Assignment -1 Report

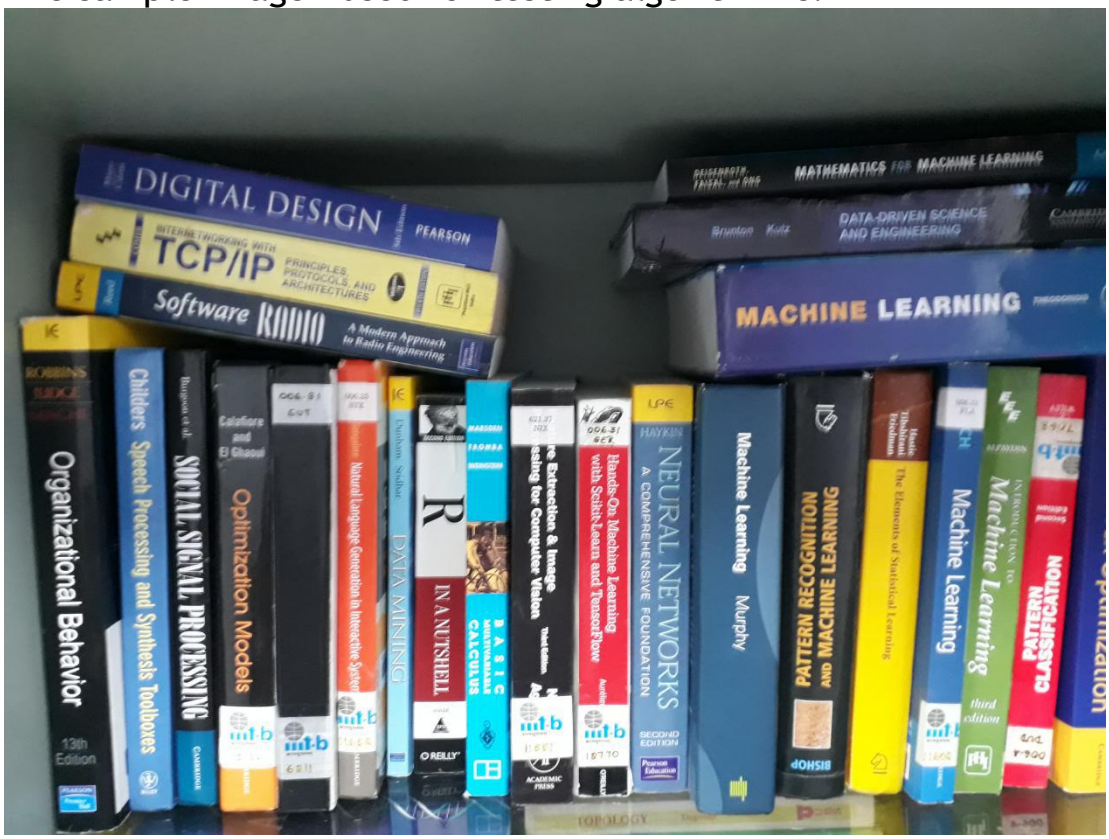
Ellanti Bharath Sai

IMT2018022

1. Detecting Number of books in the given image of book shelf.

My approach is to find edges of each book and count them to find number of books in the image.

The sample image I used for testing algorithm is:

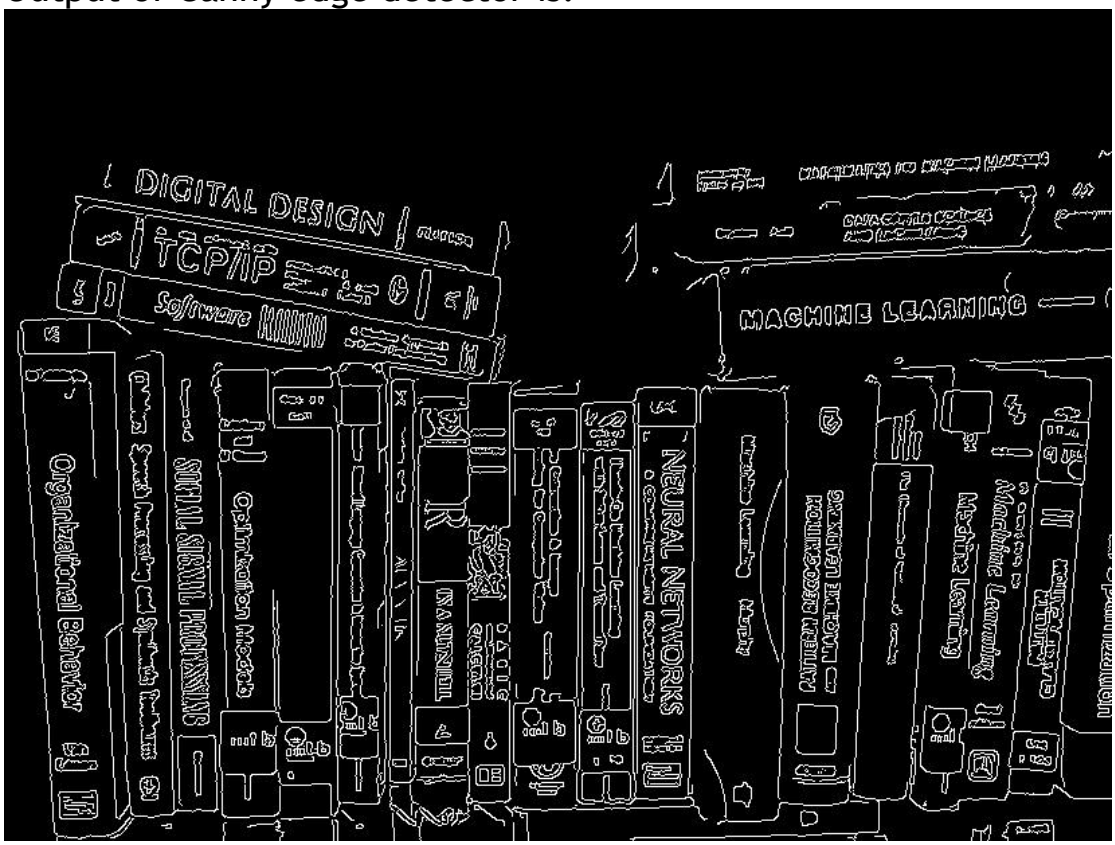


Initially before proceeding to the algorithm I first did some preprocessing to the image. The image is first converted to gray

scale from BGR format as opencv reads images in BGR format rather than RGB format. The reason I converted the image to grayscale is because, to count the number of books we need to know the edge of each book. To find edges of image we check the change in intensity of pixel values, hence color has no important role to play, so for simplicity of algorithm we changed from color to grayscale format.

I did some smoothening on the grayscale image to decrease noise from the image. Then I used Canny edge detector to detect edges.

Output of Canny edge detector is:



As we can see in the above image text is also detected as edges, but for us this is an unexpected behaviour, we can increase smoothness of the image to decrease noise, but increasing smoothness also leads to loss of information about edges of the book. Hence by doing different parameter checks for smoothening and canny edge detector above is the possible better image.
By checking different threshold values for canny edge detector, finally taking the median of the image and then constructs lower and upper threshold based on a percentage of this median gave better results.

As we can see there are small holes in between the edges of text, this causes noise in the image, hence to remove this we closing morphological transformation.

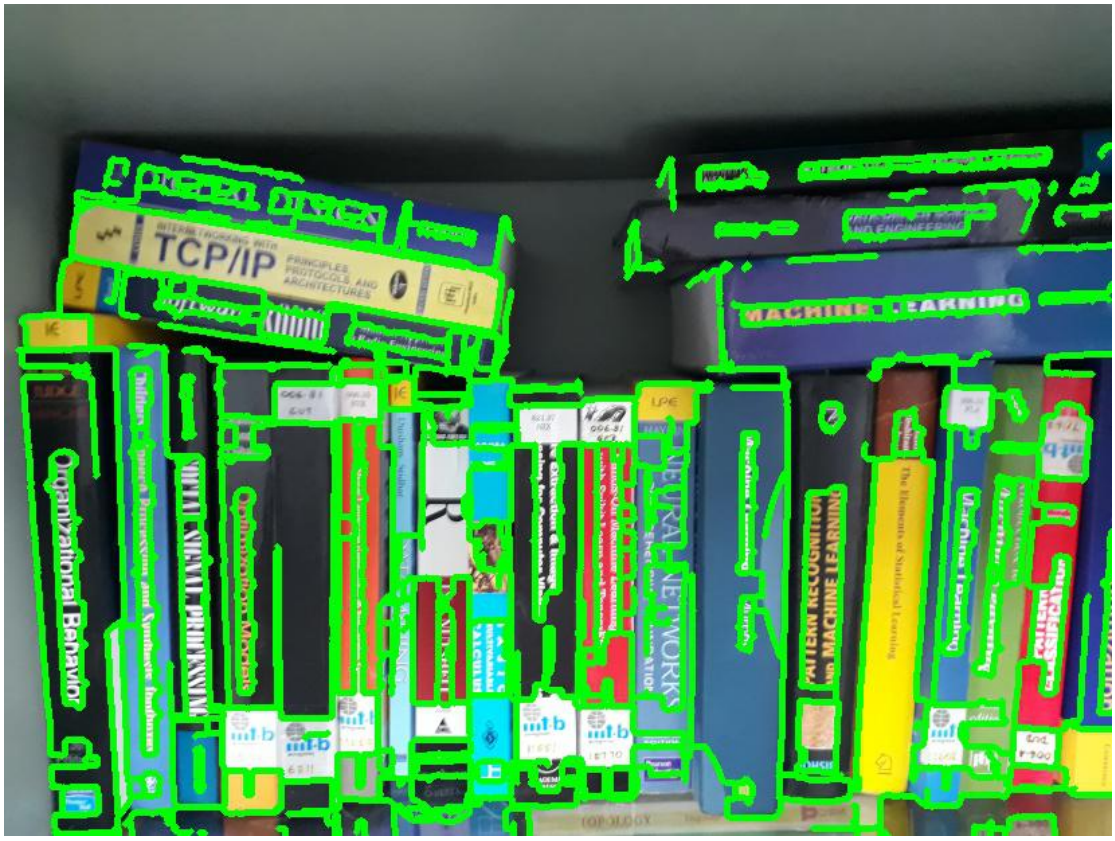
The output after morphological Transformation is.

We can see in the above image the text is removed from the image.

To get number of lines from this canny edge image I used Contours technique.

Contours are the lines that joins two points with a function defined in two variables and that function should have same value along the line joining the two points. In image processing Contours joins the lines along the boundary lines of an object that has same intensity. So by using contours we can detect the boundary of an object, size and shape of an object.

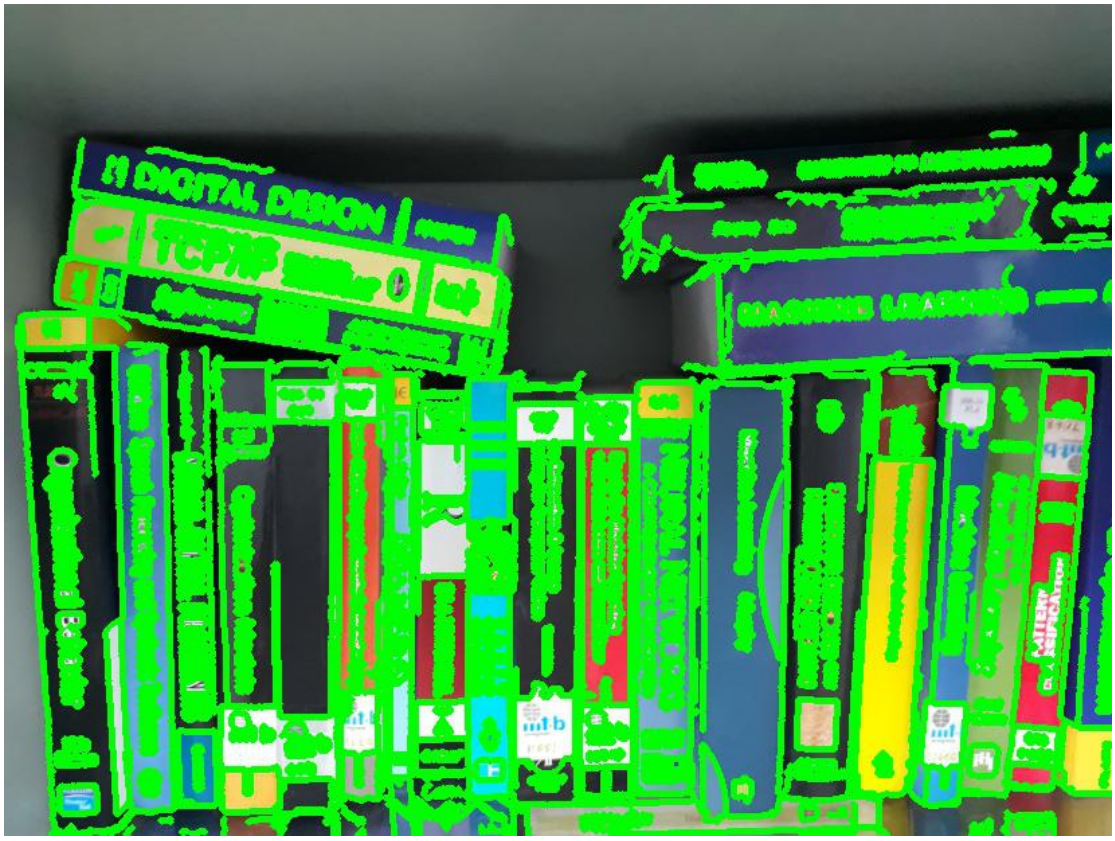
The contour image of above image is :



As there is lot of noise in edge detection so contour failed to find the edge of the books correctly. I have tried different ways to draw contours like restricting the area of contour, height of boundary, number of vertices in the boundary. But all of them failed to give expected output.

This is because by applying morphological transformation, I removed text from image but I created boundary of that text which makes the contour confusing to decide which is book and which is not, so morphological transformation is not a good way, hence I removed it after.

Output of contours after removing morphological transformation:



Even this becomes more worse than before. As it tries to draw contours on text in depth. So I think Contours is not a good option in detecting the edges of book. Because there is text in the image and contours can be curves also no restriction that they should be lines. Also while detecting edges of book we failed to get the actual boundary of the book, i.e we can see in the output of canny edge detection broken or distorted boundary of a book.

Then I moved to Hough Lines Transformation to detect edges. Hough Lines can detect the shape of an object even if it is broken or distorted a little bit.

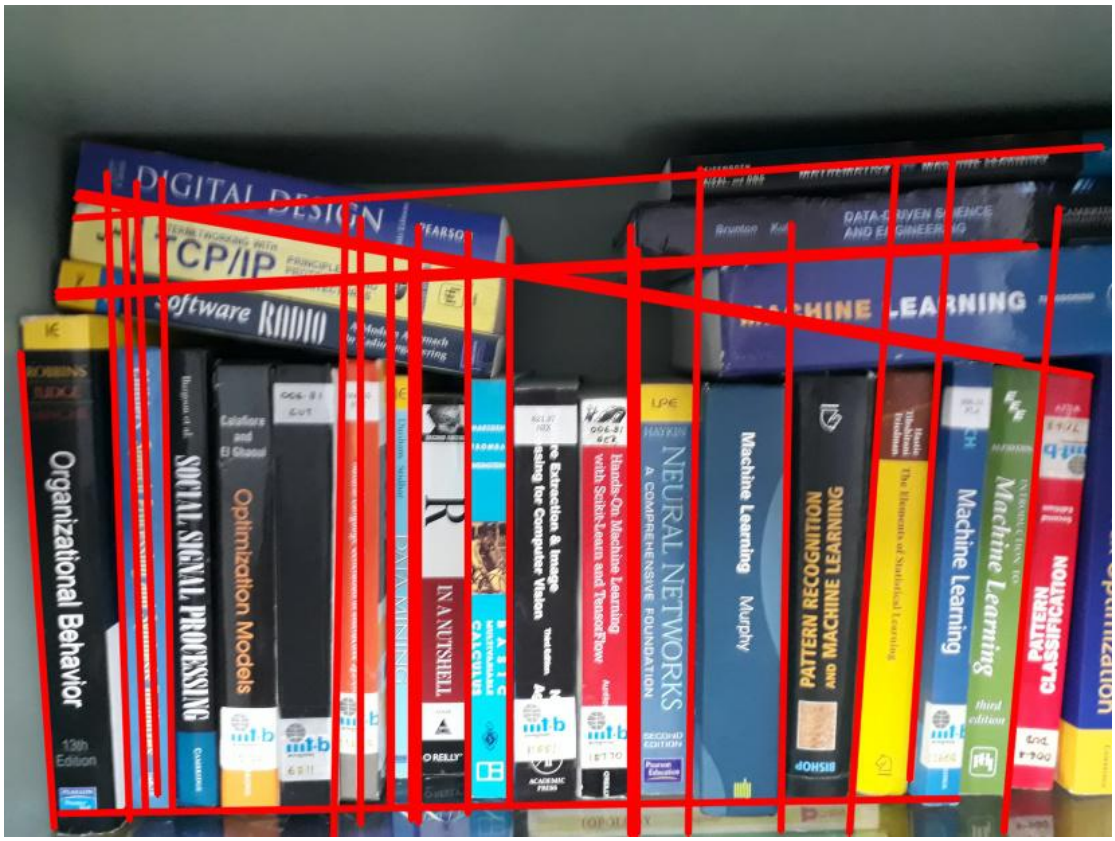
There are two types of Hough Line Transform:
1. Standard Hough Line Transform
2. Probabilistic Hough Transform

I tried both the types to check for best results and I almost got same results in both, but Probabilistic is better than standard.

With a gray scaled image smoothened a little and edge detection with Canny edge detector and no morphological transformation is applied. The output of Hough Lines is:

Standard Hough Line:

Probabilistic:

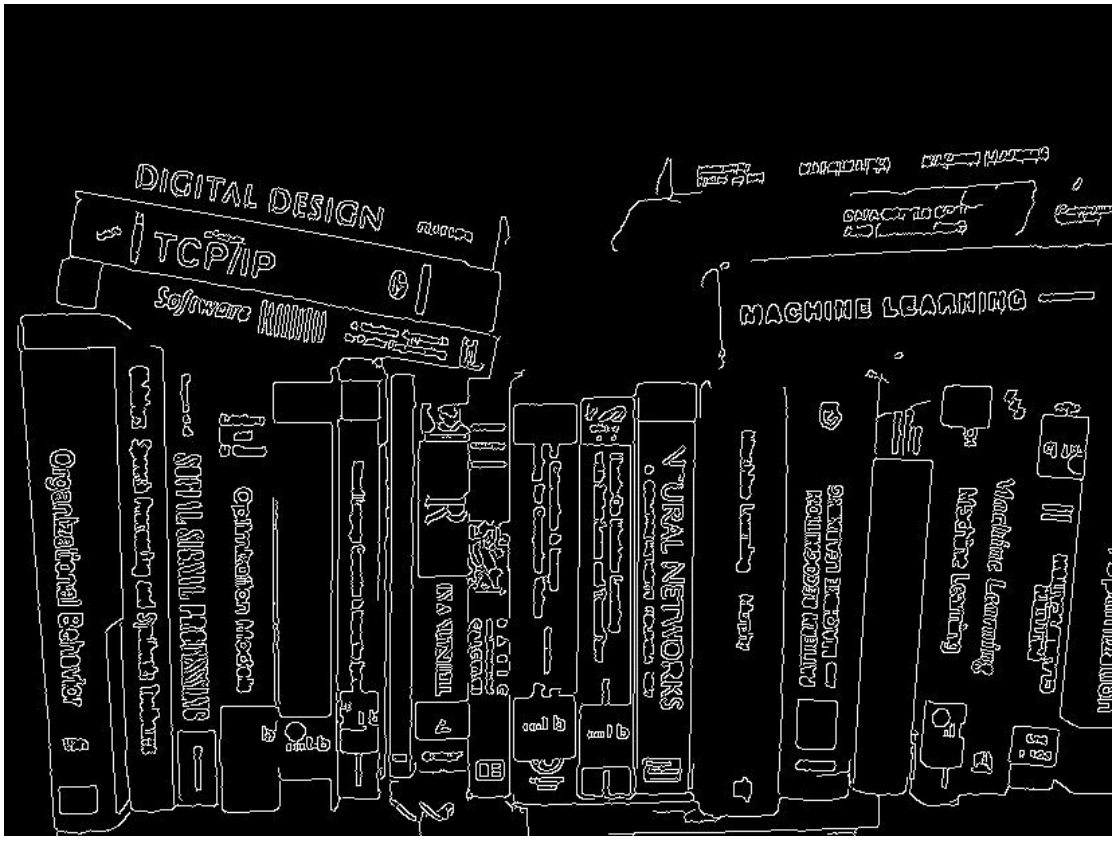


By comparing above images Probabilistic has detected the edges of book more accurately. After changing the threshold and parameters with different values above is the best solution.

These outputs are far better than contours and are actually close to the solution.

I tried different ways to optimize the solution. I did morphological transformation directly on the image before canny edge detection. The output of morphological Transformation is :

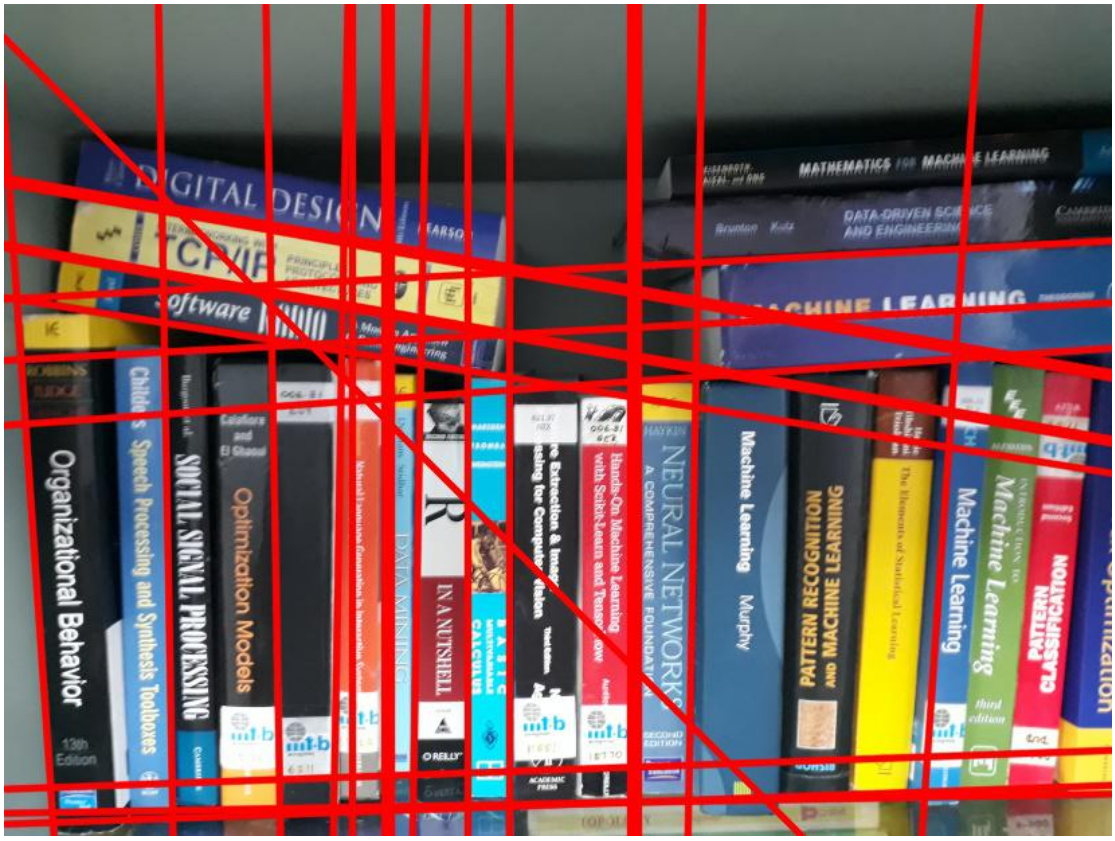
The output after smoothening the image:



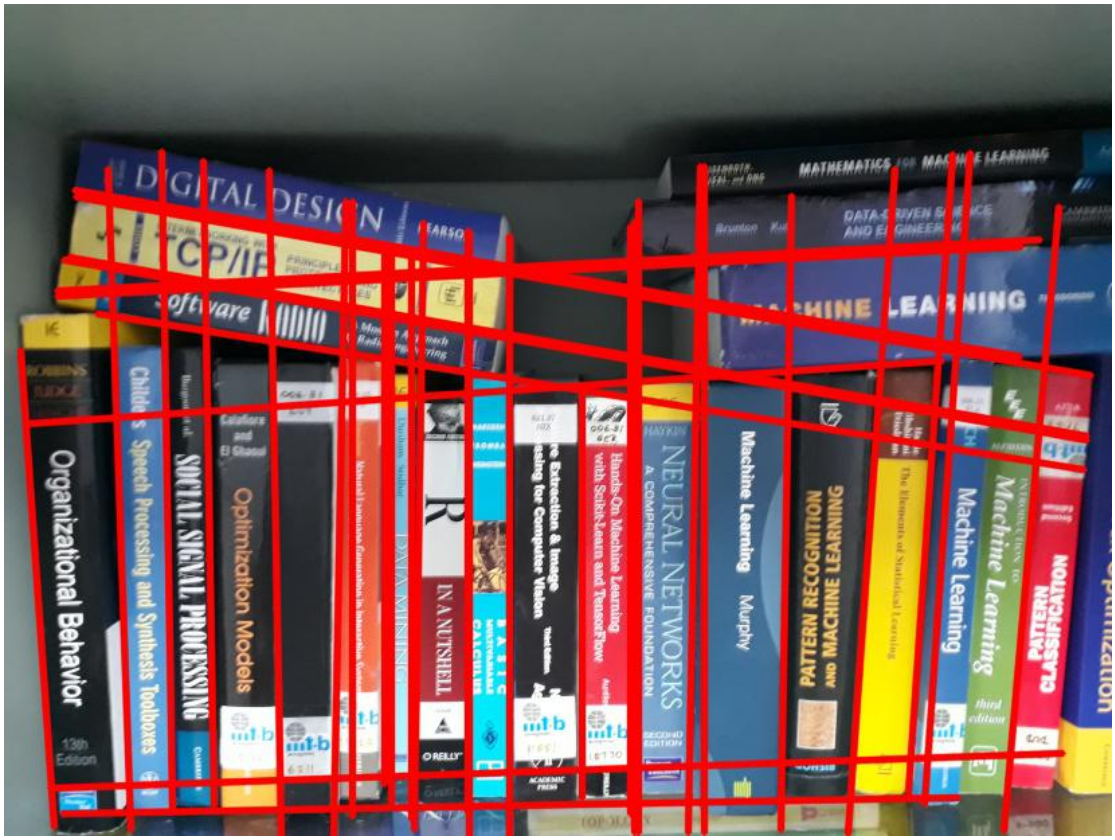
After observing the both images morphological Transform also helps to decrease the noise in the image hence instead of smoothening I used morphological transformation and I got better results than before.

The output of Hough lines after using morphological tranformation on image and removing smoothening of image are:

Standard:



Probabilistic:



There is actually an improvement in detecting edges from before pictures. In the above probabilistic picture we can see that there is line between almost all books in the image. But we have problem of having two or more lines at same place. After different combinations of parameters above is the best solution.

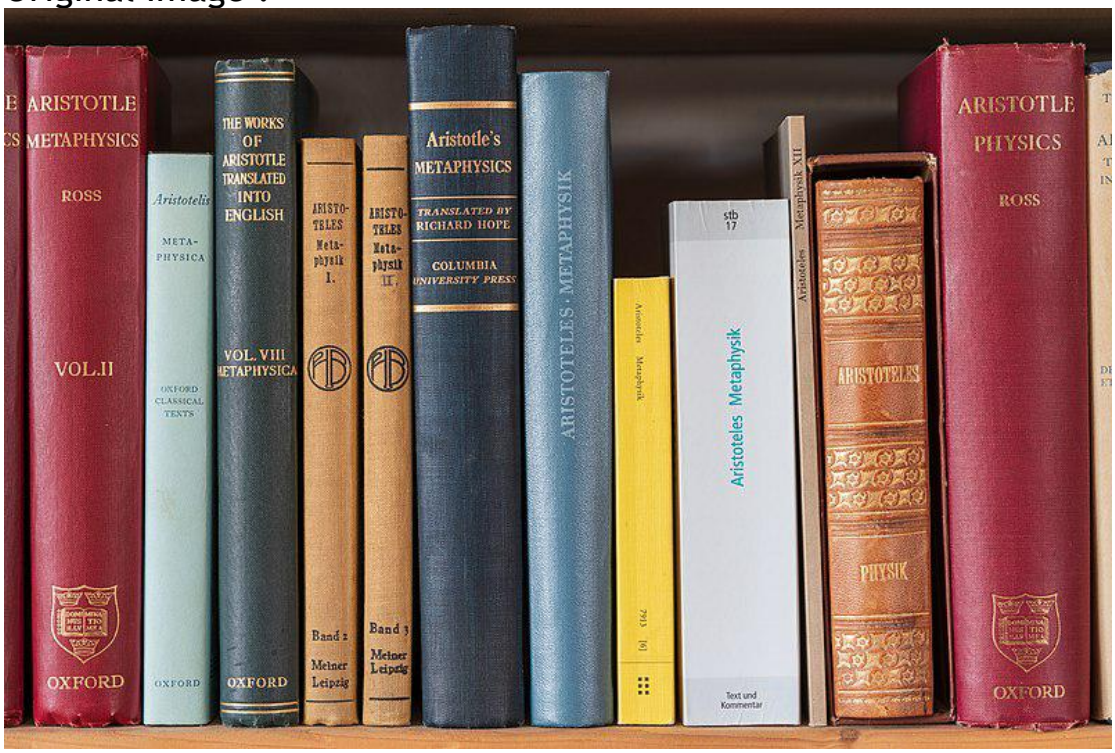
Finally to count no of books in the image I counted number of Hough Lines. At the best case there should be line between every two adjacent books and hence number of books would be number of lines.

From the above pictures I got output as 27 books for Standard and 35 books for Probabilistic. Actually there are 24 books in the image.

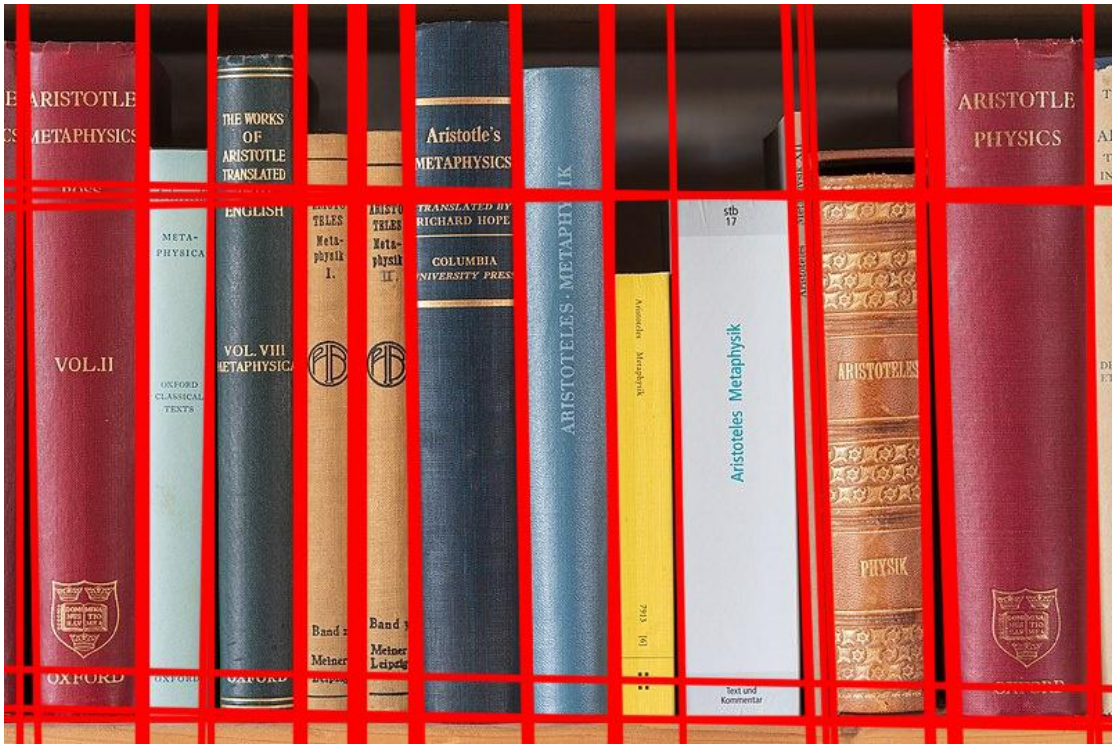
Conclusion:
Even though Standard output is close to actual output but the edge detection is not efficient in Standard. Whereas Probabilistic is good in detecting edges of book but due to repeat of lines at same place it didn't gave the expected output.

For other Image

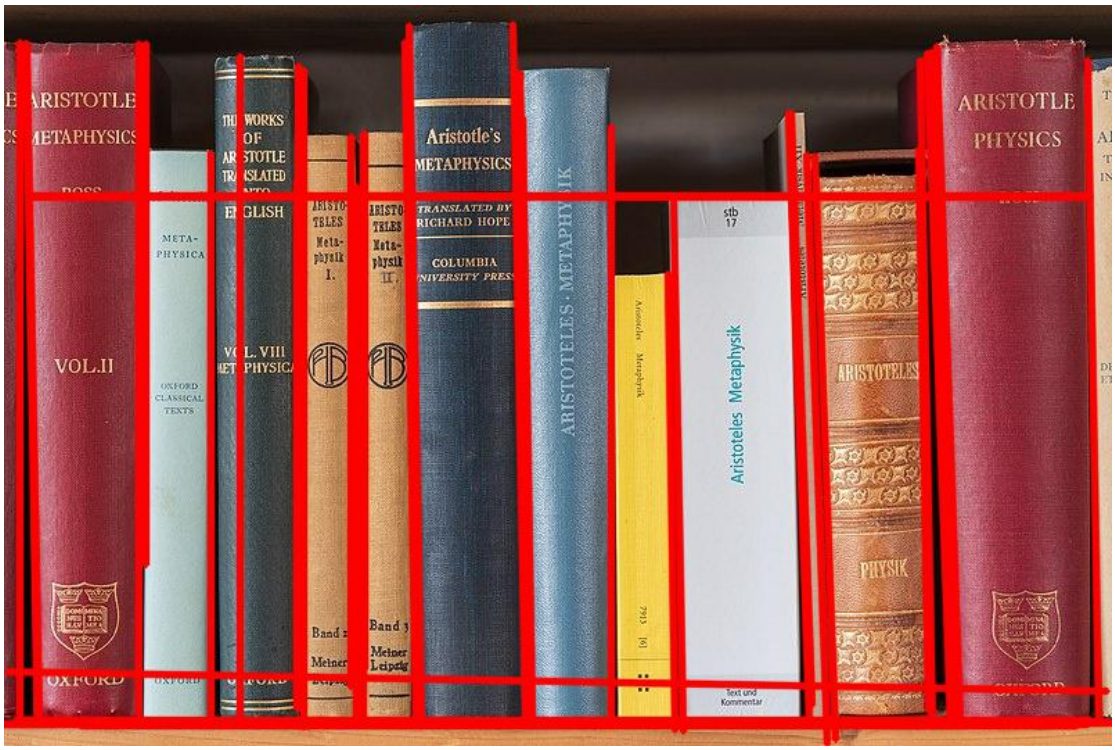
Original Image :



Standard:



Probabilistic:



In the above images the books are differentiated nicely but cluster of lines are formed at same place.

2 Detecting The Face pixels in an image.

The sample image used for testing:

Initially I tried to detect edges of the image and draw contours to see the boundary of image.

The output of Canny edge Detection would be:

We get the perfect edge of the image without any background noise, also we get the edge of the bat. If we try to draw contours for this edges we get

we see that we can detect face also some parts like bat and t shirt are also detected. But this is not a good way because in this image except Rishabh Pant all background is blurred, hence we could detect only the edges of pant image and draw contours. This won't be the case for all images. For some images background cannot be blurred and hence we get noise in the detection.

Then my new approach is to differentiate the pixels that have the color of skin tone and that do not have.

To do this initially I loaded the image. Here I didn't change the image to gray scale since here color of the pixel has the main role in detecting the face pixels.

I then changed the image from BGR format to HSV format since HSV has less noisy color information than RGB. I read in many online pages that HSV is more preferred than RGB for image processing, human detection etc.

As skin tone varies for different people, I decided to take a range of colors as skin tone. I fixed a lower bound and upper bound for skin tone and tried to filter the pixels that fall in the given range. Then I used this mask that I obtained from filtering the original image to place on the original image so that detected face pixels are only shown.

After filtering mask would look like:

After applying this mask on the original image final Output would look like :

We could see in the above with the face some back ground noise is also captured since they fall in the color range we specified. I tried different ranges for skin tone and above is the best one.

I tried this algorithm on some other picture and the result is

Original Image:

After running algorithm on this image :

Almost all face pixels are detected along with some noise.

References

Part -1

Morphological transformation on the image :

https://docs.opencv.org/master/d9/d61/tutorial_py_morphological_ops.html

Contours:

https://docs.opencv.org/3.4/dd/d49/tutorial_py_contour_features.html

Reference Code for Hough transforms:

https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html.

Part-2

For detections using colour spaces:

https://realpython.com/python-opencv-color-spaces/

For skin color tone range:

https://stackoverflow.com/questions/8753833/exact-skin-color-hsv-range