AIR CANVAS

Design & developed by

G.Prabhas Reddy	2011CS020141
G.Punith	2011CS020142
G.Deepak	2011CS020143
G.Lavanya	2011CS020144
G.Dhanush	2011CS020145

GUIDED BY

T.Vinay Simha Reddy M.Tech, (Ph. D)

Department of Computer Science & Engineering (AI &ML)

Malla Reddy University,Hyderabad

2020-2024

MALLA REDDY UNIVERSITY

(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

CERTIFICATE

This is to certify that this is the bonafide record of the application development entitled,submitted,by,**G.PrabhasReddy**:**2011CS020141,G.Punith:2011CS020142,G.Deepak:2011CS020143,G.Lavanya:2011CS020144,G.Dhanush: 2011CS020145**, B. Tech II year II semester, Department of CSE (AI&ML) during the year 2021-22. The results embodied in this report have not been submitted to any other university or institute for the award of any degree or diploma.

Internal Guide
Prof T.Vinay Simha Reddy,MTech, (Ph.D)

Head of the department
Dr. Thayyaba Khatoon
CSE(AI&ML)

External Examiner

ACKNOWLEDGEMENT

We would like to express our gratitude to all those who extended their support and suggestions to come up with this application. Special Thanks to our mentor Prof. T.Vinay Simha Reddy whose help and stimulating suggestions and encouragement helped us all time in the due course of project development.

We sincerely thank our HOD Prof. Dr. Thayyaba Khatoon for her constant support and motivation all the time. Thanks to God for showering his kindness on us and directing us in accomplishing this task. A special acknowledgement goes to a friend who enthused us from the back stage. Last but not the least our sincere appreciation goes to our family who has been tolerant understanding our moods, and extending timely support

INDEX

ABSTRACT

Writing in air has been one of the most fascinating and challenging research areas in field of image processing and pattern recognition in the recent years. It contributes immensely to the advancement of an automation process and can improve the interface between man and machine in numerous applications. Several research works have been focusing on new techniques and methods that would reduce the processing time while providing higher recognition accuracy. Object tracking is considered as an important task within the field of Computer Vision. The invention of faster computers, availability of inexpensive and good quality video cameras and demands of automated video analysis has given popularity to object tracking techniques. Generally, video analysis procedure has three major steps: firstly, detecting of the object, secondly tracking its movement from frame to frame and lastly analysing the behaviour of that object. For object tracking, four different issues are taken into account; selection of suitable object representation, feature selection for tracking, object detection and objct tracking. In real world, Object tracking algorithms are the primarily part of different applications such as: automatic surveillance, video indexing and vehicle navigation etc. The project takes advantage of this gap and focuses on developing a motion- to-text converter that can potentially serve as software for intelligent wearable devices for writing from the air. This project is a reporter of occasional gestures. It will use computer vision to trace the path of the finger. The generated text can also be used for various purposes, such as sending messages, emails, etc. It will be a powerful means of communication for the deaf. It is an effective communication method that reduces mobile and laptop usage by eliminating the need to write.

INTRODUCTION

In the era of digital world, traditional art of writing is being replaced by digital art. Digital art refers to forms of expression and transmission of art form with digital form. Relying on modern science and technology is the distinctive characteristics of the digital manifestation. Traditional art refers to the art form which is created before the digital art. From the recipient to analyse, it can simply be divided into visual art, audio art, audio-visual art and audio-visual imaginary art, which includes literature, painting, sculpture, architecture, music, dance, drama and other works of art. Digital art and traditional art are interrelated and interdependent. Social development is not a people's will, but the needs of human life are the main driving force anyway. The same situation happens in art. In the present circumstances, digital art and traditional art are inclusive of the symbiotic state, so we need to systematically understand the basic knowledge of the form between digital art and traditional art. The traditional way includes pen and paper, chalk and board method of writing. The essential aim of digital art is of building hand gesture recognition system to write digitally. Digital art includes many ways of writing like by using keyboard, touch-screen surface, digital pen, stylus, using electronic hand gloves, etc. But in this system, we are using hand gesture recognition with the use of machine learning algorithm by using python programming, which creates natural interaction between man and machine. With the advancement in technology the need of development of natural 'human – computer interaction (HCI)' [10] systems to replace traditional systems is increasing rapidly

.

DESIGN AND IMPLEMENTATION

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 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 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 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.

STEPS IN DETAIL:

1. Colour Tracking of Object at fingertip. First of all, The incoming image from the webcam is to be converted to the HSV colour space for detecting the coloured object at the tip of finger. The below code snippet converts the incoming image to the HSV space, which is very suitable and perfect colour space for Colour tracking. Now, We will make the Trackbars to arrange the HSV values to the required range of colour of the coloured object that we have placed at our finger row. 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 coloured object. This Mask is a black and white image with white pixels at the position of the desired colour.

2. Contour Detection of the Mask of Colour 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 them,M ask, to make it free of impurities and to detect contour easily. 2 air canvas 2. Project De sign.

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 colour using colour array.
   Green : Changes the marker to Green colour using colour array.
   Yellow : Changes the marker to Yellow colour using colour array.
   Blue : Changes the marker to Blue colour using colour array.

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

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

Design Overview :

The basic goal of Air Canvas is to map the coordinates of the user's pointer finger to the screen, where coloured circles are drawn and connected to simulate a crude brush stroke. Our project consisted of three hardware devices: a Raspberry Pi board, a PiTFT display screen, and a Raspberry Pi Camera. The PiTFT screen is connected to the RPi's 40-pin connector, and the camera module is attached directly into the camera port. We designated each of the PiTFT's physical buttons as program controls:

- Pin 17: colour change between red, green, and blue
- Pin 22: increase brush size
- Pin 23: decrease brush size
- Pin 27: quit/end program

OpenCV

We began our project by searching for open source hand gesture recognition software that utilized OpenCV in combination with Python. In doing so, our project's design changed as we discovered different image processing algorithms. Our primitive implementation sought to use hand gestures to control our color and size variables. To do so, we first set out to create an image mask that would separate the hand from the background. With some trial and error using OpenCV, we successfully captured an image, Gaussian blurred it, and applied a binary mask to starkly contrast the hand shape from the background. This is a method obtained from Izane's Finger Detection tutorial1, chosen because of its use of convexity detection; in other words, determining the valleys between the fingers.However, we discovered that thecamera's sensitivity to the lab room's lighting made this a difficult endeavor, and we often encountered extraneous silhouettes in our processed image.

PyGame Drawing

Our next step was to independently develop the PyGame side of the project, which supported the PiTFT drawing functionality. First, we chose our method of drawing: as with most digital art programs, the brush head is a single shape, and a brush stroke is, simply put, the chosen shape repeated over the length of the stroke. We decided to draw very simple circles at a set of linearly spaced coordinates in PyGame using the function *pygame.draw.circle().* The circles were spaced such that the overlap of each dot would resemble a connected line. We were able to display two straight lines in PyGame with this method. Additionally, we added a list of colors — red, blue, and green — to toggle between as desired, as well as a function to increase and decrease the radius of our brush head. With the PyGame end completed, we set off to combine the two main functionalities of our project.

Multiprocessing Attempt

At this point in our project, we attempted to implement our program using multiple processes of the Raspberry Pi. The intended implementation involved capturing the feed with a master process and handing off singular frames via queue to three worker processes for image analysis. The master process would then receive the coordinates of the pointer finger from its workers and draw accordingly. However, using multiple processes did not pan out as desired. The results are discussed in the Results section.

PROBLEM DEFINITION

The project focuses on solving some major societal problems –

1. **People hearing impairment**: Although we take hearing and listening for granted, they communicate using sign languages. Most of the world can't understand their feeling, their emotions without a translator in between.

2. **Overuse of Smartphones:** They cause accidents, depression, distractions, and other illnesses that we humans can still discover. Although its portability, ease to use is profoundly admired, the negatives include life threatening events.

3. **Paper wastage** is not scarce news. We waste a lot of paper in scribbling, writing, drawing, etc.… Some basic facts include - 5 litres of water on average are required to make one A4 size paper, 93% of writing is from trees, 50% of business waste is paper, 25% landfill is paper, and the list goes on. Paper wastage is harming the environment by using water and trees and creates tons of garbage. Air Writing can quickly solve these issues. It will act as a communication tool for people with hearing impairment. Their air-written text can be presented using AR or converted to speech. One can quickly write in the air and continue with your work without much distraction. Additionally, writing in the air does not require paper. Everything is stored electronically.

ALGORITHM OF WORKFLOW

This is the most exciting part of our system. Writing involves a lot of functionalities. So, the number of gestures used for controlling the system is equal to these number of actions involved.

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

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

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

4.Preprocess the mask with morphological operations.(Erotion 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 .

The basic functionalities we included in our system are :

1. Writing Mode - In this state, the system will trace the fingertip coordinates and stores them.
2. Colour Mode – The user can change the colour of the text among the various available colours.
3. Backspace - Say if the user goes wrong, we need a gesture to add a quick backspace.

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 colour monitor

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 These
# trackbars will be used for setting
# the upper and lower ranges of the # HSV required
for particular 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 These arrays
# will hold the points of a particular colour
# in the array which will further be used
# to draw on canvas bpoints =
[deque(maxlen = 1024)] gpoints =
[deque(maxlen = 1024)] rpoints =
[deque(maxlen = 1024)]
ypoints = [deque(maxlen = 1024)]
```

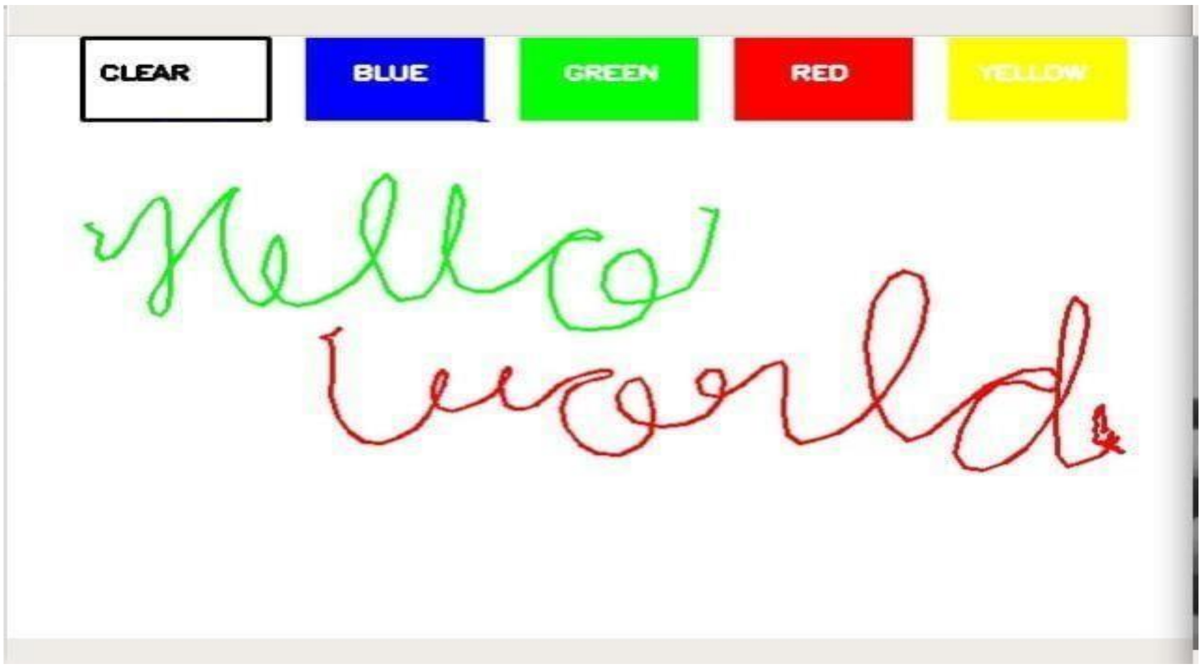
```
# These indexes will be used to mark position
# of pointers in colour array
blue_index = 0
green_index = 0 red_index
= 0
yellow_index = 0

# The kernel to be used for dilation purpose kernel
= np.ones((5, 5), np.uint8)

# The colours which will be used as ink for
# the drawing purpose 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

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)

    # Getting the updated positions of the trackbar
    # and setting the HSV values
    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)
```

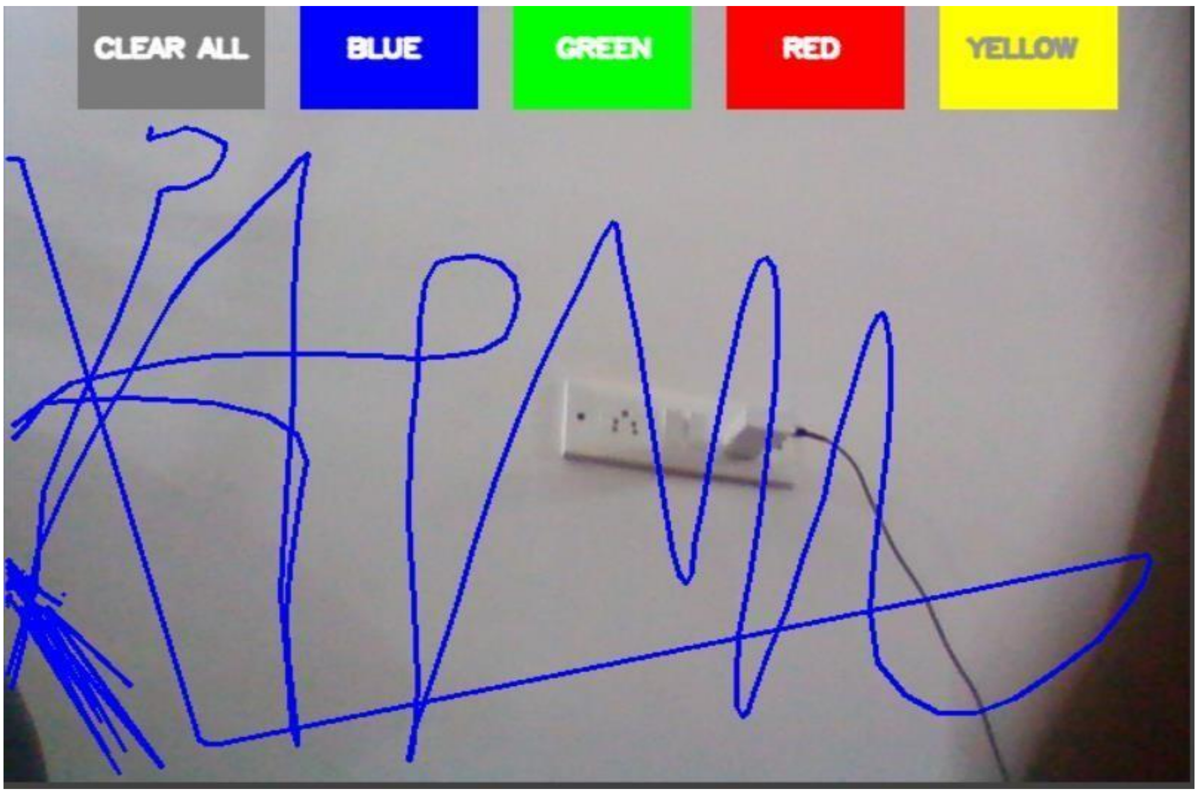
```
        # Identifying the pointer by making its
        # mask
        Mask = cv2.inRange(hsv, Lower_hsv, Upper_hsv)
        Mask = cv2.erode(Mask, kernel, iterations = 1)
        Mask = cv2.morphologyEx(Mask, cv2.MORPH_OPEN, kernel)
        Mask = cv2.dilate(Mask, kernel, iterations = 1)

        # Find contours for the pointer after
        # identifying 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:

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

Result


CLEAR
BLUE
GREEN
RED
YELLOW
Hello
world


CLEAR ALL
BLUE
GREEN
RED
YELLOW

CONCLUSION

The system has the potential to challenge traditional writing methods. It eradicates the need to carry a mobile phone in hand to jot down notes, providing a simple on the-go way to do the same. It will also serve a great purpose in helping especially abled people communicate easily. Even senior citizens or people who find it difficult to use keyboards will able to use system effortlessly. Extending the functionality, system can also be used to control IoT devices shortly. Drawing in the air can also be made possible. The system will be an excellent software for smart wearables using which people could better interact with the digital world. Augmented Reality can make text come alive.

There are some limitations of the system which can be improved in the future. Firstly, using a handwriting recognizer in place of a character recognizer will allow the user to write word by word, making writing faster. Secondly, hand-gestures with a pause can be used to control the real-time system as done by [1] instead of using the number of fingertips. Thirdly, our system sometimes recognizes fingertips in the background and changes their state. Air-writing systems should only obey their master's control gestures and should not be misled by people around. Also, we used the EMNIST dataset, which is not a proper air-character dataset.
Upcoming object detection algorithms such as YOLO v3 can improve fingertip recognition accuracy and speed. In the future, advances in Artificial Intelligence will enhance the efficiency of air-writing.

PROBLEM IDENTIFICATION

1. <u>Fingertip detection</u>

The existing system only works with your fingers, and there are no highlighters, paints, or relatives. Identifying and characterizing an object such as a finger from an RGB image without a depth sensor is a great challenge.

2. <u>Lack of pen up and pen down motion</u>

The system uses a single RGB camera to write from above. Since depth sensing is not possible, up and down pen movements cannot be followed. Therefore, the fingertip's entire trajectory is traced, and the resulting image would be absurd and not recognized by the model. The difference between hand written and air written 'G'.

3. <u>Controlling the real-time system</u>

Using real-time hand gestures to change the system from one state to another requires a lot of code care. Also, the user must know many movements to control his plan adequately.

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 colourful point or a simple coloured 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 do not require keypad, stylus, pen or glove etc for character input.

RESOURCES

- 14th IAPR International Conference on Document Analysis and Recognition (ICDAR) IEEEXplore2017

- Part 2:Detection and Recognition of Writing Activity. IEEE.IEEE Transactions of human machine systems.

- A Novel Human-3DTV Interaction System Based on Free Hand Gestures

- Drawing in PyGame: https://www.pygame.org/docs/ref/draw.html

- “Guide to Set Up Pi Camera & Some OpenCV Image Processing Basics” by Xitang Zhao: https://blackboard.cornell.edu/bbcswebdav/pid-4087974-dt-content- rid-24070270_1/courses/12153_2019SP/12153_2019SP_ImportedContent_20190408043030/Pi_camera_and_OpenCV_v1%281%29.pdf