**Air Canvas Using Python**

Report Submitted in partial fulfilment of requirement for degree of

Report submitted in partial fulfilment of the requirement for

the degree of



B.Tech

In

**Computer Science and Enginnering**

Under the supervision of

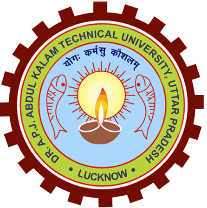
**Mr. Akash Luthra**

By

Sneha Kesarwani (1803310216)

Shalini Jha (1803310188)

Priya Verma (1803310148)



Raj Kumar Goel Institute of Technology, Ghaziabad

Dr. A.P.J. Kalam Technical University , Lucknow

Session:- 2020-21

**DECLARATION**

This is to certify that Synopsis Report Entitled “Air Canvas using Python” which is submitted in partial fulfilment of the requirement of the award of degree B.Tech in Computer Science and Engineering to R.K.G.I.T, Ghaziabad ,Dr. A.P.J. Abdul Kalam Technical University , Lucknow comprises only original work and studies carried out by students himself. The matter embodied in this synopsis has not been for the award of any other degree

**Date:**

**Name Of Student (Roll No.)**

Sneha Kesarwani (1803310216)

Shalini Jha (1803310188)

Priya Verma (1803310148)

**Approved By:**

**Mr. Akash Luthra Mr. Piyush Gupta**

**Project Guide Project Co-ordinator**

**CSE Department CSE Department**

**RKGIT, Ghaziabad RKGIT, Ghaziabad**

**Counter sign By**

**Dr. Sachi Gupta**

**HOD,CSE Department**

**RKGIT, Ghaziabad**

**TABLE OF CONTENTS**

Title………………………………………………………………………………………………………………………. i

Declaration……………………………………………………………………………………………………………. ii

Abstract…………………………………………………………………………………………………………………. iii

List of Figures ………………………………………………………………………………………………………… iv

**CHAPTER 1 INTRODUCTION**

* 1. Fundamentals of Air Canvas
  2. Aim and Objective
     1. Objective
     2. Software Used

**CHAPTER 2 DESIGN AND TESTING**

2.1 Design Overview

2.2 Open CV

2.3 PyGame Drawing

2.4 Open CV and PyGame Integration

2.4.1 Coordinates and interpolation

**CHAPTER 3**  **WORKING OF PROJECT**

3.1 Air Canvas in action

3.2 Technology Used

3.3 Methodology

**CHAPTER 4 RESULT AND FUTURE SCOPE**

**CONCLUSION**

**REFERENCES**

**PROPOSED TIME BOUND PROGRESS OF PROJECT**

**Abstract**

This Project presents the use of air canvas ,we have achieved a hands-free drawing program that uses OpenCV to detect the user’s pointer finger. Colorful lines can be drawn wherever the user desires and the brush can even be modified. It is truly like drawing in the air. Using the libraries of python ,open CV and numPy this project is build up.Air Canvas can be upgraded to run authentically in real time. Moreover, we relied on open source OpenCV code for hand recognition, which had its own issues that we worked hard to circumvent.

This Project is highly use for digital arts ,artist, professionals by eradicating traditional or manual method of canvas

The concept of this project is to allow the world of canvas to go digital , and further be implemented on smart classes or E-classes for better understanding and teaching

Purposes. This is achieved by using a smart platform of Python and open cv . It is easy to use for anyone with a Smartphone and does not require maintainence oce

Set up.

**LIST OF FIGURES**

**CHAPTER – 1**

**INTRODUCTION**

* 1. **FUNDAMENTALS OF AIR CANVAS**

Computer vision project implemented with OpenCV

Ever wanted to draw your imagination by just waiving your finger in air. In this post 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 preffered 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.

**ALGORITHM**

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.
3. Preprocess the mask with morphological operations.(Erotion and dilation)
4. 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)
5. Finally draw the points stored in array on the frames and canvas .

Requirements: python3 , numpy , opencv installed on your system.

**1.2 AIM AND OBJECTIVE**

The aim of this project is to create digital platform to perform canvas based on python open CV which helps us to create canvas in air.

**1.2.1 OBJECTIVE**

To create a simple prototype for a drawing tool that uses hand gesture recognition software to paint on a PiTFT screen. Core objectives includes:

* Using open CV to recognize the pointer finger.
* Mapping Coordinates extracted from hand recognition software to PyGame to produce a drawing.
* Implementing additional features such as color change , size change, and both on-screen and physical buttons
  + 1. **SOFTWARE USED**

Integrated development environment (IDE) :- Pycharm Community Edition 2020.2.3

Android Studio

Java Virtual Machine (JVM)

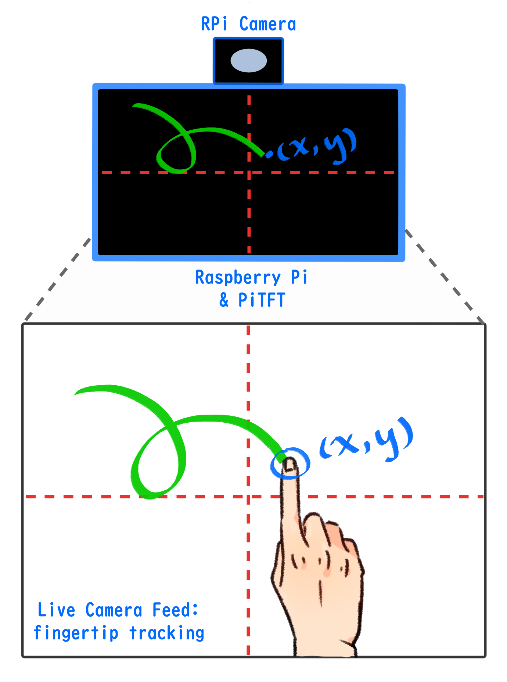
**CHAPTER 2 DESIGN AND TESTING**

**2.1 DESIGN OVERVIEW**

The basic goal of Air Canvas is to map the coordinates of the user’s pointer finger to the screen, where colored 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: color change between red, green, and blue
* Pin 22: increase brush size
* Pin 23: decrease brush size
* Pin 27: quit/end program

A basic visual of Air Canvas can be seen in Figure 1:



**2.2 Open CV**

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 the camera’s sensitivity to the lab room’s lighting made this a difficult endeavor, and we often encountered extraneous silhouettes in our processed image.

We abstracted a function that would calculate the farthest point (representing the index fingertip when held up) and pass it along to our PyGame drawing functions. In doing so, we made some modifications and adaptations. First, we had to map proportionately from the live camera feed to the PiTFT screen, which was exactly half the size of the feed. Next, to eliminate the use of the keyboard, we mapped the histogram trigger to an on-screen PiTFT button. Furthermore, due to the abundance of natural-toned colors in our lab room, we decided to use blue nitrile gloves during our work for a stronger contrast with the background. The bold color helped the color histogram better determine the location of the user’s hand.

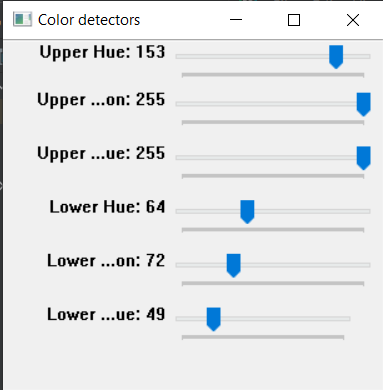
**2.3 PyGame Drawing**

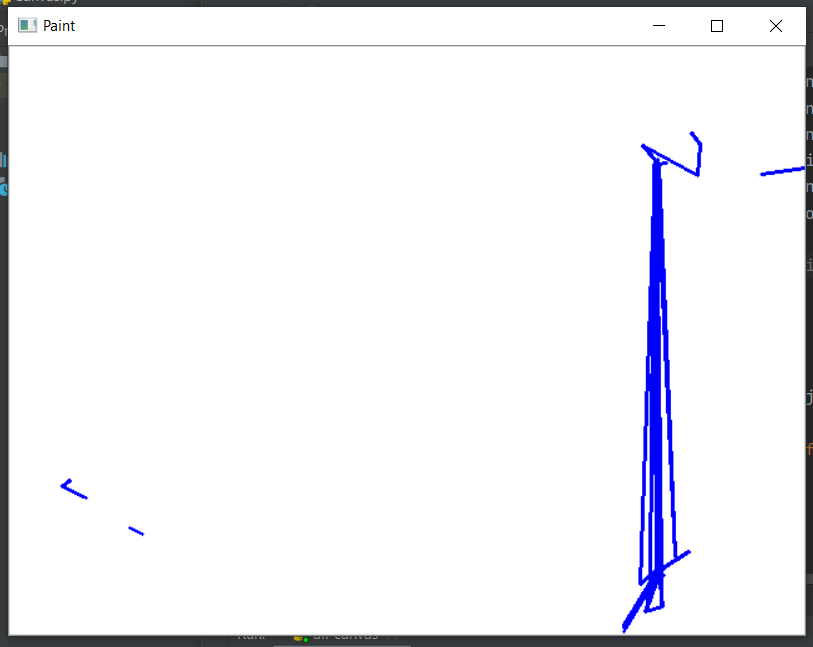
Our next step was to independently develop the PyGame side of the project, which supported the PiTFT drawing functionality. First, we chose our method of drawing: as with most digital art programs, the brush head is a single shape, and a brush stroke is, simply put, the chosen shape repeated over the 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.

**2.4 OpenCV and PyGame Integration**

### **2.4.1 Coordinates and Interpolation**

The combined code required us to feed the coordinates acquired from OpenCV processing to the PyGame program we had written. We soon realized that the frequency of images processed was too slow to draw a continuous line. To solve this problem, we chose to interpolate between the current point of drawing and the previous point. After creating these variables, we also reduced any jitter produced by erroneous detection by setting a threshold distance between current and previous point. Should the distance between the two exceed 10 pixels, the current point would not be considered for drawing and discarded as an outlier coordinate. We then interpolated between valid current and previous points by drawing more circles of the same color and radius with the aforementioned method of linearly filling in space. Figure 3 is a screenshot of a drawing made with finger tracking and simple interpolation.





**CHAPTER 3 WORKING OF PROJECT**

For this project, we will be using face detection using Haar Cascade method. Therefore, we need to download the cascade classifier for face detection from Github.

Let us first understand what is Haar Cascade?

Haar Cascade is a machine learning object detection algorithm used to identify objects in an image or video and based on various​​ features.   
If you want to read a more detailed version of it, check out this [link](https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html).

Now, we will start by importing the libraries and define a variable to capture video from my webcam.

import cv2  
import numpy as np

cap = cv2.VideoCapture(0)

Let us import our classifier file which we downloaded from Github.

face = cv2.CascadeClassifier('haarcascade\_frontalface\_alt.xml')

write a while loop and capture the image frames. Also, we need to mirror the frames so that we can see it right.

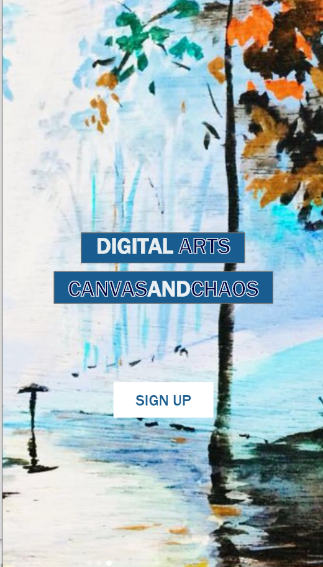
while 1:  
 ret, frame = cap.read() ##Read image frame  
 frame = cv2.flip(frame, +1) ##Mirror image frame

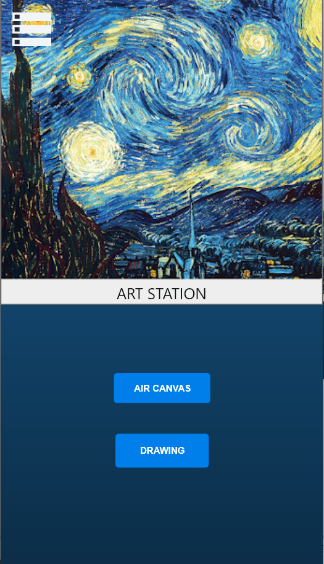
All we have to do now is to convert the image into grayscale and use *detectMultiScale()* method to detect faces in the image frame.

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)  
detect\_face = face.detectMultiScale(gray, 1.2, 1)

*detectMultiScale()*requires three arguments. First is the source image, second is the scale factor which specifies how much the image size is reduced at each image scale and the third is the minimum neighbours which specify how many neighbours each candidate rectangle should have to retain it. This method returns the coordinates of the rectangle detecting a face.

**3.1 Air Canvas in action**





**3.2 Technology Used**

The system is a combination of application and software components. Here in this section am to describe about the application.

**Software used:**

**Pycharm:**

PyCharm is designed by programmers, for programmers, to provide all the tools you need for productive Python development.

**CHAPTER 4 RESULT AND FUTURE SCOPE**

**RESULT**

Overall, we achieved the goals we set out to accomplish and learned a considerable amount about OpenCV, multicore processing, and PyGame programming in the process. In our debugging process, we also encountered some problems involving the PiTFT touchscreen, which we were able to solve by investigating the operating system updates we’d installed during the process of our lab.

**FUTURE SCOPE**

Given more time to work on this project, we would improve hand contour recognition, explore our original Air Canvas goals, and try to understand the multicore module.

To enhance hand gesture tracking, we would have to delve more into OpenCV. There are many different methods of contour analysis, but in this particular algorithm, it may be worthwhile to take a look at the color histogram used to create the contours in question. Furthermore, we could experiment with different interpolation methods. PyGame includes a line drawing method (pygame.draw.line()) that could prove useful in producing smoother, cleaner lines. On the same vein, implementing a variety of brush shapes, textures, and even an eraser would make Air Canvas more robust as a drawing program. Allowing the user to save their final work or watch their drawing process as a playback animation could also be unique features that resemble real creativity software. Perhaps there would even be a way to connect Air Canvas to actual digital drawing programs such as Adobe Photoshop, Clip Studio Paint, or GIMP! Finally, we could make significant strides by figuring out how multicore processing works with in-order information processing.

**CONCLUSION**

We consider our project to be an overall success! With Air Canvas, we have achieved a hands-free drawing program that uses OpenCV to detect the user’s pointer finger. Colorful lines can be drawn wherever the user desires and the brush can even be modified. It is truly like drawing in the air!

Of course, Air Canvas has many flaws that may be interesting areas of research in the future. The first is the issue of frame rate: image processing slowed down the camera feed and produced a cumbersome lag that impedes on the usability of the program. It would be best optimized with multicore functionality, which we attempted in this project. If the timing problems with queueing data between processes can be managed such that frame information is passed in order, perhaps Air Canvas can be upgraded to run authentically in real time. Moreover, we relied on open source OpenCV code for hand recognition, which had its own issues that we worked hard to circumvent.