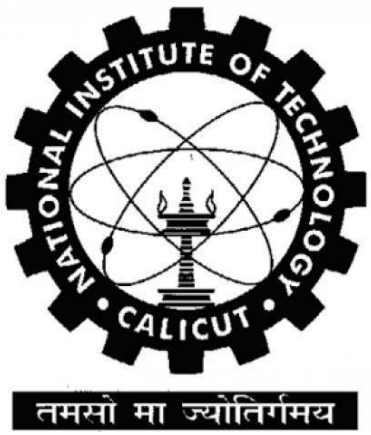
National Institute of Technology, Calicut



Cartooning Of An Image - Report 4

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)

1. Introduction :

Cartoons are often used in a variety of contexts. In this project our team had worked on the idea of digital cartooning of an image. As we know cartoons are artistically made it requires elegant and fine human artistic skills.While portraying there are a lot of cartoons for any animated film, the artist must spend time properly defining the cartoon sketch in order to achieve a successful result in less time, one needs a cartooning tool. As cartooning a digital image appears to be a fascinating and enjoyable project to work on. We just need a bilateral filter and an edge detection mechanism to create cartoon images from a digital image. These bilateral filters will help us reduce the image's colour or shading palette, which is a crucial move for the cartoon look, and edge detection will be used to achieve the perfect result.

2. Origin of Work:

Émile Cohl a french cartoonist and animator is referred as the *"Father of Animated Cartoon."*

The term cartoon was first used in the Middle Ages to describe a prerequisite sketch for a work of art like a painting or picasso. It's a simple drawing that exaggerates the features of its subjects in a satirical way, usually in a newspaper or magazine. Previously, cartoons were only intended to be entertaining. Cartoons have been used as a teaching tool for a long time.

Nevertheless, time is the most important consideration in today's world. Since drawing the cartoon photos by hand took a long time. As a result, we now have access to a wide range of technologies to help us do our jobs more efficiently. As a result, we must employ those technologies in order to create a cartoon-like image.

3. Literature review :

- **Cartooning an Image Using Opencv and Python - By Apurva Rathore [M190376CA]**

  According to this research paper, Vaishali Sudarshan and Amritesh Singh's algorithm for converting an image to its cartoon form can be divided into two parts: 1. Smoothing, quantizing, and grayscale conversion of the RGB image. 2. To track, blur, and keep the real RGB colour image's edges.

  The tasks Median Filter, Edge Detection, Morphological Operations, and Edge Filtering are used to classify the image's edges. The methods Bilateral Filtering and Quantize Colors are used to smooth out colours.

  The final step is to recombine the two images that were worked on. When both the colour and edge image processing are finished, this will be achieved by overlaying the edges onto the colour image.

- **Image-to-Image Translation - By Saif Ali Khan [M190379CA]**

  Visualize Converting images from one domain to another is no longer a problem thanks to translation. It is primarily used to enhance image clarity, stylize paint pictures, cartoon images and drawings, and colourize grayscale pictures and drawings.

  Bi-directional cross-domain translation models were also used for this implementation of Image-to-Image Translation. It changes the images that aren't matched. For the purposes of image cartooning, they use an unpaired image-to-image conversion process.

  We decompose images into multiple representations that enable the network to learn different features with distinct objectives, rendering the learning process controllable and tunable, in comparison to previous black-box models that guide network training with failure words.

- **Ruben Winastwan - By Naziya Khanam [M190393CA]**

  To convert an image into a cartoon, we simply need to use two steps: edge detection and region smoothing, according to this article. Noise is reduced by smoothing regions. Depending on the filters we use, we can get a variety of different cartoon effects.
  He's using pencil sketch, Detail enhancement, bilateral filter, and pencil edges to create this image.

  He converts the picture to grayscale and blurs it with GaussianBlur() in the pencil sketch. The more blur effects are used, the more each pixel's value shifts in relation to its origin, giving us a sharper pencil drawing.He smoothed the picture and reduced the noise with a Bilateral filter.We can get a nice cartoon effect by using bitwise & of pencil sketches and bilateral filtered images.

- **Automatic Cartoon Colorization -By Shivangi Kesharwani [M190402CA]**

  Domonkos Varga and his team created this paper about automatic cartoon colorization. They've shown that proper colour filling in cartoon photos on various sizes is possible. There were several research avenues to pursue.

  To solve the problem of learning-based recoloring, they used additional details such as color-sketches of a vibrant reference image. Another field of research is the fine-perfect VGG-16-based algorithms on the cartoon domain, which they have used**.**

  The method can also be extended to decolorize highly textured natural images as cartoon-like samples, in which the cartoon section and the textures can be removed, and the artificially de-textured image can then be filled with recoloring to produce a cartoon-like coloured image.

- **Cartoon Photo Effect Application(TOONIFY) - By Nidhi Redekar [M190366CA]**

  Mr. Kevin Dade used the graphics engine's cel-shading effects in this research project. Cel shading is a technique for making 2D graphics appear 3D. OpenCV for Android, Bilateral Filters, and Median Filters were also used by Mr. Kevin**.** He split his algorithm for creating a cartoon image into two sections, detecting image edges and quantization image colours, and then combining the two images. TOONIFY was the name given by Mr. Kevin to this tool.

- **International Research Journal Of Engineering and Technology (IRJET)- By Anjali Sharma [M190374CA]**

  In this research paper, the author is transforming realistic images and videos into Cartoon images and videos by Using Generative Adversarial Network(GAN) and in project specific terms also known as “Cartoon GAN” technique which is basically based on converting an image into a very sharp and clear cartoon/animated form. This technique mainly uses 2 loss functions which are Adversarial loss function and Content Loss function for getting the sharp and fine image. We can also convert the video clips into the animation clips(cartoonized form) with the help of CV2 which is known as Computer vision.

- **A Personalized Image-based Cartoon System - By Kundan Singh Bhadoriya [M190661CA]**

  This research work is done by Mr. Hong Chen and the team and they named it **PicToon**. PicToon is a cartoon system that can generate a personalized cartoon image from an input picture. This tool consists of 3 major components: an image-based Cartoon Generator, an interactive Cartoon Editor and a speech-driven Cartoon Animator. In their cartoon generation approach ,an effective sampling scheme along with a flexible facial template is implemented to automatically extract a facial sketch from an input image. The aim of Mr. Hong, while this research was to create a user-friendly platform for making cartoons and that too with a vision of an artist.

- **CartoonGAN -By Saurabh Shahi [M190378CA]**

  In this research work,Mr. Yang Chen and his team refers to GAN which stands for Generative Adversarial Network. Two CNNs (Convolutional Neural Networks) are required to complete a GAN structure. The first is the generator G, which is responsible to generate output that deceives the discriminator. ,And the other one is the discriminator D,that determines whether the image is from a real target manifold or a synthetic one. Combination of Both generator and discriminator networks gives a cartoon picture as output.
  The generator network G used in this paper is to map input images to the cartoon manifold. After the model has been conditioned, it is stylized into a cartoon. G starts with a flat convolution point, then uses two down-convolution blocks to compress and encode the images spatially. Discriminator is used to check whether the input image is a cartoon image or not . this can't be taken for granted for producing the correct output image.

4. Objectives:

This project's main goal is to transform photos or snapshots into cartoons. As a result, it would be beneficial in today's cartooning situation.

- As we all know, anime movies and cartoons are popular in magazines for social entertainment and visual irony, as well as for political commentary and editorial opinion.

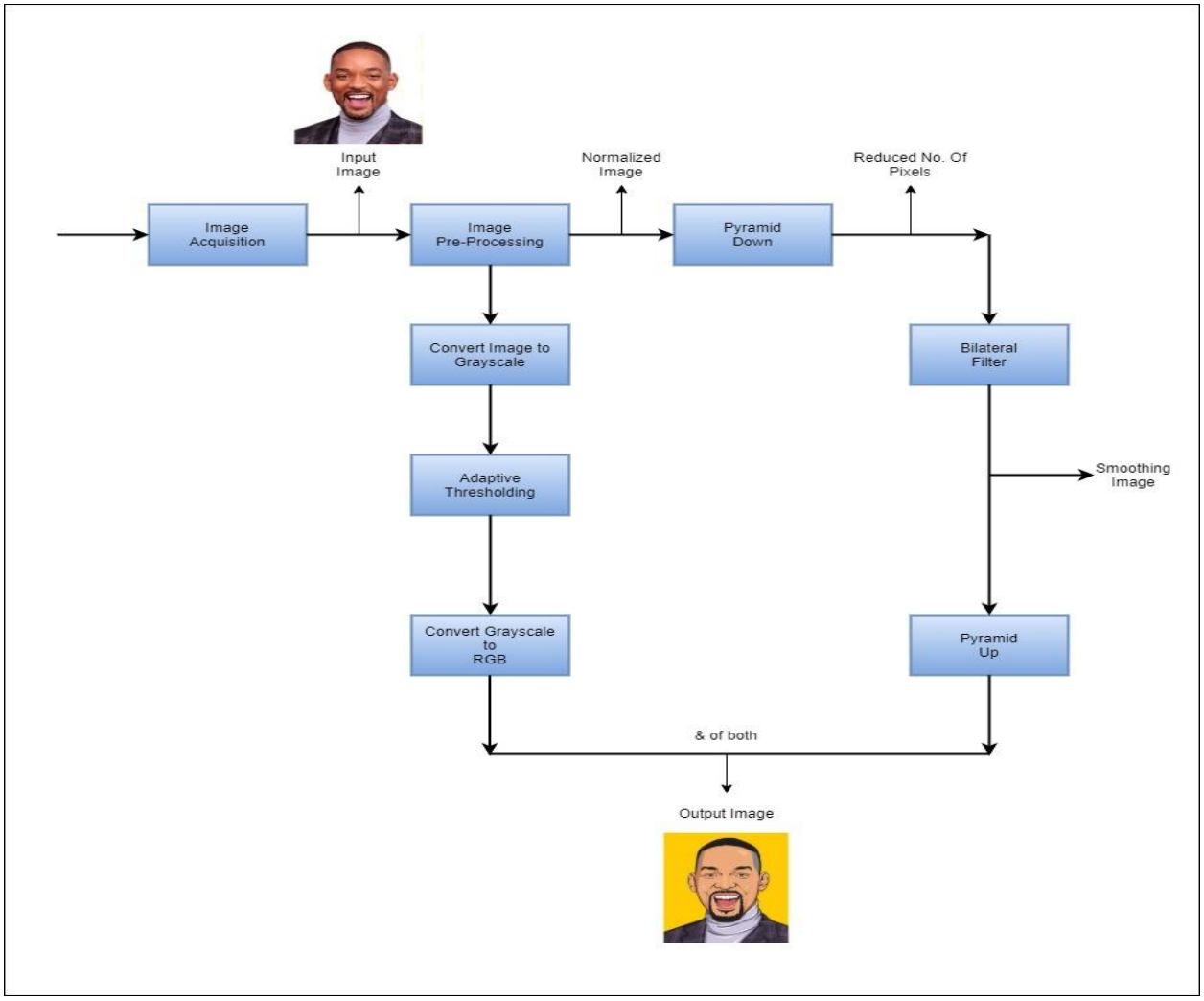
- This is a quick rundown of cartoons, comic strips, and short films such as anime animation.

- The majority of books with cartoons are magazine-format "comic books," or editions of newspaper cartoons.

- We will teach our children about our world, today's condition, and what we are facing in every area with the aid of this project. They will be able to comprehend the current situation more readily as they are more familiar with cartoons and animes than with posts.

- Political cartoons are drawings (usually of a cartoon character) that are used to convey editorial commentary on government, politicians, and current events. Such cartoons have a place in a culture that values press freedom, which is also important in politics.

5. Block diagram:



To get the cartoonized image generally there are two main operations that we have to apply to get the output. One is to reduce the colour palette and the other is to identify the edges. Basically reducing the colour palette is done by the Bilateral filter, it refers to reducing the noise from the image and making the image smooth. To Identify edges we have to apply Adaptive thresholding. It's a process of converting an image into one particular pixel either white or black given a particular threshold. Combination of both the operations result in the image into cartoon effect.

6. Data collection:

In our project 'Cartooning of an image', we are converting a person's image to its cartoon aspect. We have taken the image dataset from https://www.kaggle.com/prasunroy/natural-images. This dataset originally includes 6899 images from which we have taken approx 400 images with size 16kb to 20kb and a dimension of 256 x 256 (width = 256 pixels and height = 256 pixels). The resolution of these images is 96 dpi and the bit depth is 24.

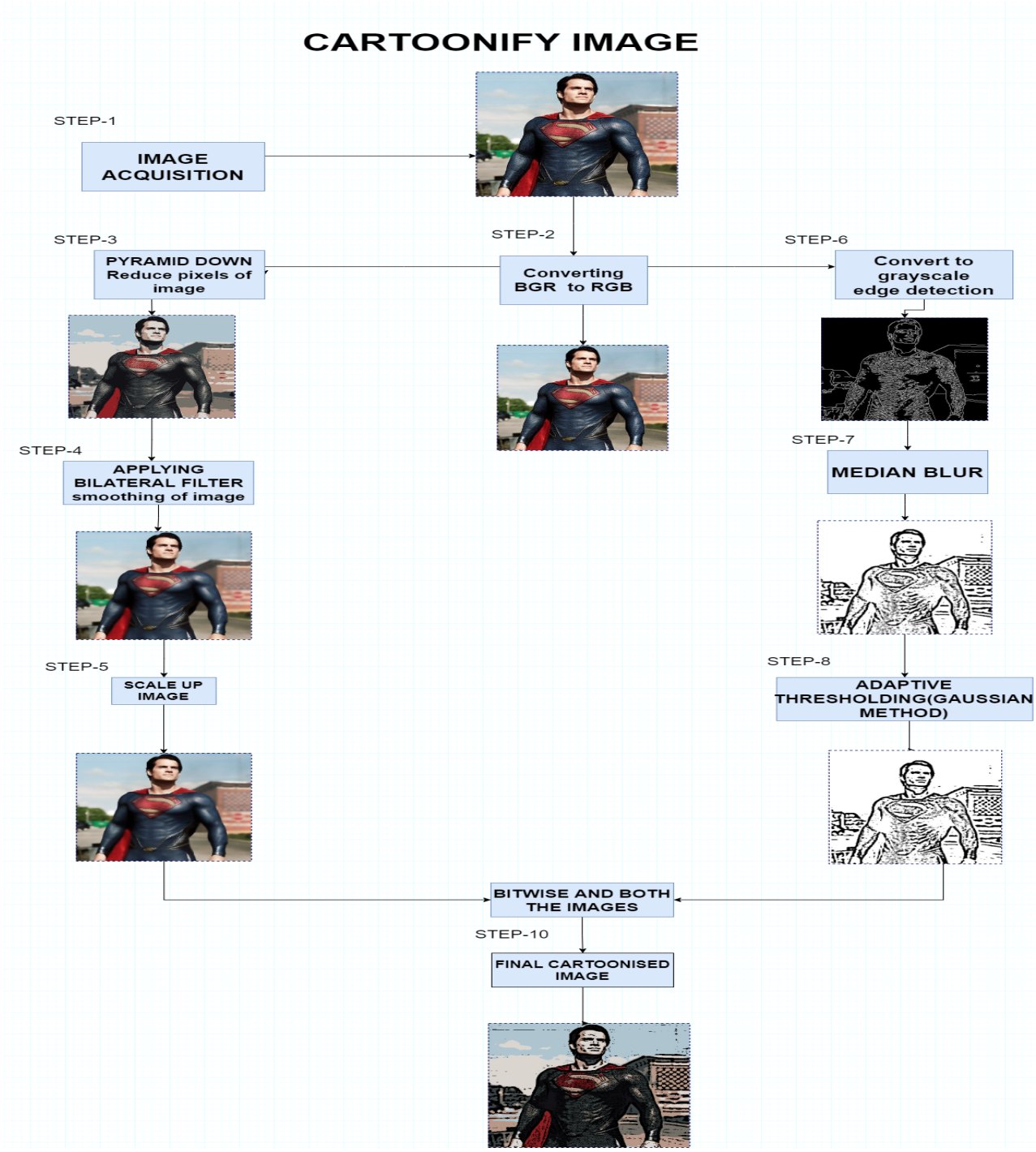
7. Working Procedure:

By using OpenCV in Python we are going to process our images and first, we will import all the essential libraries. Then we can read the images and store them into an array by providing the path of the dataset in our system. We will also print the images to verify that the images are correctly stored in a variable.

Now, as the image dataset is imported into the program, we can perform various operations for cartoonizing those images by reducing the color palette and edge identification through Bilateral Filter and Adaptive Thresholding and get the necessary output.

8. Methodology and Classification:

Diagram: A Block diagram of the Methodology that we have used is represented here in steps.



Explanation:

Image Acquisition is the process of retrieving the image from the source. Images can be of any format which means in jpeg, jpg, or png format. OpenCV image reader function (imread()) reads the image as a NumPy array ndarray of row (height) x column (width) x color (3) and here the order of color is in BGR format, so image should be converted in the RGB format. After getting the image in a correct format,two operations are performed on this Image.

1. Image restoration is the operation on corrupt images such as noisy, camera mis-focus, blur images etc. and estimates the clean and original image.For Image Smoothing we have used the Bilateral filter which actually removes the high frequency content(noise,edges) and reduces the color palatte.

2. Detect the Edges Adaptive thresholding is used, and before applying this Median blur is performed so that it will help in later processing.Adaptive thresholding takes a grayscale or color image as input so RGB image has to be converted into Grayscale.For each pixel in the image threshold is calculated. OpenCV provides **adaptiveThreshold()**, it uses two methods, one is to find out the mean of neighboring pixels and other is to find out the total weight of neighboring pixels,where weight is the Gaussian window. At last we can combine the results of the first and second operation and we get the cartoon effect.

Classification : IP Algorithms

1. Median Blur
2. Adaptive Thresholding

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.

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. 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.

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

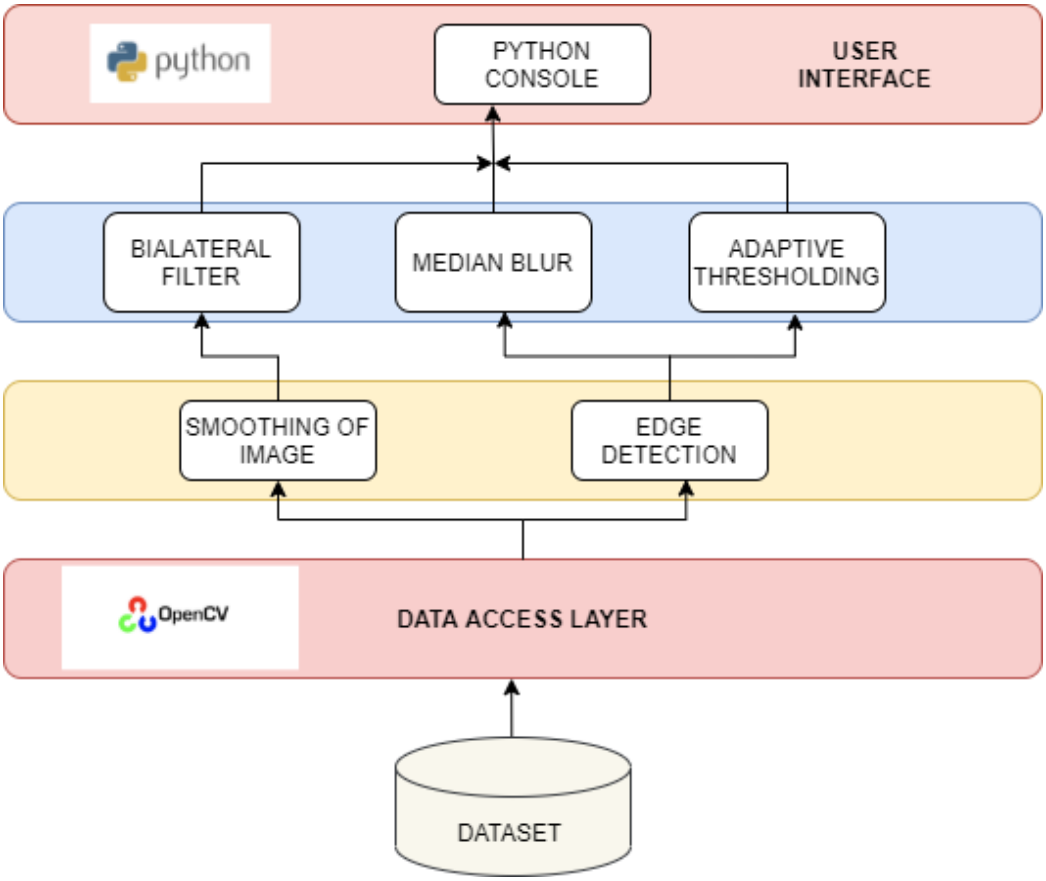
234	67	245	234
123	45	23	245
34	0	234	24
255	235	145	253

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

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

9. Design and implementation:

Design:-



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.

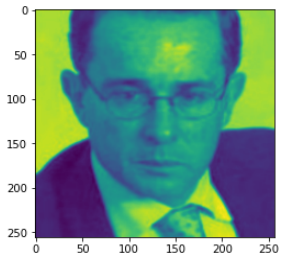
- Then after combining all the algorithms Adaptive Threshold, Median Blur and Bilateral Filter we will get an our required output in the python console i.e. Cartooning of an image that we have shown in the result part below.

10. Discussion :

- 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.

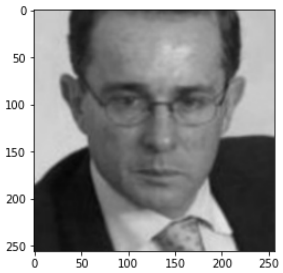
```
In [3]:
instances = []

# 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>
```



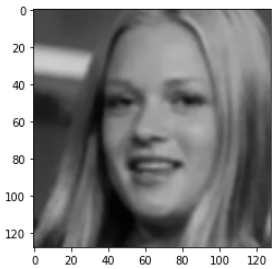
- We have to convert all the images to RGB format for later processing to remove that bluish effect and in the image below we can see the difference between RGB image and BGR image.

```
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>
```



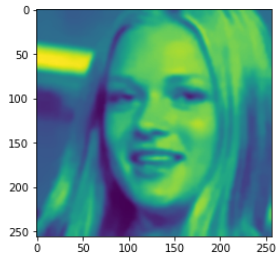
- Here, we are converting a RGB image to GrayScale image, As processing of grayscale images is faster than RGB images. In the image below, we can see how the grayscale image is different from RGB.

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])
```

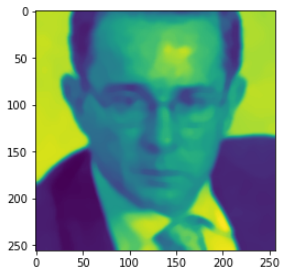
Out[11]:
```
<matplotlib.image.AxesImage at 0x169ffd56f40>
```



- 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])
```

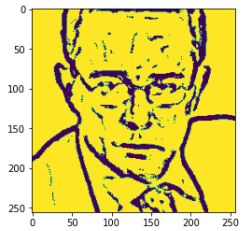
Out[12]:
```
<matplotlib.image.AxesImage at 0x169ffdb8910>
```



- 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])
```

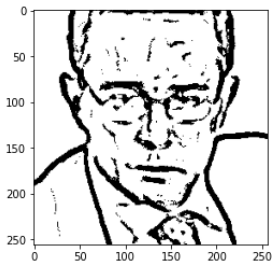
```
Out[13]: <matplotlib.image.AxesImage at 0x169ffe21160>
```



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

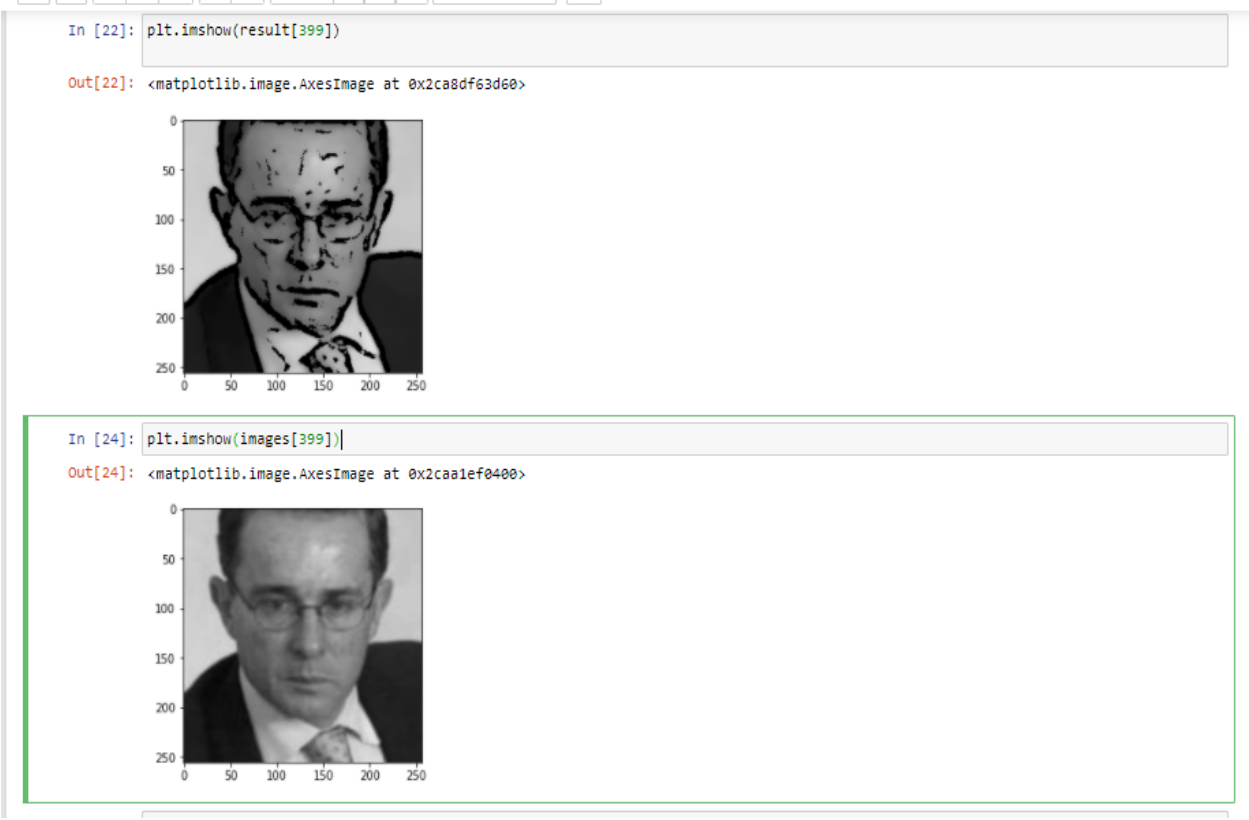
```
In [14]: img_togray = []

         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>
```



11. 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.



12.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/