HAND GESTURE CONTROLLED PAINT

NAME: VISHNU TAMMISHETTI
SBU ID: 113082746

CONTENTS:

S. No.	Content	Page No.
1.	Abstract	3
2.	Introduction, and Motivation	4
3.	Procedure (Technical)	5
4.	Validation	11
5.	Future work	12
6.	Conclusion	13
7.	References	14

ABSTRACT

It is undeniable that computers today can do a lot more complex tasks than the ones 10 years ago. Computers have been making and will continue to make our lives easier by doing the hard tasks. But however intelligent or useful a system may be, it should be usable. Intuitiveness and usability play a key role in determining how good a product would do in the market.

With this project of mine, I plan to develop a virtual sketch pad that can be used to paint and sketch on a computer without a mouse and keyboard. One of the most intuitive ways of communicating with the computer is gestures. This application, hand gesture-controlled paint will help in drawing with natural hand movements. I also plan to provide options to change the color of the pen and an option to erase the paintings.

I believe with the present-day scenario; a lot of students are forced to study online; such an application would help professors and students to write and present their peers their ideas with ease.

INTRODUCTION

History tells us that best performance can be exploited from an application when it is made to behave like a human. That is the whole motivation behind AI, and all the fancy neural networks we use today. These are made to replicate the human brain, and they fetch great results. Similarly, as the interaction between us and the computers becomes humane and intuitive, the usability of the application invariably increases. When the way we interact with computers becomes just like how we interact in the physical world we tend to use the application well.

MOTIVATION

MS Paint is one of the first if not the first computer application that I have ever used. I am almost certain that it is one of the first applications that everyone uses. MS Paint provides functionalities which help us to sketch on the computer using a mouse. It is a feature rich application with predefined shapes, colors, different widths for the lines and many more.

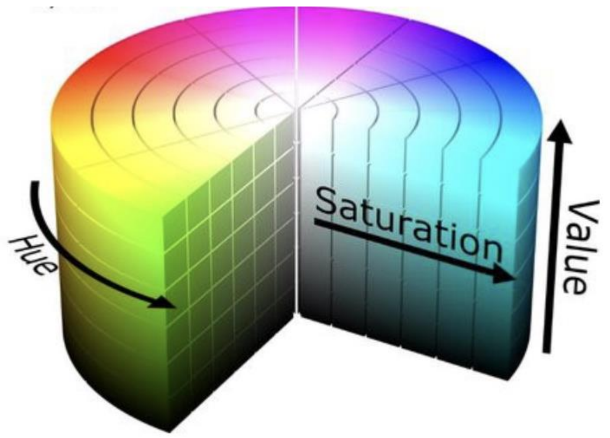
However feature rich this application is, I always felt that it wasn’t very intuitive. Drawing and sketching with a mouse on a desk is not human like and is not how someone draws on a sketch pad. Undoubtedly, more natural way of drawing is using hands. Hence, as part of the course which focuses on the design of computer technology, and bridging the gap between humans and the computers, this is an attempt to make this evergreen application more intuitive and user friendly. This would make drawing on the computer as easy as drawing on the paper.

PROCEDURE

The medium of interaction for the application is by gestures which will be recorded using the built-in web camera or any camera that can be connected to the system. The camera will capture the live feed of the user and provides the frames for further processing to detect gestures and draw the gestures. A video is nothing but a sequence of fast-moving frames. We process each frame and add the gestures to each frame real-time and show the user these updated frames.

HSV Color Space:

HSV color space is used through the application to detect the marker, movements and drawings. Thought the RGB model of colors is the usual way of making and defining colors, HSV color space is generally used in graphic software as it is said to best represent how we humans relate to colors. HSV represents the RGB colors in three dimensions Hue, Saturation and Value. While Hue represents the angle of the color used, Saturation tells the amount of color used. Brightness of the color is represented by the Value.

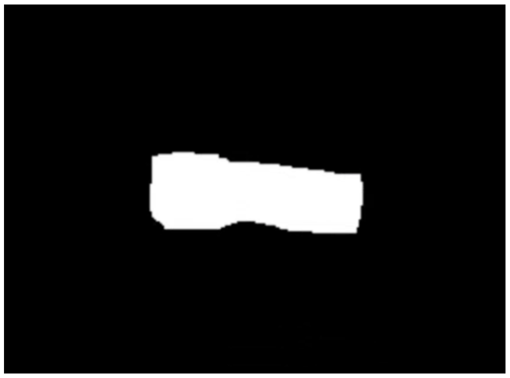


The procedure and background working of the application is as follows,

Steps:

1. Masking:

As the frames are captured, each frame is masked using the upper and lower values of Hue, Saturation and Value to detect the marker and ignore everything other than the marker which will be detected using the upper and lower thresholds of the HSV we define. The first step in creating a mask for the video feed is to detect the points that we want to be detected. In our case, this will be our marker.


This is our object which is masked.

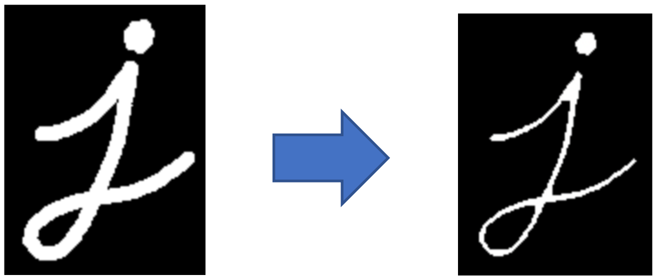
1.1 Filtering:

We filter out the points which fall in the range of the HSV values we defined earlier. This will be our marker. We can do this by using the openCV's inRange method. The next step is to perform erosion with the mask.

1.2 Erosion:

Erosion is the process of eroding away the boundaries of the main object. The object in the foreground will be in white. In erosion, all the pixels right by the boundary will be ignored.

Example:

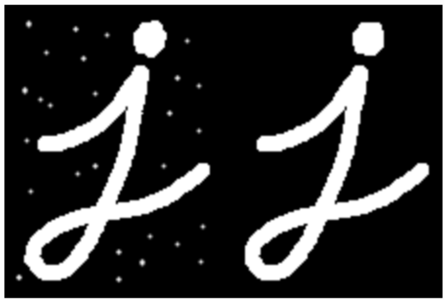


This erosion can be performed using the openCv's erosion method.

1.3 Noise elimination:

The next step is to remove the noise from the image. This is important because we don't want noise to influence the cursor's position. In this step we remove all the unwanted noise so that our mask is only of the marker.

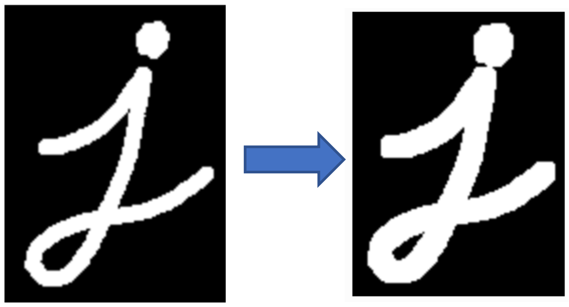
Example:



1.4 Dilation:

After eliminating the noise, we further dilate the size of the object in the foreground. Like observed above, erosion shrinks our object. This step is performed to ensure that we enlarge the size of the object in foreground. Since we also removed noise in the earlier step, there is no possibility of dilating unwanted portions and our marker will be the only object dilated. This operation can be achieved using the method openCv's dilate method.

Example:

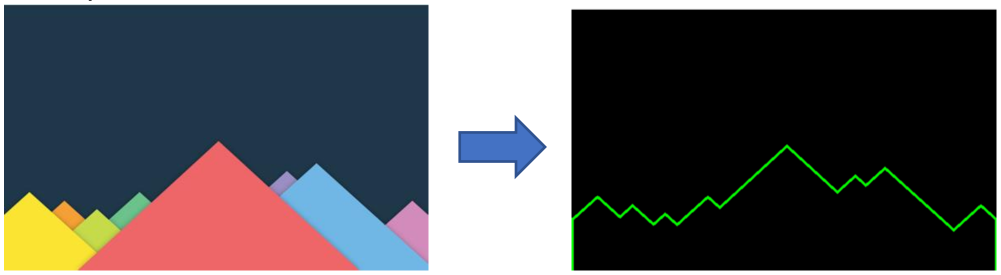


It is to be noted that the noise elimination and dilation can be swapped without effecting a considerable change in performance as erosion step also attempts to remove the noise.

2. Finding Contours:

A contour is nothing but a curve joining all the pixels which have same color or the pixels of same intensity. It can be used to get a boundary or outline of the object we are trying to detect. We pass our masked to this method to find contours.

Example:



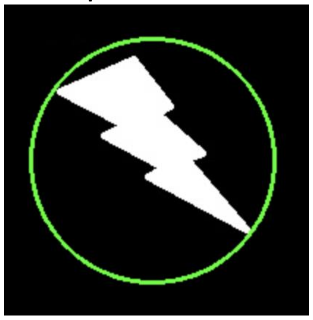
As we can observe from the above picture, using the contours, we can draw an outline of our object.

However these contours are a sequence of lot of points. To use as a marker, we cannot make use of all these points, but we need a single point which will be representative of these contours. Hence we find the centroid of the object's enclosing circle.

2.1 Minimum Enclosing Circle:

Minimum enclosing circle is a circle which is the smallest circle that can be used to circle the entire object. Now we get the center of this minimum enclosing circle to be our cursor, and we draw a circle around our cursor for ease of use.

Example:



3. Moments:

Finding the center of standard objects is pretty straight forward, however, we are dealing with blobs. A blob is a group of pixels which share the same property. Our cursor object will be represented as a blob. This will not be of a perfect shape to calculate the centroid for our use.

We can calculate this using **Image Moments.** Image moment is a weighted average of pixel intensities in an image. With this, we can calculate the centroid of our cursor object.

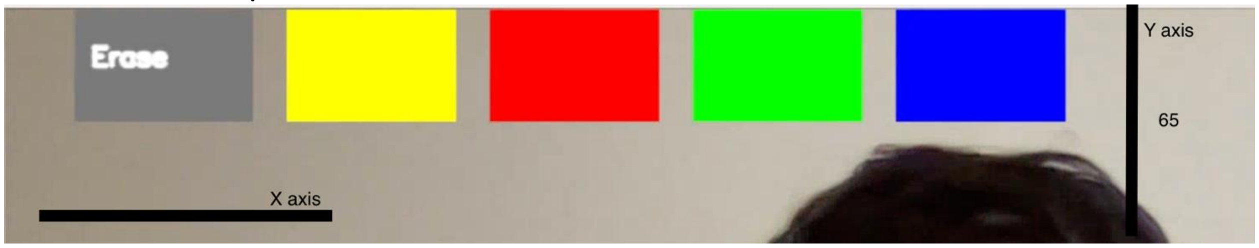
Formula to calculate the centroid is given as follows,

$$C_x = {}^{M_{10}}/_{M_{00}}$$
$$C_y = {}^{M_{01}}/_{M_{00}}$$

4. Sketching:

After performing all the above steps, we will obtain the centroid of our cursor. This will be the pointer with which we will be drawing, erasing and interacting with the application. All the above steps are performed for each frame. The centroid is calculated in every frame and the color state is maintained.

At each frame, we check the position of the cursor. If the **y** coordinate of our cursor is less than or equal to 65, it means that the cursor is now in the menu area.



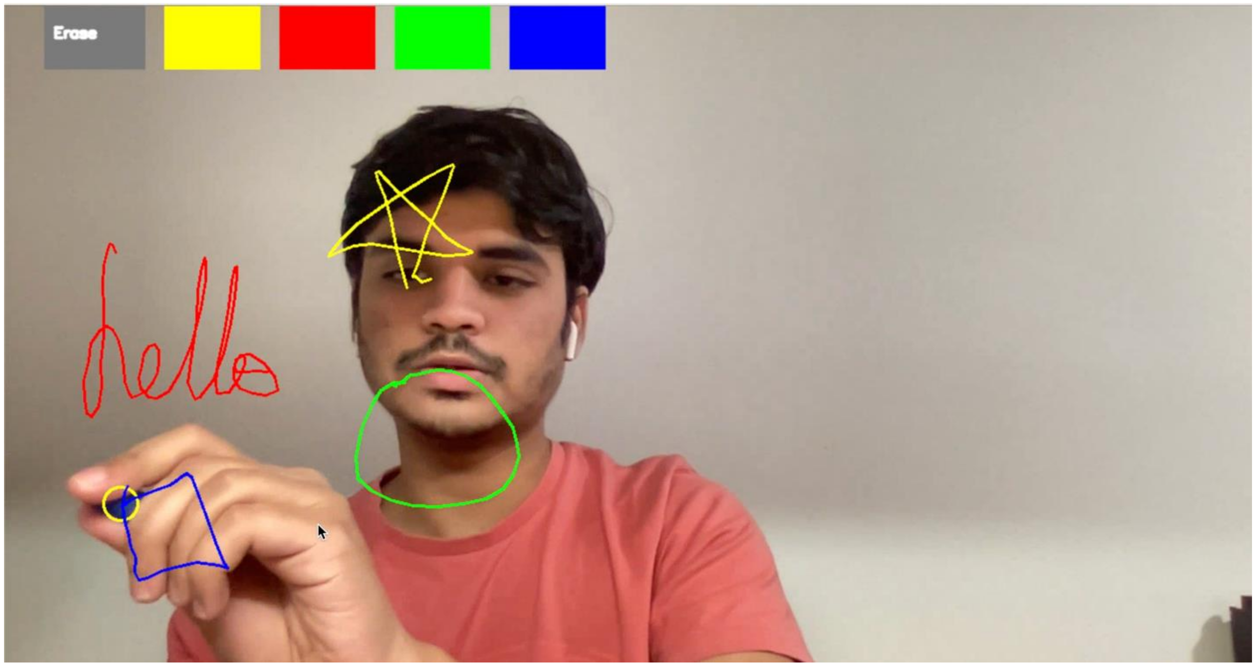
Like observed in the above picture, the menu area will be above 65 in the Y-axis. When the position is above 65, we check the x-coordinate of the cursor. Based on the X-coordinate position of the cursor, we check which option on the menu our cursor is on.

The X-coordinates are as follows,

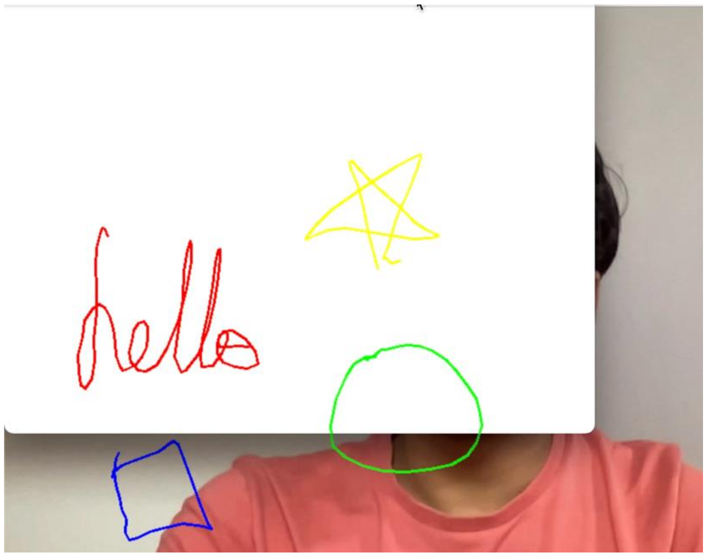
X – Coordinate	Menu option
<= 40	Erase
> 40 and <= 160	Yellow
> 160 and <= 275	Red
> 275 and <= 390	Green
> 390 and <= 505	Blue

When the X – Coordinate is less than 40, we call the Erase function, and everything on the screen will be erased. When it is less than 160, it will be yellow, and similarly for 275, 390, and 505, the colors will be Red, Green, and Blue.

In each frame, we draw the pixel using openCV's line method.



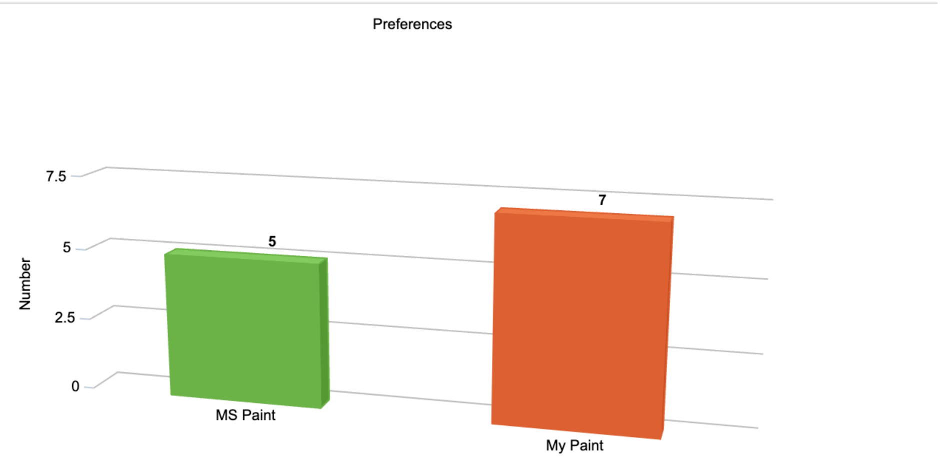
This will also be simultaneously drawn on the plain sketch pad.



VALIDATION

For validation, I have approached 12 students (My fellow students) for testing the application. I have given them each 10 minutes to get familiarized with both the MS Paint and my virtual paint applications before asking them to test it.

The major feedback that I received was that my virtual paint, though was easier to use, it is lacking a lot of features like predefined shapes, color fill options, eraser and a few more which are available in MS Paint.



Although 7 people have said My Paint is more intuitive to use, everybody pointed out that it is not having enough features, while in MS Paint they liked the wide variety of options it provides them.

Based on the feedback and my observation, I believe my application has a lot of scope for improvement to make it usable.

FUTURE WORK

I believe Virtual sketching using hand gestures has a lot of scope and in general applications with gesture control will see major advancements in the near future.

I would like to let more users other than my peers try out this application to get proper feedback. Based on the feedback that I got so far, I will add more features to my application. Features like drawing predefined shapes and erasing only a particular part of the sketch would be the next priority. When we draw with hand, it is difficult to get some shapes right. Hence having predefined sketches of such shapes would prove to be useful. Another limitation is that I will have to use an object to draw in this application. I plan to make this work only using hands without any objects. There has been decent research in this area and I believe Mediapipe is something that can be integrated into this application to do that.

This application can also be used in different ways with small tweaks. We can integrate this with some additional features into the online meeting applications we use today like Zoom, Meet and Teams. This can help us with giving presentations and also teaching in case of online classes. This will keep things as natural as possible like in a classroom. With the time in which we live, we are moving closer every day to a digital world where the interaction would all be digital, and these would be a step in the right direction I believe.

CONCLUSION

The MS Paint application though feature rich, is not very intuitive to draw on. Using a mouse and keyboard to sketch is not the best way to draw. There is nothing more intuitive than gestures for drawing. Hence, I developed an application which tracks human gestures, and makes it act like a cursor with which we can draw.

Hence as a student of HCI, I developed an application aimed at improving the interaction between the application and user. This application keeps track of an object with which we draw and sketches on the canvas using the hand movements we make. This will help in creating more accurate sketches than the ones done with MS Paint. Apart from being just a virtual painting application the procedures used in this application can also be helpful if we integrate these with other cutting-edge applications like Zoom, Meets and Teams for example.

From the study made on 12 people, the major feedback was that though this application is more intuitive to use, it lacks some major features. Therefore, there is a lot of scope for improvement in this application but this is a step in the right direction.

References:

- https://en.wikipedia.org/wiki/HSL_and_HSV
- http://www.robogrok.com/2-2-1_Masking_and_Image_Segmentation.php
- https://dl.acm.org/doi/pdf/10.1145/258734.258818
- https://ai.stanford.edu/~syyeung/cvweb/tutorial1.html
- https://towardsdatascience.com/computer-vision-the-importance-of-filtering-c0d9640c6e72
- https://en.wikipedia.org/wiki/Erosion_(morphology)
- https://homepages.inf.ed.ac.uk/rbf/HIPR2/erode.htm
- https://docs.opencv.org/3.4/db/df6/tutorial_erosion_dilatation.html
- https://docs.opencv.org/3.4/d5/d69/tutorial_py_non_local_means.html
- https://www.bogotobogo.com/python/OpenCV_Python/python_opencv3_Image_Non-local_Means_Denoising_Algorithm_Noise_Reduction.php
- https://www.geeksforgeeks.org/erosion-dilation-images-using-opencv-python/
- https://docs.opencv.org/3.4/d4/d73/tutorial_py_contours_begin.html
- https://docs.opencv.org/3.4/dd/d49/tutorial_py_contour_features.html
- https://docs.opencv.org/3.4/d0/d49/tutorial_moments.html