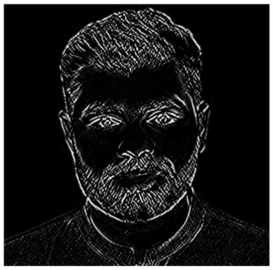
Face Segmentation & Edge Detection

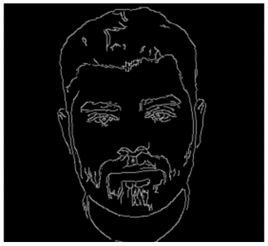
To perform edge detection and face segmentation I used Python and OpenCV. To achieve Edge Detection I used Canny Edge Detector, Sobel Edge Detector from OpenCV & My Custom Function so that I could compare the outputs. To implement segmentation I used Kmeans algorithm. All tasks were performed on Google Colab. My Goal was to include my facial features but with Edge and little noise.

First, I uploaded my image in Google Colab using the files.upload() method and used CV2 from open CV: to read the file, convert the file into gray & Gaussian Blur. To further remove the noise from the image 'cv2.filter2D() 'function was used, as it helps convolve the image with a 5x5 averaging kernel. Afterwards, I applied Sobel, Canny Operator, My Custom Method on the image and finally applied Segmentation. Now, I will explain these techniques in the following,

Sobel Edge Detection: For this operator, cv2.Sobel function is used. This function helps compute gradients along the X and Y axes separately and also on a combined X and Y plane. This helps highlight different aspects of edges in the image. The resultant output of the XY plane created an edge but also focused on my other facial features, image .

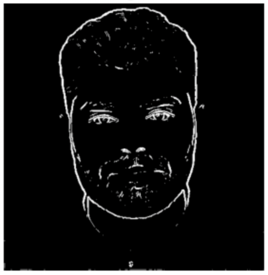


Canny Edge Detection: On the preprocessed image, canny edge detection was applied. This method is also from the OpenCV library. It takes 3 parameters: image, low threshold, high threshold. Here I had to experiment with the threshold values so that I could get a good edge of my face. My optimal values for thresholds are: **low: 40 , high: 135**. Good Edge with little noise and covering most of my facial features.



Canny Edge Detection

My Custom Edge: I applied my own code to implement edge detection in Python. To perform this, gaussian_laplace was used for average filtering, smoothing the image and python's convolve function, this helped to detect image edges. Centering to detect the edge was done by comparing the values of gradient_magnitude to threshold values(which were experimented, here the threshold value was set to **0.175**). Shows a good edge but needs more tuning.



Custom Edge Detection

Segmentation Using KMeans Clustering Algorithm: For this first the image was segmented into different colors so that there is differentiation of areas according to pixels. Then 5 clusters were used on the masked image so that each cluster has its own area of pixels whose values we can change to properly have a segmented final result of images. My segmented image shows that the background was changed to black, face was labeled into 2 distinct areas: one with hair(blue color) and the other area with red color.