Shashank Reddy
IMT2018069
shashank.reddy@iiitb.org

Final Report
AI 825 - Visual Recognition

2021-02-08

1 Counting the number of books on a bookshelf-

The following are the various methods I used for the given problem.

1.1 Preprocessing

First I did some basic preprocessing such as histogram equalization and blurring. After this I applied the Canny Edge detector. I tuned the parameters for all the functions based on the given image, thus the process is not very generalizeable.

1.2 Finding Countours

I used the findContours() method in Open CV to find the contours. After this, I looked for contours that are described by 4 points (implying that it is a rectangle) inorder to detect the books. This method did not prove quite useful as it was detecting many contours described by 4 points that were necessarily not rectangles, and it failed to detect a single contour that bounds a book well.

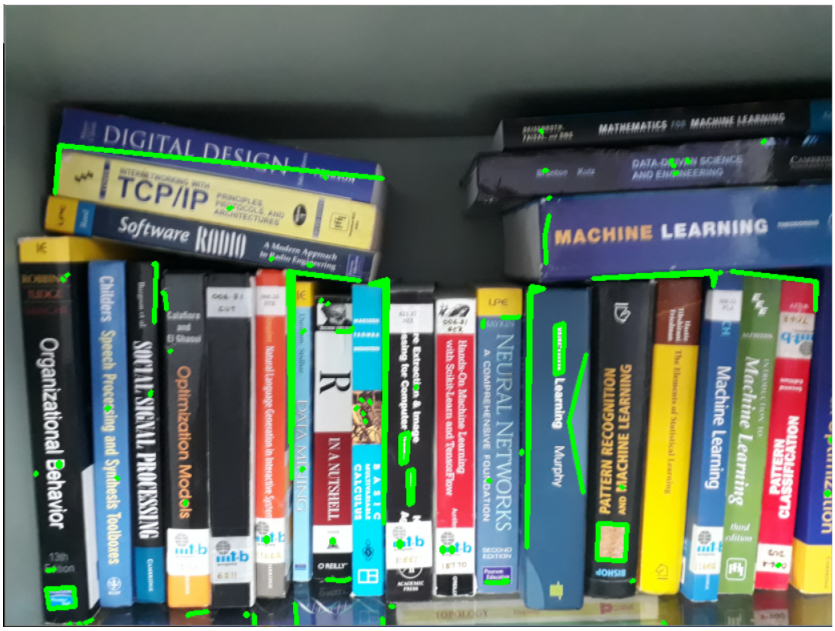

Figure 1: Contours detected

1.3 HoughLines

After this, I used the HoughLines() method in OpenCV. The idea being that it will detect parallel boundary lines for the edges of the book. This method detects the required lines well, however we get multiple lines that are clustered together when there are thick lines.

Figure 2: Initial image obtained by basic HoughLines

1.4 Filtering close lines

To remove lines clustered together, I wrote a basic filter function. This function takes the euclidean distance between the (rho, theta) values of 2 lines and deletes one of the lines if the distance is above a certain threshold. This threshold was obtained by trial and error. After this, I approximate the number of books roughly as the number of lines detected.

Figure 3: Image obtained after filtering

1.5 Clustering

To get a deeper understanding of these lines, I decided to cluster the lines (in the (rho,theta) space), to find the optimal number of clusters, I used the elbow method. In this method we basically plot the number of clusters vs a loss function, in this case inertia. Inertia is the standard loss function that we use in k-means clustering, that is the sum of squared distance between a point and it's cluster center. In this graph, we look for a point after which the graph becomes linear, this point is called the elbow and is assumed as the optimal number of clusters. From this, I found that we can find 3 cluster centres. My understanding from this is that we can see that the books are aligned in 3 orientations, vertical, horizontal and slightly angled to the right and horizontal and slightly angled to the left.

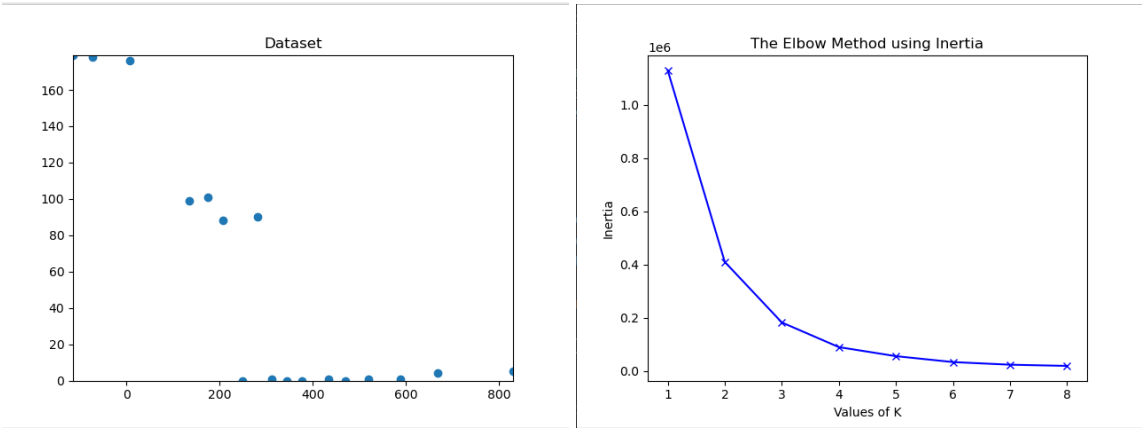

Figure 4: Lines in the (rho,theta) space; Inertia vs Number of clusters

1.6 Segmentation using Gabor Filters

I tried to segment the image using gabor filters. The results of this are not very promising thus I chose not to pursue this further as I expected that it would be able to detect the books as a separate entity. This is probably because the titles of the book are highly textured.

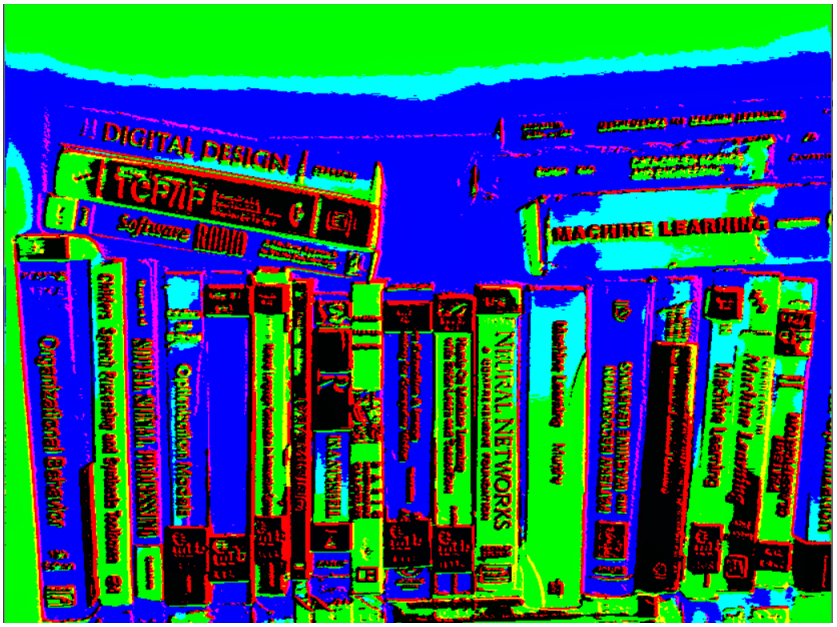

Figure 5: Texture based segmentation

2 Face detection

The following are the various approaches to this problem.

2.1 Segmentation based on skin color

To detect the face, I basically detected pixels that had the color of the skin. For this I specified a range of values as the skin color and used the inRange() function in OpenCV to pick the pixels that had this color. For this, I first converted the picture from RGB to a different color space. I did this because I found it hard to find the range for the color of skin in this space. I used 2 color spaces for this, HSV and YCrCb. This also segmented regions in the background as part of the face as it fell within the range. This was more apparent in the HSV space as it segmented the bat grip as a part of the face.

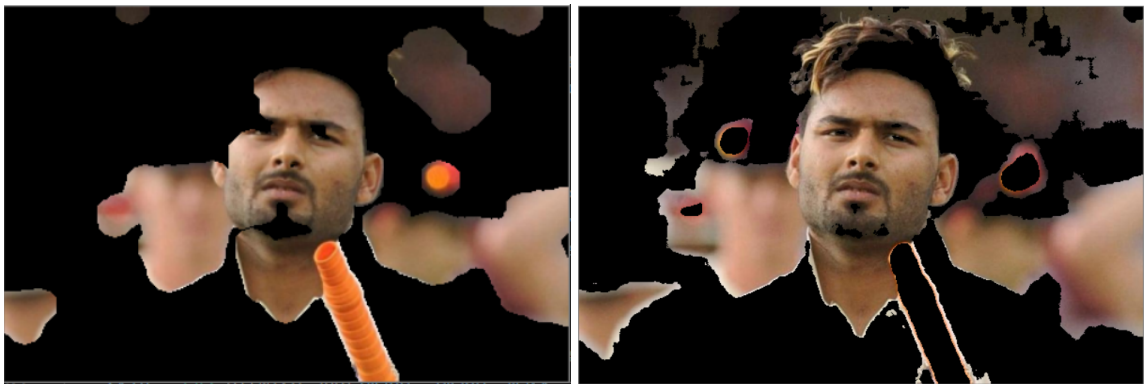

Figure 6: Skin segmentation in the HSV space; Skin segmentation in the YCrCb space.

2.2 Removing the background

Since the pixels in the background are being selected as a part of the face, I decided to try and blur or remove the background so that these pixels would not get considered. I did this by using a custom function that uses the findContours() in OpenCV to find the largest contour by area. I then consider all the pixels within this contour as the foreground and the rest as the background. I then ran the skin segmentation algorithm on this image.

Figure 7: Skin segmentation after removing the background: HSV Color Space; YCrCb Color Space

2.3 Segmentation using Gabor Filters

I tried to segment the image using gabor filters. First when I applied this algorithm without removing the background it didn't segment the image well, however after removing the background we can see that it clearly segments the hair, the jersey and the face. However again, it segments the grip of the bat as the face, which is the same as the problem encountered in the HSV color space.

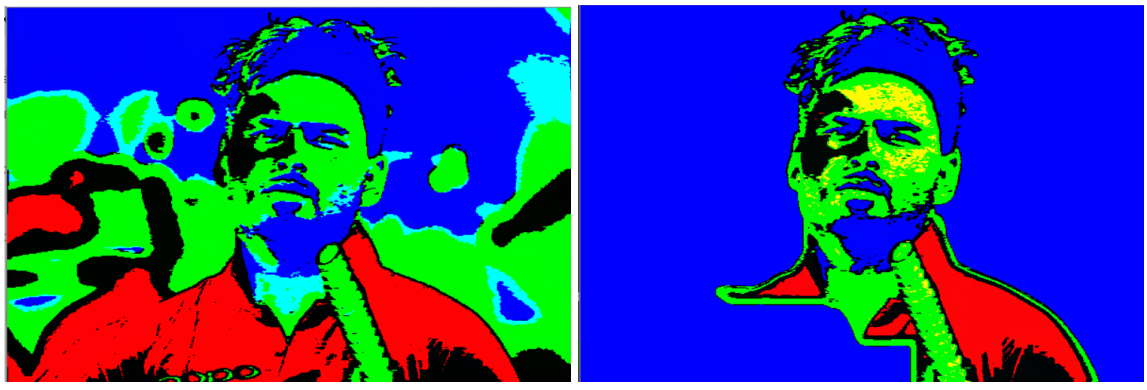

Figure 8: Gabor Filters segmentation: without removing the background; after removing the background

3 Applying the algorithms on different images

In both these algorithms I tuned a lot of the hyper parameters very specific to the image. In the case of the first problem, some of these include the size of the Gaussian kernel, the thresholds for canny, the limit to filter the hough lines, etc. and in the case of the second problem these include hyper parameters mentioned in the remove_background function, scale of the image etc. So when I applied these algorithms to different pictures initially they produced very bad results however after tuning all these hyper parameters the algorithm performed rather well.