**AIR CANVAS**

Major Project

Project Report

**DECCAN COLLEGE OF ENGINEERING AND TECHNOLOGY**

(Affiliated to Osmania University)

Hyderabad - 500001

Submitted by

|  |  |
| --- | --- |
| Aaaa Baaa Cccc | 123-123-123 |
| Aaaa Baaa Cccc | 123-123-123 |
| Aaaa Baaa Cccc | 123-123-123 |

**Department of Computer Science and Engineering**

**2021-2022**

CERTIFICATE

Certified that major-project work entitled “AIR CANVAS” is a bona fide work carried out by:

|  |  |
| --- | --- |
| Aaaa Baaa Cccc | 123-123-123 |
| Aaaa Baaa Cccc | 123-123-123 |
| Aaaa Baaa Cccc | 123-123-123 |

The report has been approved as it satisfies the academic requirements in respect of mini-project work prescribed for the course.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ABSTRACT**

This paper presents a real time video based pointing method which allows sketching and writing of English text over air in front of camera. Proposed method tracks the colored fingertip in the video frames and Here Color Detection and tracking is used in order to achieve the objective. The color 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.

**TABLE OF CONTENTS**

1. **Introduction**
   1. AIR CANVAS- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -- - 5
      1. PROBLEM STATEMENT - - - - - - - - - - - - - - - -- - - - - - - - 5
      2. PROPOSED SOLUTION- - - - - - - - - - - - - - - - - - - - - - - - - 5
      3. FEATURES OF AIR CANVAS -- - - - - - - -- - - - - -- - - - - - -5
2. **Project Design**
   1. PROJECT DESIGN- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 6
      1. ALGORITHM- - - - - - - - - - - - - - - - - - - - - -- - - - - -- - - - - -7
   2. HARDWATE AND SOFTWARE REQUIREMENTS- - - - - - - - - - 7
3. **Project Implementation**
   1. Project Code- - - - - -- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -1

**References** 1

Chapter 1

**Introduction**

* 1. AIR CANVAS

Air canvas helps to draw on a screen just by waiving your finger fitted with a colorful point or a simple colored cap. We will be using the computer vision techniques of OpenCV to build this project. The preferred 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.

* + 1. PROBLEM STATEMENT

Ever thought, waiving your finger into the air can draw on a real canvas. How this air canvas in Computer Vision Projects works.

* + 1. PROPOSED SOLUTION

In this computer vision project that is a Air canvas which helps to draw on a screen just by waiving your finger fitted with a colorful point or a simple colored cap. It was OpenCV which came to the rescue for these computer vision projects. The proposed method provides a natural human-system interaction in such way that it does not require keypad, stylus, pen or glove etc. for character input.

* + 1. FEATURES OF AIR CANVAS

Can track any specific-colored pointer. User can draw in four different colors and even change them without any hassle. Able to rub the board with a single location at the top of the screen. No need to touch the computer once the program is run.

Chapter 2

**Project Design**

2.1 PROJECT DESIGN

Ever wanted to draw your imagination by just waiving your finger in air. Here we will learn to build an Air Canvas which can draw anything on it by just capturing the motion of a colored marker with camera. Here a colored 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 preferred 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 Color Detection and tracking is used in order to achieve the objective. The color 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.

STEPS IN DETAIL:

1. Color Tracking of Object at fingertip. First of all, the incoming image from the webcam is to be converted to the HSV color space for detecting the colored object at the tip of finger. The below code snippet converts the incoming image to the HSV space, which is very suitable and perfect color space for Color tracking. Now, We will make the Trackbars to arrange the HSV values to the required range of color of the colored object that we have placed at our finger.ow, When the trackbars are setup, we will get the real-time value from the trackbars and create range. This range is a numpy structure which is used to be passed in the function cv2.inrange(). This function returns the Mask on the colored object. This Mask is a black and white image with white pixels at the position of the desired color.
2. Contour Detection of the Mask of Color Object Now, after detecting the Mask in Air Canvas, Now is the time to locate its center position for drawing the Line. Here, In the below Snippet of Code, we are performing some morphological operations on the Mask, to make it free of impurities and to detect contour easily
3. Drawing the Line using the position of Contour Now Comes the real logic behind this Computer Vision project, we will form a python deque (A data Structure). The deque will store the position of the contour on each successive frame and we will use these stored points to make a line using OpenCV drawing functions. Now, we will use the position of the contour to make decision, if we want to click on a button or we want to draw on the sheet. We have arranged some of the buttons on the top of Canvas, if the pointer comes into their area, we will trigger their method. We have four buttons on the canvas, drawn using OpenCV.

Clear: Which clears the screen by emptying the deques.

Red: Changes the marker to red color using color array.

Green: Changes the marker to green color using color array

Yellow: Changes the marker to yellow color using color array.

Blue: Changes the marker to blue color using color array.

Also, to avoid drawing when contour is not present, we will Put an else condition which will capture that instant.

1. Drawing the points

Now we will draw all the points on the positions stored in the deques, with respective color.

2.1.1 ALGORITHM

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

2.Prepare the canvas frame and put the respective ink buttons on it.

3.. Adjust the trackbar values for finding the mask of colored marker.

4.Preprocess the mask with morphological operations. (Erosion and dilation)

5.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)

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

2.2 HARDWARE AND SOFTWARE REQUIREMENTS

Operating System: Any Operating System

Supporting software: Python, Numpy, Opencv

Processor: Intel Core i5 7th Gen 2.50GHz RAM: 8GB

Monitor: Any color monitor

Chapter 3

**Project Implementation**

**Project Code**

import numpy as np

import cv2

from collections import deque

#default called trackbar function

def setValues(x):

   print("")

# Creating the trackbars needed for adjusting the marker colour

cv2.namedWindow("Color detectors")

cv2.createTrackbar("Upper Hue", "Color detectors", 153, 180,setValues)

cv2.createTrackbar("Upper Saturation", "Color detectors", 255, 255,setValues)

cv2.createTrackbar("Upper Value", "Color detectors", 255, 255,setValues)

cv2.createTrackbar("Lower Hue", "Color detectors", 64, 180,setValues)

cv2.createTrackbar("Lower Saturation", "Color detectors", 72, 255,setValues)

cv2.createTrackbar("Lower Value", "Color detectors", 49, 255,setValues)

# Giving different arrays to handle colour points of different colour

bpoints = [deque(maxlen=1024)]

gpoints = [deque(maxlen=1024)]

rpoints = [deque(maxlen=1024)]

ypoints = [deque(maxlen=1024)]

# These indexes will be used to mark the points in particular arrays of specific colour

blue\_index = 0

green\_index = 0

red\_index = 0

yellow\_index = 0

#The kernel to be used for dilation purpose

kernel = np.ones((5,5),np.uint8)

colors = [(255, 0, 0), (0, 255, 0), (0, 0, 255), (0, 255, 255)]

colorIndex = 0

# Here is code for Canvas setup

paintWindow = np.zeros((471,636,3)) + 255

paintWindow = cv2.rectangle(paintWindow, (40,1), (140,65), (0,0,0), 2)

paintWindow = cv2.rectangle(paintWindow, (160,1), (255,65), colors[0], -1)

paintWindow = cv2.rectangle(paintWindow, (275,1), (370,65), colors[1], -1)

paintWindow = cv2.rectangle(paintWindow, (390,1), (485,65), colors[2], -1)

paintWindow = cv2.rectangle(paintWindow, (505,1), (600,65), colors[3], -1)

cv2.putText(paintWindow, "CLEAR", (49, 33), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (0, 0, 0), 2, cv2.LINE\_AA)

cv2.putText(paintWindow, "BLUE", (185, 33), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (255, 255, 255), 2, cv2.LINE\_AA)

cv2.putText(paintWindow, "GREEN", (298, 33), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (255, 255, 255), 2, cv2.LINE\_AA)

cv2.putText(paintWindow, "RED", (420, 33), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (255, 255, 255), 2, cv2.LINE\_AA)

cv2.putText(paintWindow, "YELLOW", (520, 33), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (150,150,150), 2, cv2.LINE\_AA)

cv2.namedWindow('Paint', cv2.WINDOW\_AUTOSIZE)

# Loading the default webcam of PC.

cap = cv2.VideoCapture(0)

# Keep looping

while True:

    # Reading the frame from the camera

    ret, frame = cap.read()

    #Flipping the frame to see same side of yours

    frame = cv2.flip(frame, 1)

    hsv = cv2.cvtColor(frame, cv2.COLOR\_BGR2HSV)

    u\_hue = cv2.getTrackbarPos("Upper Hue", "Color detectors")

    u\_saturation = cv2.getTrackbarPos("Upper Saturation", "Color detectors")

    u\_value = cv2.getTrackbarPos("Upper Value", "Color detectors")

    l\_hue = cv2.getTrackbarPos("Lower Hue", "Color detectors")

    l\_saturation = cv2.getTrackbarPos("Lower Saturation", "Color detectors")

    l\_value = cv2.getTrackbarPos("Lower Value", "Color detectors")

    Upper\_hsv = np.array([u\_hue,u\_saturation,u\_value])

    Lower\_hsv = np.array([l\_hue,l\_saturation,l\_value])

    # Adding the colour buttons to the live frame for colour access

    frame = cv2.rectangle(frame, (40,1), (140,65), (122,122,122), -1)

    frame = cv2.rectangle(frame, (160,1), (255,65), colors[0], -1)

    frame = cv2.rectangle(frame, (275,1), (370,65), colors[1], -1)

    frame = cv2.rectangle(frame, (390,1), (485,65), colors[2], -1)

    frame = cv2.rectangle(frame, (505,1), (600,65), colors[3], -1)

    cv2.putText(frame, "CLEAR ALL", (49, 33), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (255, 255, 255), 2, cv2.LINE\_AA)

    cv2.putText(frame, "BLUE", (185, 33), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (255, 255, 255), 2, cv2.LINE\_AA)

    cv2.putText(frame, "GREEN", (298, 33), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (255, 255, 255), 2, cv2.LINE\_AA)

    cv2.putText(frame, "RED", (420, 33), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (255, 255, 255), 2, cv2.LINE\_AA)

    cv2.putText(frame, "YELLOW", (520, 33), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (150,150,150), 2, cv2.LINE\_AA)

    # Identifying the pointer by making its mask

    Mask = cv2.inRange(hsv, Lower\_hsv, Upper\_hsv)

    Mask = cv2.erode(Mask, kernel, iterations=1)

    Mask = cv2.morphologyEx(Mask, cv2.MORPH\_OPEN, kernel)

    Mask = cv2.dilate(Mask, kernel, iterations=1)

    # Find contours for the pointer after idetifying it

    cnts,\_ = cv2.findContours(Mask.copy(), cv2.RETR\_EXTERNAL,

        cv2.CHAIN\_APPROX\_SIMPLE)

    center = None

    # Ifthe contours are formed

    if len(cnts) > 0:

        # sorting the contours to find biggest

        cnt = sorted(cnts, key = cv2.contourArea, reverse = True)[0]

        # Get the radius of the enclosing circle around the found contour

        ((x, y), radius) = cv2.minEnclosingCircle(cnt)

        # Draw the circle around the contour

        cv2.circle(frame, (int(x), int(y)), int(radius), (0, 255, 255), 2)

        # Calculating the center of the detected contour

        M = cv2.moments(cnt)

        center = (int(M['m10'] / M['m00']), int(M['m01'] / M['m00']))

        # Now checking if the user wants to click on any button above the screen

        if center[1] <= 65:

            if 40 <= center[0] <= 140: # Clear Button

                bpoints = [deque(maxlen=512)]

                gpoints = [deque(maxlen=512)]

                rpoints = [deque(maxlen=512)]

                ypoints = [deque(maxlen=512)]

                blue\_index = 0

                green\_index = 0

                red\_index = 0

                yellow\_index = 0

                paintWindow[67:,:,:] = 255

            elif 160 <= center[0] <= 255:

                    colorIndex = 0 # Blue

            elif 275 <= center[0] <= 370:

                    colorIndex = 1 # Green

            elif 390 <= center[0] <= 485:

                    colorIndex = 2 # Red

            elif 505 <= center[0] <= 600:

                    colorIndex = 3 # Yellow

        else :

            if colorIndex == 0:

                bpoints[blue\_index].appendleft(center)

            elif colorIndex == 1:

                gpoints[green\_index].appendleft(center)

            elif colorIndex == 2:

                rpoints[red\_index].appendleft(center)

            elif colorIndex == 3:

                ypoints[yellow\_index].appendleft(center)

    # Append the next deques when nothing is detected to avois messing up

    else:

        bpoints.append(deque(maxlen=512))

        blue\_index += 1

        gpoints.append(deque(maxlen=512))

        green\_index += 1

        rpoints.append(deque(maxlen=512))

        red\_index += 1

        ypoints.append(deque(maxlen=512))

        yellow\_index += 1

    # Draw lines of all the colors on the canvas and frame

    points = [bpoints, gpoints, rpoints, ypoints]

    for i in range(len(points)):

        for j in range(len(points[i])):

            for k in range(1, len(points[i][j])):

                if points[i][j][k - 1] is None or points[i][j][k] is None:

                    continue

                cv2.line(frame, points[i][j][k - 1], points[i][j][k], colors[i], 2)

                cv2.line(paintWindow, points[i][j][k - 1], points[i][j][k], colors[i], 2)

    # Show all the windows

    cv2.imshow("Tracking", frame)

    cv2.imshow("Paint", paintWindow)

    cv2.imshow("mask",Mask)

    # If the 'q' key is pressed then stop the application

    if cv2.waitKey(1) & 0xFF == ord("q"):

        break

# Release the camera and all resources

cap.release()

cv2.destroyAllWindows()

References

1. Ayushman Dashz, Amit Sahuz, Rajveer Shringiz, John Gamboax Muhammad Zeshan Afzalx, Muhammad Imran Maliky, Sheraz Ahmedy and Andreas Dengely”AirScript Creating Documents in Air” 14th IAPR International Conference on Document Analysis and Recognition (ICDAR) IEEEXplore2017
2. Air-writing Recognition, Part 2: Detection and Recognition of Writing Activity in Continuous Stream of Motion Data Mingyu Chen, Ghassan AlRegib, Senior Member, IEEE, and BiingHwang Juang, Fellow, IEEE.IEEE TRANSACTIONS ON HUMANMACHINE SYSTEMS.
3. A Novel Human-3DTV Interaction System Based on Free Hand Gestures and a Touch-Based Virtual Interface by SHUN ZHANG AND SHIZHOU ZHANG IEEE Sensors J., vol. 19, no. 20, pp. 95049511, Oct. 2019.