective HGR system for low-cost color video using webcam. In this proposed model, Deep Convolutional Neural Network (DCNN) is used for extracting efficient hand features to recognize the American Sign Language (ASL) using hand gestures.

SL NO.	PUBLICATION YEAR	PROPOSED	MERITS
1	Ghotkar, A. S., Khatal, R., Khupase, S., Asati, S., & Hadap, M. (2017)	Hand gesture recognition for Indian Sign Language. 2017 International Conference on Computer Communication and Informatics. In this paper, we introduce a hand system to recognize the alphabets of Indian Sign Language. In our proposed system there are 4 modules: real time hand tracking, hand segmentation, feature extraction and gesture recognition	The first one was a new feature extraction technique, includes the feature of three different existing feature extraction techniques. easy-to-use and inexpensive approach to recognize single handed as well as double handed gestures accurately.
2	Islam, M. R., Mitu, U. K., Bhuiyan, R. A., & Shin, J. (2018). Hand Gesture Feature Extraction Using Deep Convolution Neural Network for Recognizing American Sign Language.	Interacting with computers, human Hand Gesture Recognition (HGR) is the most significant way and the major part of HCI. Extracting features and detecting hand gesture from inputted color videos is more challenging because of the huge variation in the hands. For resolving this issue, this paper introduces an effective HGR system for low-cost color video using webcam. In this proposed model, Deep Convolution Neural Network (DCNN) is used for extracting efficient hand features to recognize the American Sign Language (ASL) using hand gestures.	Distinct person hand gesture is used for validation in this paper. The proposed model shows satisfactory performance in terms of classification accuracy, i.e., 94.57%.

Table1: Literature Review

References

1. R. Cutler and M. Turk. View-based Interpretation of Realtime Optical Flow for Gesture Recognition. In Proc. IEEE Intl. Conference on Automatic Face and Gesture Recognition, pages 416–421, April 1998

2. E. Hjelmas˚ and B. K. Low. Face Detection: A Survey. Computer Vision and Image Understanding, 83(3):236–274, September 2001.

3. Druzhkov, P.N., Erukhimov, V.L., Zolotykh, N.Y "New object detection features in the OpenCV library", Pattern Recognit. Image Anal. 21, 384(2011).

4. Choi, J., Park, H., and Park, J. 2009. Interface for augmented reality using efficient hand gesture recognition. *In Proceedings of 14th Korea-Japan Joint Workshop on Frontiers of Computer Vision 2009*, 439--444.

5. Cheng Li, Kris M. Kitani; Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013, pp. 3570-3577

6. R. T. Collins, A. J. Lipton, T. Kanade, H. Fujiyoshi, D. Duggins, Y. Tsin, D. Tolliver, N. Enomoto, O. Hasegawa, P. Burt, and L. Wixson. A system for video surveillance and monitoring. Technical report, 2000.

7. J. M. Geusebroek, R. van den Boomgaard, A. W. M. Smeulders, and H. Geerts. Color invariance. IEEE Trans. Pattern Anal. Machine Intell., 23(12): 1338--1350, 2001.

Appendices

AppendixA

Code Attachments

The following is the partial / subset of the code. Code of some module(s) have been wilfully supressed.

A.1 Sample Code

```
import cv2
import numpy as np
import time
import os
import HandTrackingModule as htm



#####
brushThickness = 15
eraserThickness = 40

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(0)
cap.set(3,1280)
cap.set(4,720)
```

```
detector = htm.handDetector(detectionCon=0.85)
xp,yp = 0,0
imgCanvas = np.zeros((720,1200,3),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]:
```

```
    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 = (255,255,0)
        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
    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

    imgGray = cv2.cvtColor(imgCanvas,cv2.COLOR_BGR2GRAY)
    _, imgTnv = cv2.threshold(imgGray,50,255, cv2.THRESH_BINARY_INV)

    # 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.waitKey(1)
```