Digital Image Processing

Virtual whiteboard using openCV

Anagha Manoj - AM.EN.U4CSE20207

Gopika Menon - AM.EN.U4CSE20245

Emeric C Alex - AM.EN.U4CSE20223

ABSTRACT

As we have seen, one of the major reasons for spreading of coronavirus was surface touching, and to avoid this sort of spread in the virus we have come up with this project - “virtual whiteboard”. As the title already suggests, we won't be touching the surface; instead, we'll be drawing in the air, and anything we draw there will be shown on the screen or monitor. This will assist us in stopping the virus's spread during pandemics. Although most transactions in this pandemic have been performed online, many individuals still use ATMs to deposit money or transfer money physically, which is unsafe and increases the likelihood of the virus spreading at such times. As a result, we can use this as a major preventative measure. We may apply this concept not only in ATMs but also in offices for biometrics, class rooms or in cyber cafés, among many other places.

So basically, the screen detects the color whose properties we declare in the code. H, S, V are Hue, Saturation and Value respectively. This color code is similar to the RGB color code. You will have to run the object Tracking code beforehand in order to get the HSV value. All you have to do is slide the taskbars until the only remaining white parts of the “Thresholded Image” windows are corresponding to objects that you want to detect in real life.

After debugging the webcam automatically turns on and we get to draw In Front of the webcam and then the output screen appears and we get the output with the help of the HSV values which we have already declared in code. The HSV adjustment plays a major role in this implementation, based on its value, the text that the person drawing on air will be reflected and visible clearly at the output.

OBJECTIVES

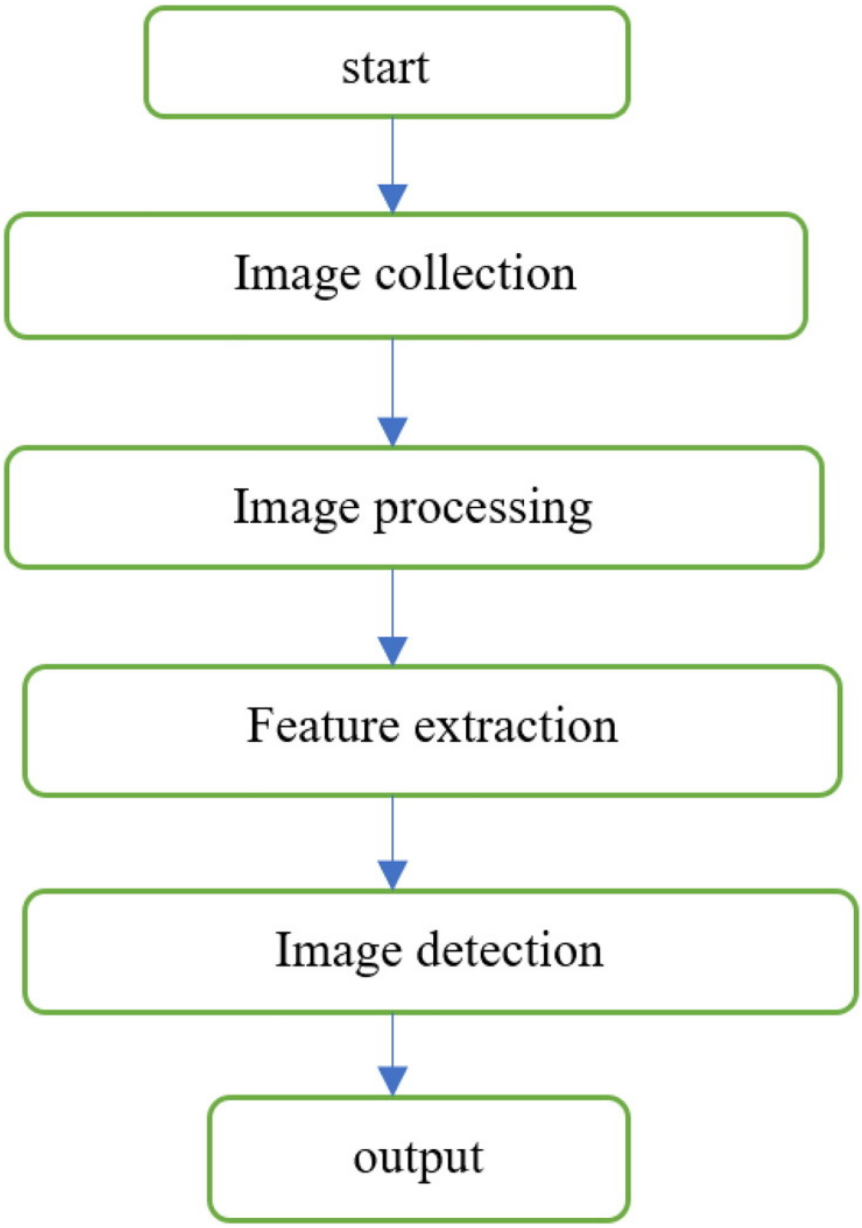
- Using feature extraction and image detection, showing the output, i.e. whatever we drew on air

ASSUMPTIONS

The solution will work for only certain pen colors blue, green, yellow and red. You can readjust the hue, saturation etc properties. First and foremost, after the code is executed, we get a white screen shown on the screen and anything we write on air. In Front of camera it tracks the article whose properties we have proclaimed in the code. After that tracked points are connected and projected on the screen.

SYSTEM ARCHITECTURE

The screen detects the color whose properties we declare in the code. H, S, V are Hue, Saturation and Value respectively. This color code is similar to the RGB color code. You will have to run the object Tracking code e beforehand in order to get the HSV value. All you have to do is slide the taskbars until the only remaining white parts of the “Thresholded Image” windows are corresponding to objects that you want to detect in real life. After debugging the webcam automatically turns on and we get to draw In Front of the webcam and then the output screen appears and we get the output with the help of the HSV values which we have already declared in code. The HSV adjustment plays a major role in this implementation, based on its value, the text that the person drawing on air will be reflected and visible clearly at the output.



MODULES

- Color detection is an image processing technique where we can detect any color in a given range of HSV color space.
- Image processing is a method to perform some operations on an image, in order to get an enhanced image or to extract some useful information from

it. It is a type of signal processing in which input is an image and output may be image or characteristics/features associated with that image.

- Image processing is a method to perform some operations on an image, in order to get an enhanced image or to extract some useful information from it. It is a type of signal processing in which input is an image and output may be image or characteristics/features associated with that image.
- Image segmentation is the process of labeling every pixel in an image, where each pixel shares the same certain characteristics.

Programmed using packages:

- Open CV : Open CV (Open-Source Computer Library) is a programming feature library specifically targeted at computer vision in real time. This is an open-source platform and free for use. Its primary interface is in c++, it still preserves an older c interface that is less detailed but vast. In the Python interface, all the latest advancements and algorithms appear. It is a major open-source computer vision, machine learning and image processing library and now it plays an important part in real time activity in today's systems. We will use this to process photographs and videos to recognize human beings, faces or even handwriting. Python is able to process the OpenCV array structure for review as it is combined with different libraries such as NumPy. We use vector space to recognize the image pattern and its different characteristics and perform arithmetic computations on these traits. It is accessible on Windows, Linux, iOS etc., with Python, C++, C and Java as interfaces.

- Python: It is a high-level programming language famous for code reusability and simplicity. Even though it is slower it has an important characteristic of python that it can be easily extended within c. Because of this characteristic we can write computationally intrinsic codes in C++/C. Python supports

different forms of programming patterns such as procedural programming, object-oriented programming, functional programming etc. It consists of the NumPy library. NumPy is highly powerful and optimized for mathematical operations. In order to take advantage of multi-core computing, all items are written in optimized C or C++.

Algorithm and explanation:

1. Import necessary packages.
2. Read frames from a webcam
3. Create the canvas window
4. Detect the green color
5. Draw on the canvas

CONTRIBUTION OF EACH MEMBER OF THE TEAM

Anagha: I worked on setting up the track bar function in which the trackbars will be used in order to set the upper and lower ranges of HSV required for a particular color. Also created various different arrays in order to hold the particular color points of the colors which will be used to draw on the canvas. I also set up the kernel for the dilation purpose and the colors which will be used as ink for the drawing.

Gopika: I set up the canvas and the process that involves the looping of the camera feed and updating the positions of the trackbar and setting up the HSV values. I also added the color buttons to the frame for color selection. Also set up the methods to stop the application and release the camera and resources.

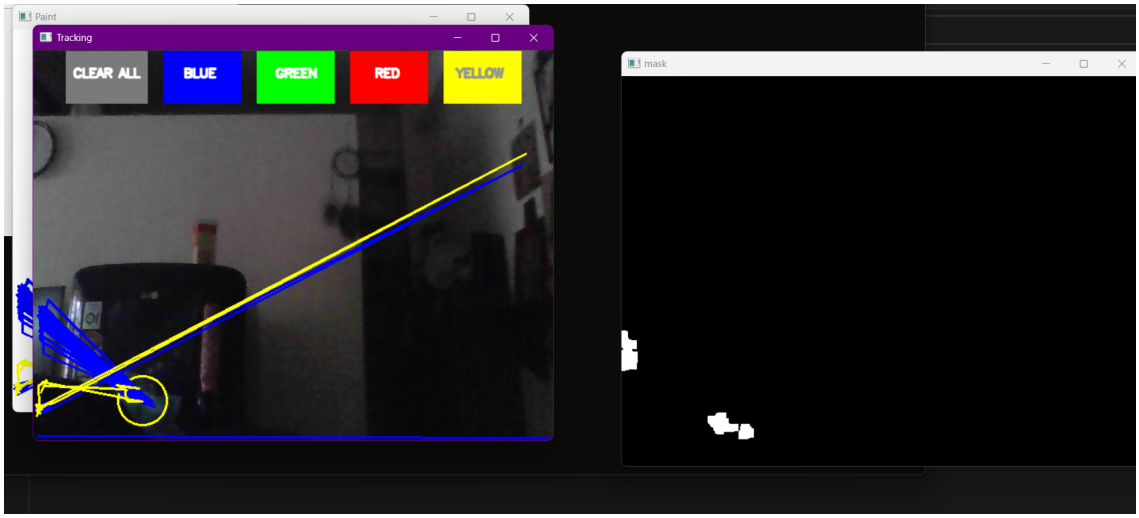
Emeric: I played a part in identifying the pointer and finding its contours. Using this, we can check if the user wants to click on any button on the screen. Also set up the code required to draw the lines of all the colors on the frame and the code to show all the window(i.e. Tracking, paint, mask)

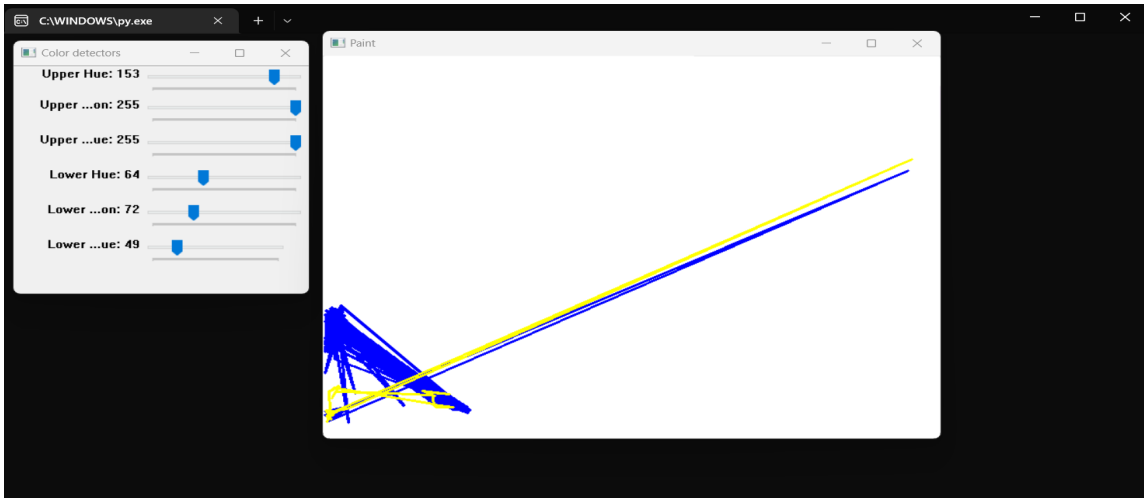
INPUT TO THE SYSTEM

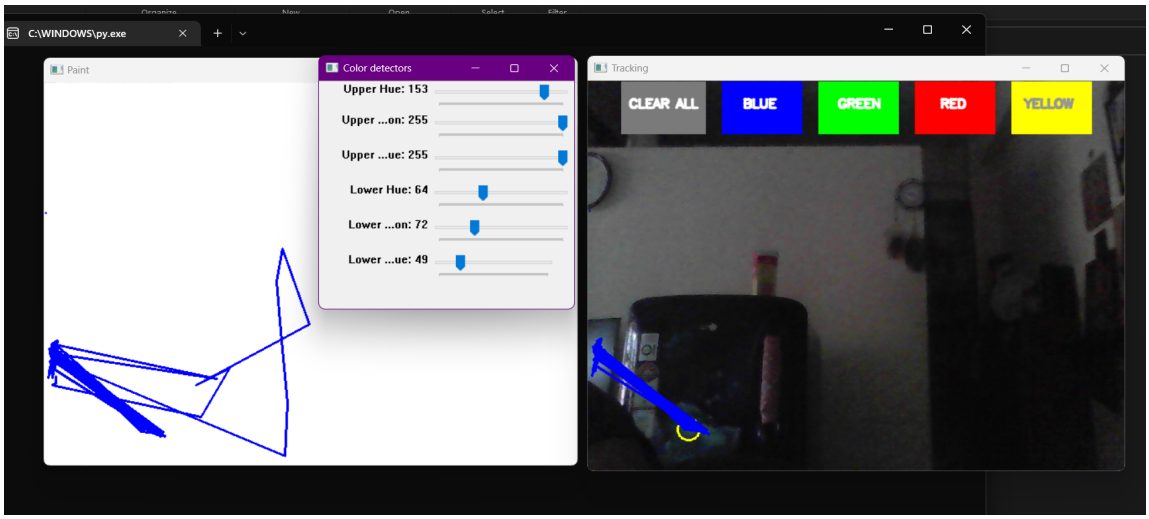
Input for the project consists of whatever we draw on air.

OUTPUT

After debugging the webcam automatically turns on and we get to draw in Front of the webcam and then the output screen appears and we get the output with the help of the HSV values which we have already declared in code. The output here is the text that the person drew on air.




C:\WINDOWS\py.exe
Color detectors
Upper Hue: 153
Upper ...on: 255
Upper ...ue: 255
Lower Hue: 64
Lower ...on: 72
Lower ...ue: 49
Paint


C:\WINDOWS\py.exe
Paint
Color detectors
Upper Hue: 153
Upper ...on: 255
Upper ...ue: 255
Lower Hue: 64
Lower ...on: 72
Lower ...ue: 49
Tracking
CLEAR ALL
BLUE
GREEN
RED
YELLOW