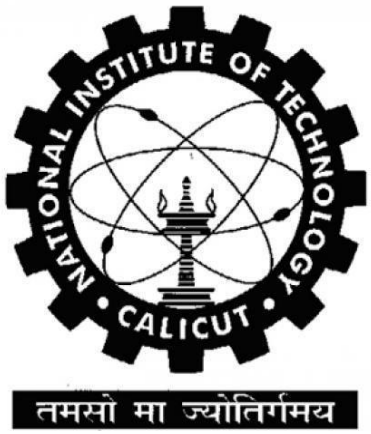
National Institute of Technology, Calicut



Cartooning Of An Image - Report 3

Image Processing

Under the guidance of :

Dr. M Prabu

Group - 6

Apurva Rathore (M190376CA)
Nidhi Redekar (M190366CA)
Naziya Khanam (M190393CA)
Shivangi Kesharwani (M190402CA)
Saif Ali Khan (M190379CA)
Kundan Singh Bhadoriya(M190661CA)
Saurabh Shahi (M190378CA)
Anjali Sharma (M190374CA)

CONTENT

1. CLASSIFICATION :

1.1 Median Blur : This is used to remove noise from the image, if in case the image contains any noise so it will restore that,so that in later processing a quality of image can be used.
In this, a median of each patch is found and replaces it with its mid Value. For example

234	67	245	234
123	45	23	245
34	0	234	24
255	235	145	253

In this image take a patch of 3*3 and find the median of it.
Values = 234, 67, 245, 123, 45 , 23, 34, 0, 234
Sort this value to find the median.
Values = 0, 23, 34, 45, 67, 123, 234, 234, 245

Median = 67
Do this for the entire image.

1.2 Adaptive Thresholding : It is a method in which a threshold is calculated and operation will be performed according to that. It is basically of 3 types:

1.2.1. Global Thresholding : In this a Threshold is fixed for entire image and operation will be performed according to that, for example :

Threshold = 150, so all the values Greater than 150
assigned 255 or all the values less than 150
assigned 0.

1.2.2. Mean Threshold : In this a threshold is calculated for each pixel, by taking mean of its neighbouring pixels. For example :

234	67	245	234
123	45	23	245
34	0	234	24
255	235	145	253

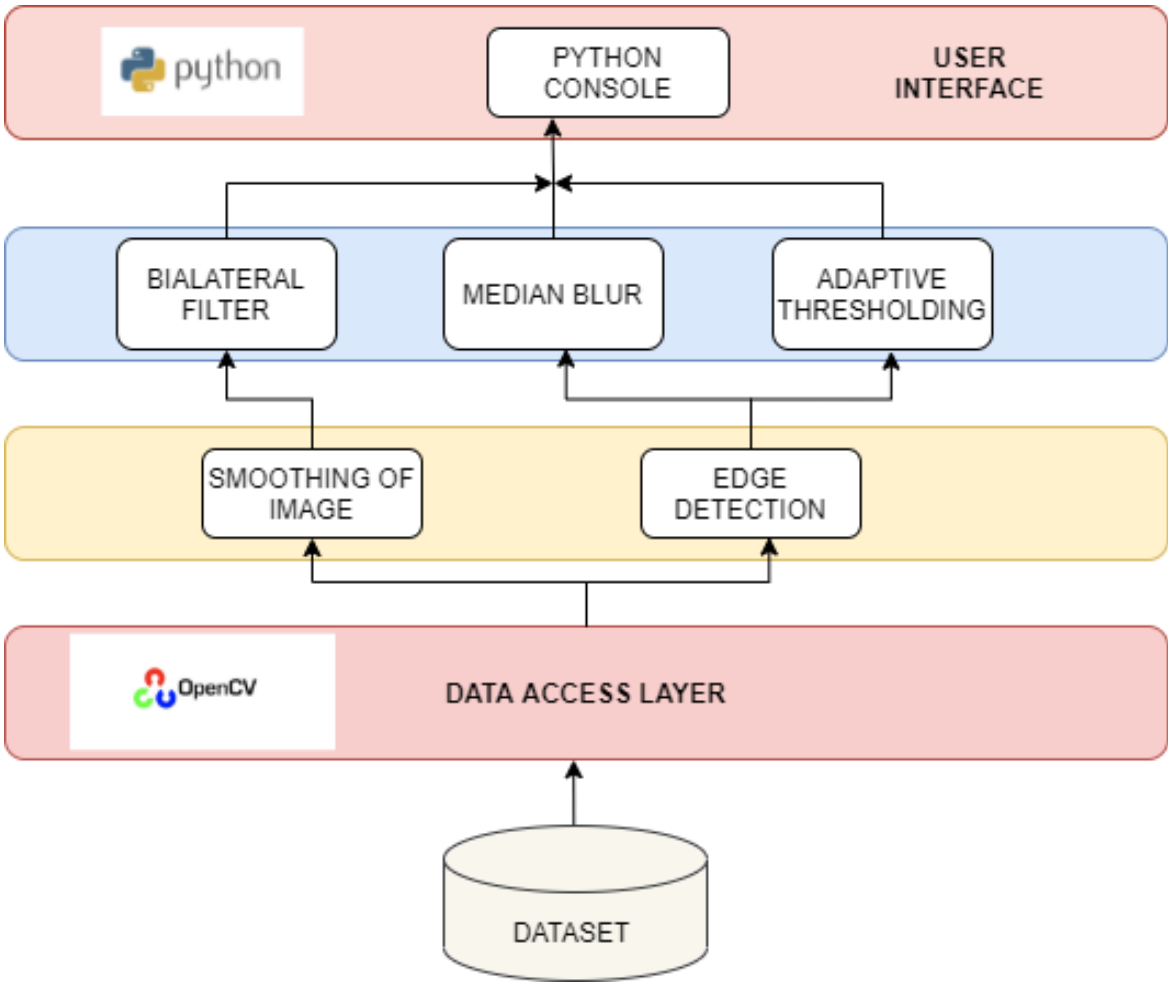
This is an image, take a patch of 3*3 and find the mean of it and that is the threshold value and do this for each patch.

For 1st patch = (234 + 67 + 245 + 123 + 45 + 23 + 34 + 234 + 0)/9
Threshold value = 111.667

1.2.3. Gaussian threshold : The threshold value is a gaussian-weighted sum of the neighbourhood values minus the constant C.

2. DESIGN AND IMPLEMENTATION :

2.1 Design :



2.2 Implementation :

In this project the implementation part is done with the help of the design that has shown everything clearly that we have used in the implementation part while coding. We have imported the entire dataset of the images and read the images using open cv2 which read the images in BGR format.

Basically Two steps is followed :

-> Edge detection
-> Image Restoration and Smoothing of the images

- Before EdgeDetection, We can perform Median Blur, it will basically reduce noise, As it is optional but noise can create false edges so its better to remove it. To detect edges, convert the RGB images to GrayScale, As in the grayscale each pixel is of the same intensity and in RGB each pixel in an image is a combination of three intensities, so processing of a single channel is faster than processing a three channel image.
  To detect image, we are using adaptive thresholding, opencv provide a method adaptiveThreshold(), that we are using.

- As the detection will take place in an image with the help of the adaptive threshold simultaneously For Image Smoothing we have used the Bilateral filter which will actually remove the high frequency content and reduce the color palette.

- Then after combining all the algorithms Adaptive Threshold, Median Blur and Bilateral Filter we will get an our required output in the python console i.e. Cartooning of an image that we have shown in the result part below.

3. DISCUSSION AND ACHIEVEMENTS :

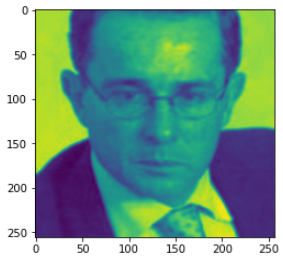
- Imported the entire dataset and read the images with cv2.imread(), it will read all images in BGR format. That we can see in the below image, it is bluish in color.

```
In [3]:
instances = []

# Load in the images
for filepath in os.listdir('Main'):
    instances.append(cv2.imread('Main/{0}'.format(filepath),0))

#print(type(instances[0]))
from matplotlib import pyplot as plt
plt.imshow(instances[399])
```

```
Out[3]: <matplotlib.image.AxesImage at 0x169ffad1f10>
```



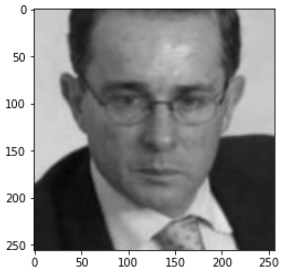
- We have to convert all the images to RGB format for later processing to remove that bluish effect and in the image below we can see the difference between RGB image and BGR image.

```
In [5]: images = []

        for i in instances:
            images.append(cv2.cvtColor(i, cv2.COLOR_BGR2RGB))

In [6]: plt.imshow(images[399])

Out[6]: <matplotlib.image.AxesImage at 0x169ffbd63a0>
```



- As we need to restore the images, if there is any noise, so by applying Bilateral Filter we can remove that noise and can make the image more smooth.

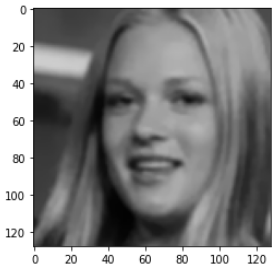
In the below image we can see the image is better than before.

```
In [8]: imagesBil = []

        for i in img_small:
            num_iter = 5
            for _ in range(num_iter):
                img = cv2.bilateralFilter(i, d=9, sigmaColor=9, sigmaSpace=7)
            imagesBil.append(img)

        plt.imshow(imagesBil[0])

Out[8]: <matplotlib.image.AxesImage at 0x169ffc98cd0>
```



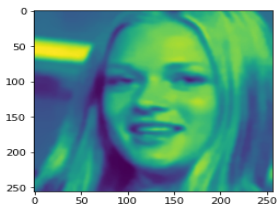
- Here, we are converting a RGB image to GrayScale image, As processing of grayscale images is faster than RGB images. In the image below, we can see how the grayscale image is different from RGB.

Edge Detection

```
In [10]: img_gray = []

         for i in img_rgb:
             img_gray.append(cv2.cvtColor(i, cv2.COLOR_RGB2GRAY))
```

```
In [11]: plt.imshow(img_gray[0])
Out[11]: <matplotlib.image.AxesImage at 0x169ffd56f40>
```

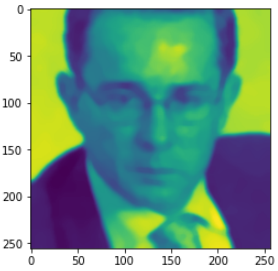


- Here, we can see the result of Median blur, it will reduce noise from the image,As it is optional but it is a good practice to use it because a noise in an image can give false results also.

```
In [12]: img_blur = []

         for i in img_gray:
             img_blur.append(cv2.medianBlur(i, 7))

         plt.imshow(img_blur[399])
Out[12]: <matplotlib.image.AxesImage at 0x169ffdb8910>
```

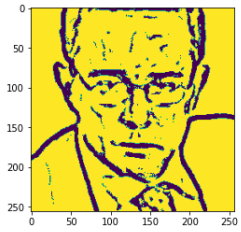


- Here, we are applying Adaptive thresholding and we can see the results, it detected the edges.

```
In [13]: img_edge = []

         for i in img_blur:
             img_edge.append(cv2.adaptiveThreshold(i, 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY, 7, 2))

         plt.imshow(img_edge[399])
Out[13]: <matplotlib.image.AxesImage at 0x169ffe21160>
```



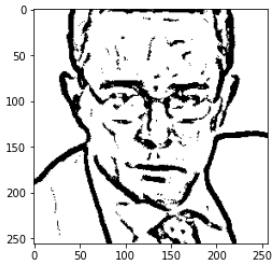
- As our processing is completed we can now convert again the grayscale image to RGB.

```
In [14]: img_togray = []

         for i in img_edge:
             img_togray.append(cv2.cvtColor(i, cv2.COLOR_GRAY2RGB))

         plt.imshow(img_togray[399])

Out[14]: <matplotlib.image.AxesImage at 0x169ffe79760>
```

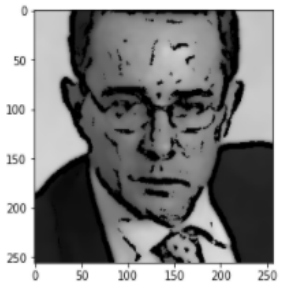


4. RESULTS :-

Here is the resulting effect that we have got. We can see the difference between original image and resulting image.That we have got after applying bitwise operator to both the images, Bilateral filter and edge detection.

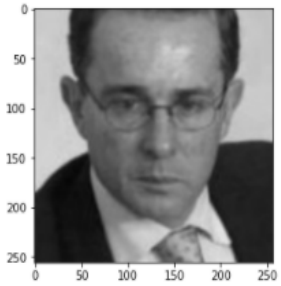
```
In [22]: plt.imshow(result[399])

Out[22]: <matplotlib.image.AxesImage at 0x2ca8df63d60>
```




```
In [24]: plt.imshow(images[399])

Out[24]: <matplotlib.image.AxesImage at 0x2caa1ef0400>
```



5. REFERENCES :-

1. https://stackoverflow.com/questions/48435977/reduce-number-of-pixels-to-obtain-low-resolution-image
2. https://www.codegrepper.com/code-examples/whatever/convert+bgr+to+rgb+opencv+python
3. https://datacarpentry.org/image-processing/08-edge-detection/
4. http://opencvexamples.blogspot.com/2013/10/applying-bilateral-filter.html
5. https://www.easytechjunkie.com/what-is-image-scaling.htm
6. https://www.tutorialspoint.com/opencv/opencv_median_blur.htm
7. https://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm
8. https://homepages.inf.ed.ac.uk/rbf/HIPR2/adpthrsh.htm#:~:text=Adaptive%20thresholding%20typically%20takes%20a,threshold%20has%20to%20be%20calculated.https://homepages.inf.ed.ac.uk/rbf/HIPR2/adpthrsh.htm#:~:text=Adaptive%20thresholding%20typically%20takes%20a,threshold%20has%20to%20be%20calculated.
9. https://www.pyimagesearch.com/2021/01/19/opencv-bitwise-and-or-xor-and-not/
10. https://stackoverflow.com/questions/44333605/what-does-bitwise-and-operator-exactly-do-in-opencv
11. https://serc.carleton.edu/earth_analysis/image_analysis/advanced/day_2_part_2.html
12. https://techterms.com/definition/image_scaling
13. https://towardsdatascience.com/loading-custom-image-dataset-for-deep-learning-models-part-1-d64fa7aaeca6
14. https://www.w3schools.com/python/matplotlib_pyplot.asp
15. https://stackoverflow.com/questions/49518537/appending-images-using-python-and-imagemagick
16. https://www.codegrepper.com/code-examples/python/load+a+list+of+images+python
17. https://note.nkmk.me/en/python-opencv-bgr-rgb-cvtcolor/
18. https://pythonmatplotlibtips.blogspot.com/2017/12/plot-on-image-matplotlib-pyplot.html
19. https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_filtering/py_filtering.html
20. https://www.quora.com/What-is-Gaussian-filtering-in-image-processing
21. https://www.tutorialspoint.com/opencv/opencv_adaptive_threshold.htm
22. https://processing.org/tutorials/pixels/
23. https://buzztech.in/image-acquisition-in-digital-image-processing/
24. https://users.cs.cf.ac.uk/Dave.Marshall/Vision_lecture/node4.html
25. https://data-flair.training/blogs/python-bitwise-operators/