Report On

# “ Cartoonify an Image with Open CV in Python”

Machine Learning with Python Mini Project

Submitted by:

**Utkarsh Kumar**

USN

18030141CSE026

Under the Guidance of (Faculty Guide)

**Prof. RADHA R**

in Semester VI of the Bachelor of Technology

Batch - 2018-2022



**ABSTRACT**

Computer Vision as you know (or even if you don’t) is a very powerful tool with immense possibilities. So, when I set up to prepare a comic of one of my friend’s college life, I soon realized that I needed something that would reduce my efforts of actually painting it but will retain the quality and I came up with the following solution.

 Currently there are lots of professional cartoonizer applications available in the market but most of them are not freeware. In order to get the basic cartoon effect, we just need the bilateral filter and some edge dectection mechanism. The bilateral filter will reduce the color palette, which is essential for the cartoon look and edge detection is to produce bold silhouettes.

I am going to use openCV python library to convert an RGB color image to a cartoon image.

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| **Chapter1** | Introduction about the topic. |
| **Chapter2** | Project Profile. |
| **Objective** | The main objective. |
| **Prerequisites** | Tools and prerequired steps. |
| **Algorithm** | Algorithm of source code. |
| **Code** | Source code. |
| **Screenshots** | Screenshots of code and output. |
| **Conclusion** | Conclusion of the project. |

**CHAPTER 1**

**INTRODUCTION**

Do you miss your childhood? Yes, everyone does! So today let’s head towards giving our pictures some cartoonic effects. This article is all about building a photo cartoonifyer using Python and OpenCV

### **What is OpenCV?**

Python is the pool of libraries. It has numerous libraries for real-world applications. One such library is OpenCV. OpenCV is a cross-platform library used for Computer Vision. It includes applications like video and image capturing and processing. It is majorly used in image transformation, object detection, face recognition, and many other stunning applications.

At the end of this project, we aim to transform images into its cartoon. Yes, we will CARTOONIFY the images. Thus, we will build a application that will transform an image into its cartoon using OpenCV.

To produce accurate carton effects, as the first step, we need to understand the difference between a common digital image and a cartoon image. In the following example, you can see how both images look like.



At the first glance we can clearly see two major differences.

1. The first difference is that the colors in the cartoon image are more homogeneous as compared to the normal image.
2. The second difference is noticeable within the edges that are much sharper and more pronounced in the cartoon.

Now, when we have clarified two main differences it is straightforward what our job is. We need to detect and emphasize the edges and apply a filter to reduce the color palette of the input image. When we achieve that goal, we would obtain a pretty cool result.

**CHAPTER 2**

**PROJECT PROFILE**

**OBJECTIVES OF THE STUDY**

The main objective of this project iswe aim to transform images into its cartoon using python and opencv.

**PREREQUISITES:**

**1. Tools to be installed on computer:**

-Python

-We will import the following Module:

* CV2: Imported to use OpenCV for image processing
* Easygui: Imported to open a file box. It allows us to select any file from our system.
* Numpy: Images are stored and processed as numbers. These are taken as arrays. We use NumPy to deal with arrays.
* Imageio: Used to read the file which is chosen by file box using a path.
* Matplotlib: This library is used for visualization and plotting. Thus, it is imported to form the plot of images.
* OS: For OS interaction. Here, to read the path and save images to that path.

**ALGORITHM:**

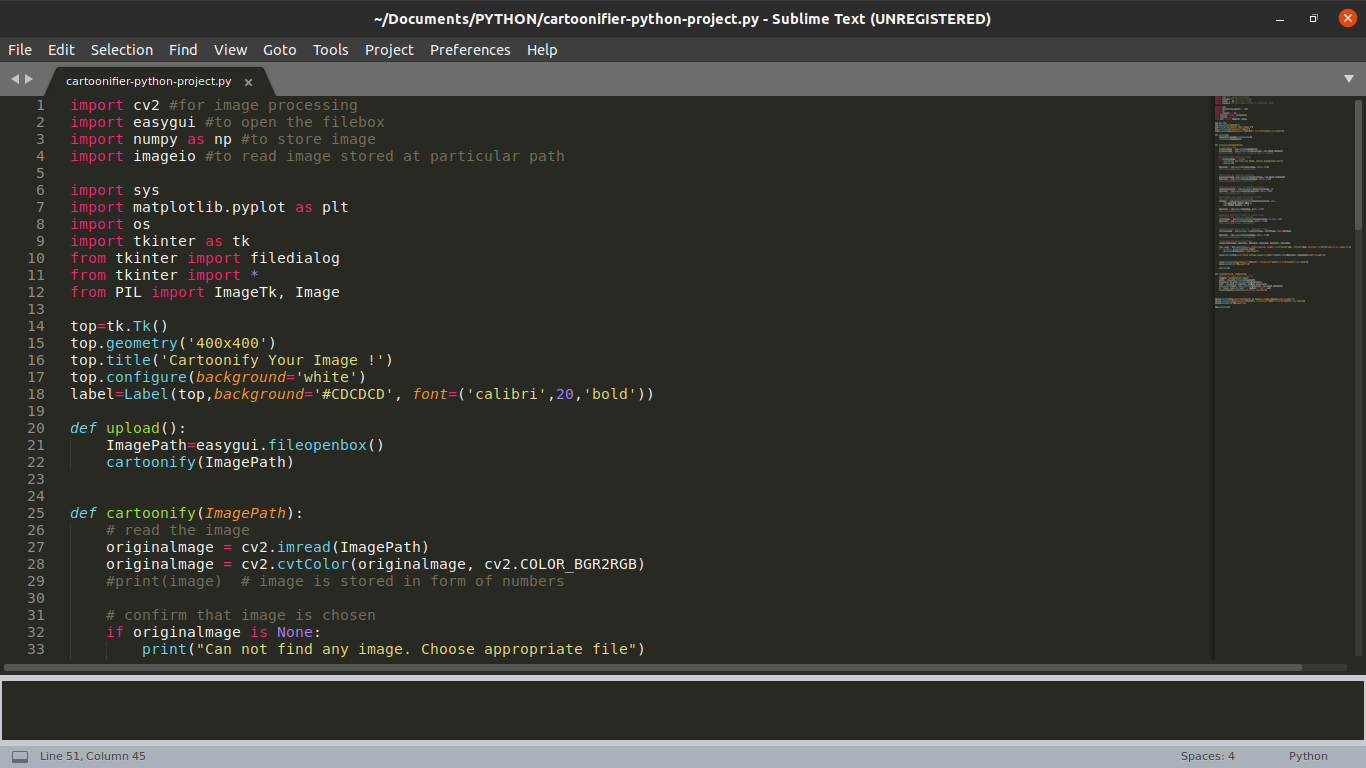
1. Firstly apply the bilateral filter to reduce the color palette of the image.
2. Then convert the actual image to grayscale.
3. Now apply the median blur to reduce image noise in the grayscale image.
4. Create an edge mask from the grayscale image using adaptive thresholding.
5. Finally combine the color image produced from step 1 with edge mask produced from step 4

**CODE:**

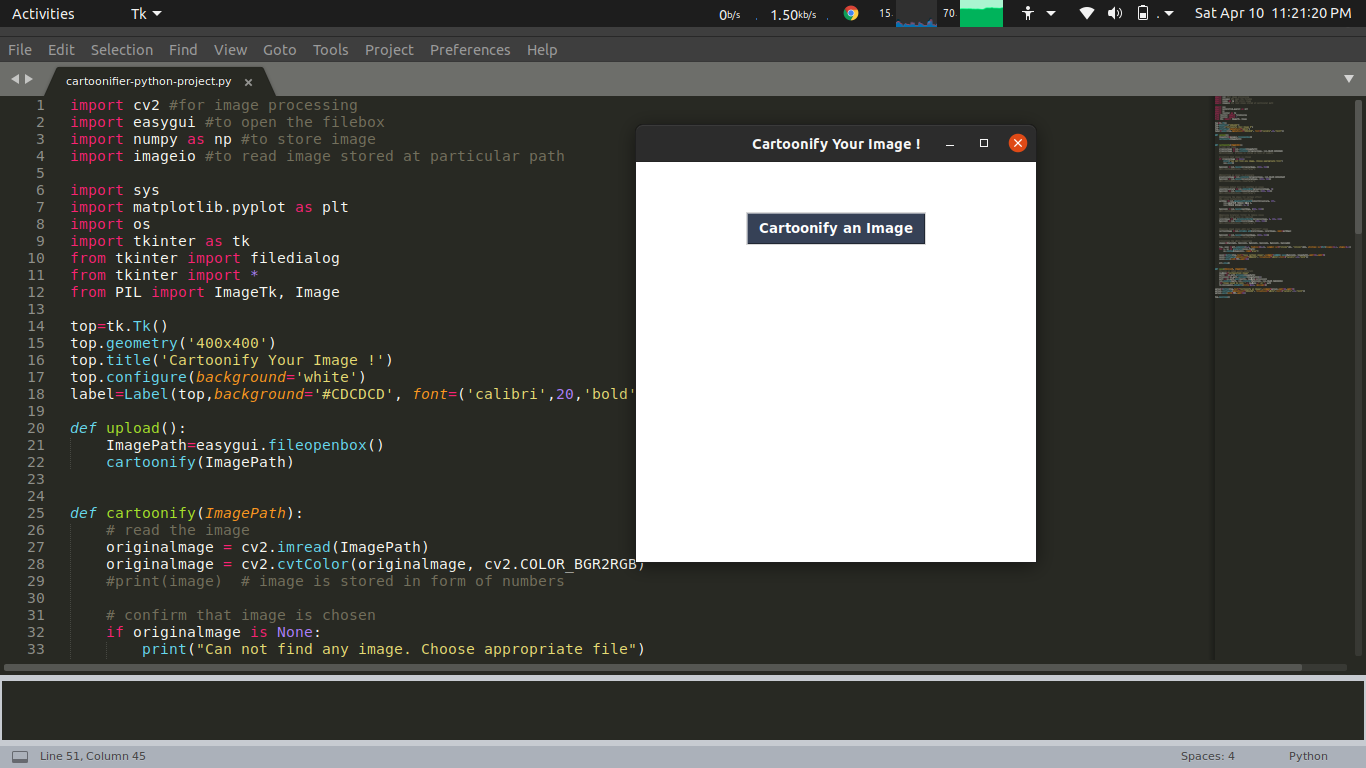
**import** cv2 *#for image processing***import** easygui *#to open the filebox***import** numpy **as** np *#to store image***import** imageio *#to read image stored at particular path***import** sys  
**import** matplotlib.pyplot **as** plt  
**import** os  
**import** tkinter **as** tk  
**from** tkinter **import** filedialog  
**from** tkinter **import** \*  
**from** PIL **import** ImageTk, Image  
  
top=tk.Tk()  
top.geometry(**'400x400'**)  
top.title(**'Cartoonify Your Image !'**)  
top.configure(background=**'white'**)  
label=Label(top,background=**'#CDCDCD'**, font=(**'calibri'**,20,**'bold'**))  
  
**def** upload():  
 ImagePath=easygui.fileopenbox()  
 cartoonify(ImagePath)  
  
  
**def** cartoonify(ImagePath):  
 *# read the image* originalmage = cv2.imread(ImagePath)  
 originalmage = cv2.cvtColor(originalmage, cv2.COLOR\_BGR2RGB)  
 *#print(image) # image is stored in form of numbers  
  
 # confirm that image is chosen* **if** originalmage **is None**:  
 print(**"Can not find any image. Choose appropriate file"**)  
 sys.exit()  
  
 ReSized1 = cv2.resize(originalmage, (960, 540))  
 *#plt.imshow(ReSized1, cmap='gray')  
  
  
 #converting an image to grayscale* grayScaleImage= cv2.cvtColor(originalmage, cv2.COLOR\_BGR2GRAY)  
 ReSized2 = cv2.resize(grayScaleImage, (960, 540))  
 *#plt.imshow(ReSized2, cmap='gray')  
  
  
 #applying median blur to smoothen an image* smoothGrayScale = cv2.medianBlur(grayScaleImage, 5)  
 ReSized3 = cv2.resize(smoothGrayScale, (960, 540))  
 *#plt.imshow(ReSized3, cmap='gray')  
  
 #retrieving the edges for cartoon effect  
 #by using thresholding technique* getEdge = cv2.adaptiveThreshold(smoothGrayScale, 255,   
 cv2.ADAPTIVE\_THRESH\_MEAN\_C,   
 cv2.THRESH\_BINARY, 9, 9)  
  
 ReSized4 = cv2.resize(getEdge, (960, 540))  
 *#plt.imshow(ReSized4, cmap='gray')  
  
 #applying bilateral filter to remove noise   
 #and keep edge sharp as required* colorImage = cv2.bilateralFilter(originalmage, 9, 300, 300)  
 ReSized5 = cv2.resize(colorImage, (960, 540))  
 *#plt.imshow(ReSized5, cmap='gray')  
  
  
 #masking edged image with our "BEAUTIFY" image* cartoonImage = cv2.bitwise\_and(colorImage, colorImage, mask=getEdge)  
  
 ReSized6 = cv2.resize(cartoonImage, (960, 540))  
 *#plt.imshow(ReSized6, cmap='gray')  
  
 # Plotting the whole transition* images=[ReSized1, ReSized2, ReSized3, ReSized4, ReSized5, ReSized6]  
  
 fig, axes = plt.subplots(3,2, figsize=(8,8), subplot\_kw={**'xticks'**:[], **'yticks'**:[]}, gridspec\_kw=dict(hspace=0.1, wspace=0.1))  
 **for** i, ax **in** enumerate(axes.flat):  
 ax.imshow(images[i], cmap=**'gray'**)  
  
 save1=Button(top,text=**"Save cartoon image"**,command=**lambda**: save(ReSized6, ImagePath),padx=30,pady=5)  
 save1.configure(background=**'#364156'**, foreground=**'white'**,font=(**'calibri'**,10,**'bold'**))  
 save1.pack(side=TOP,pady=50)  
   
 plt.show()  
   
   
**def** save(ReSized6, ImagePath):  
 *#saving an image using imwrite()* newName=**"cartoonified\_Image"** path1 = os.path.dirname(ImagePath)  
 extension=os.path.splitext(ImagePath)[1]  
 path = os.path.join(path1, newName+extension)  
 cv2.imwrite(path, cv2.cvtColor(ReSized6, cv2.COLOR\_RGB2BGR))  
 I= **"Image saved by name "** + newName +**" at "**+ path  
 tk.messagebox.showinfo(title=**None**, message=I)  
  
upload=Button(top,text=**"Cartoonify an Image"**,command=upload,padx=10,pady=5)  
upload.configure(background=**'#364156'**, foreground=**'white'**,font=(**'calibri'**,10,**'bold'**))  
upload.pack(side=TOP,pady=50)  
  
top.mainloop()

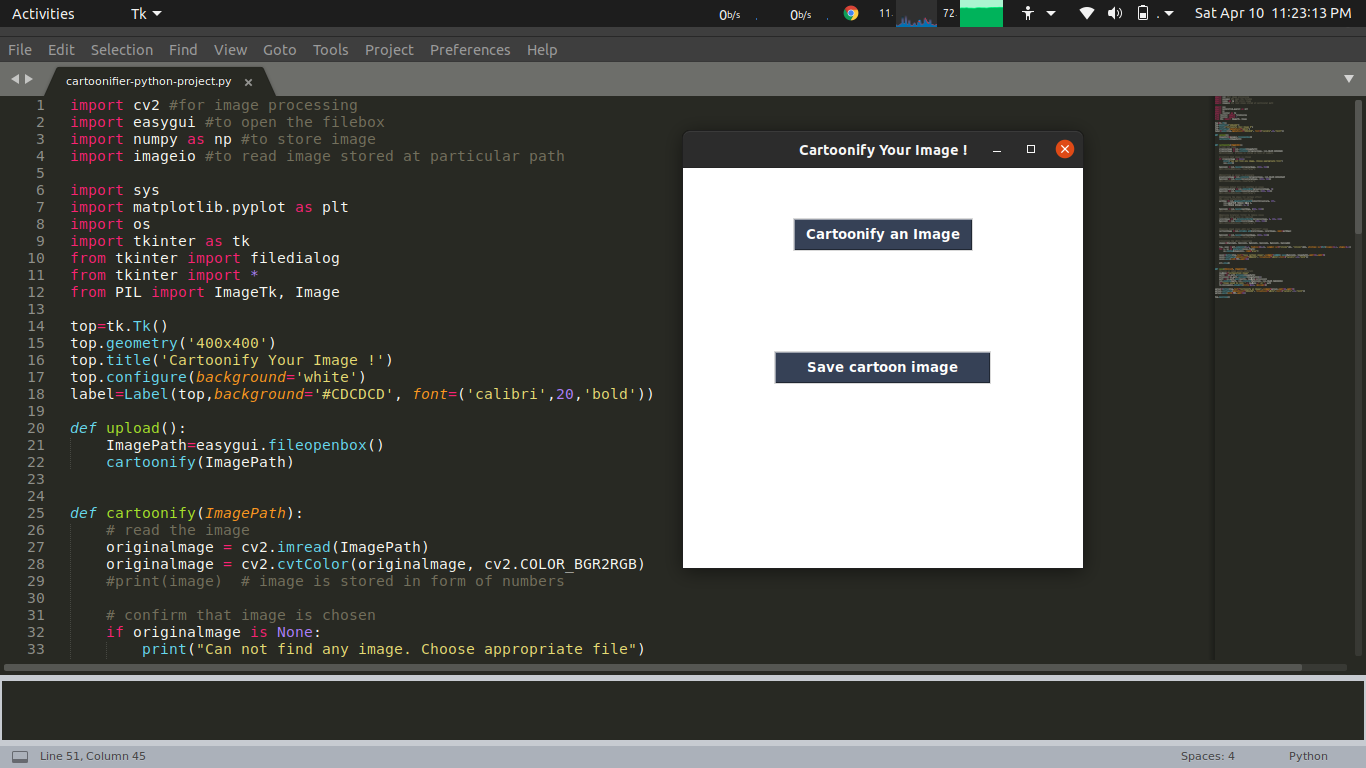
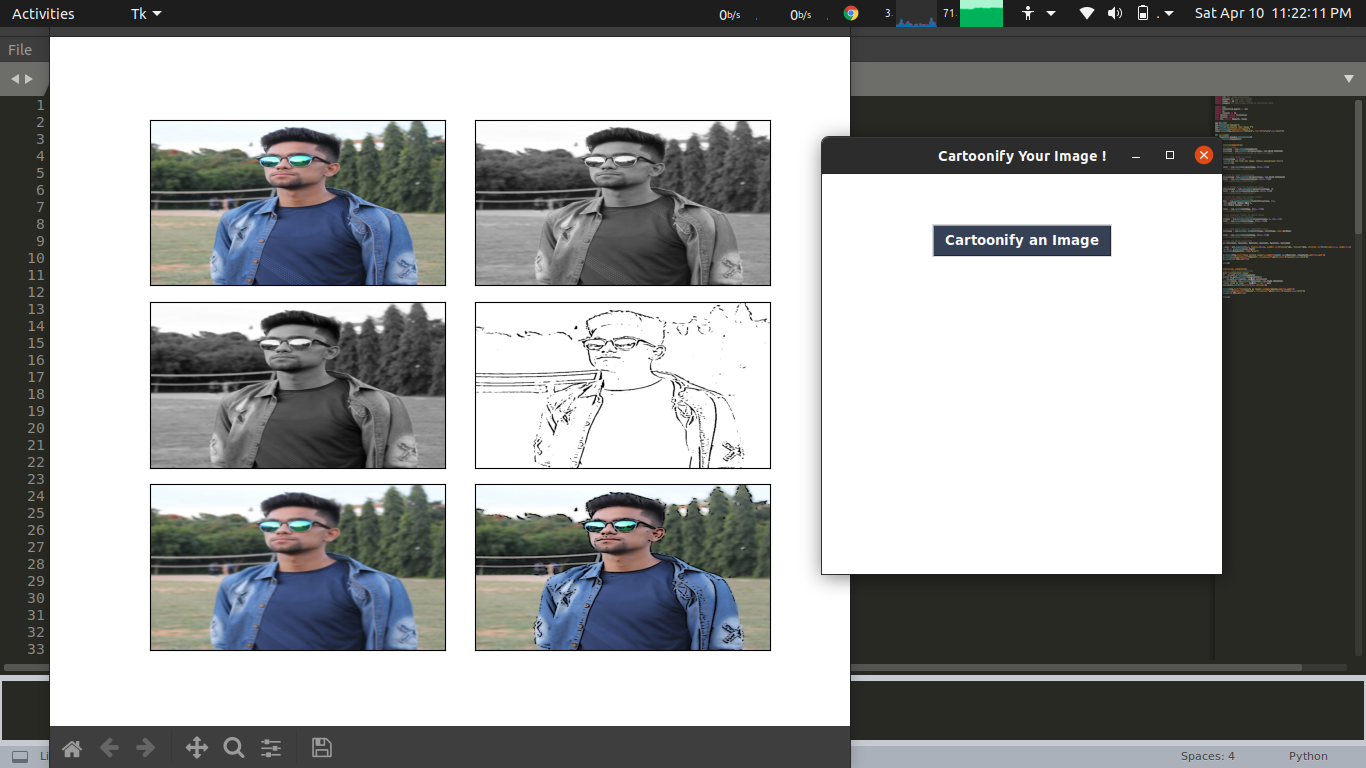
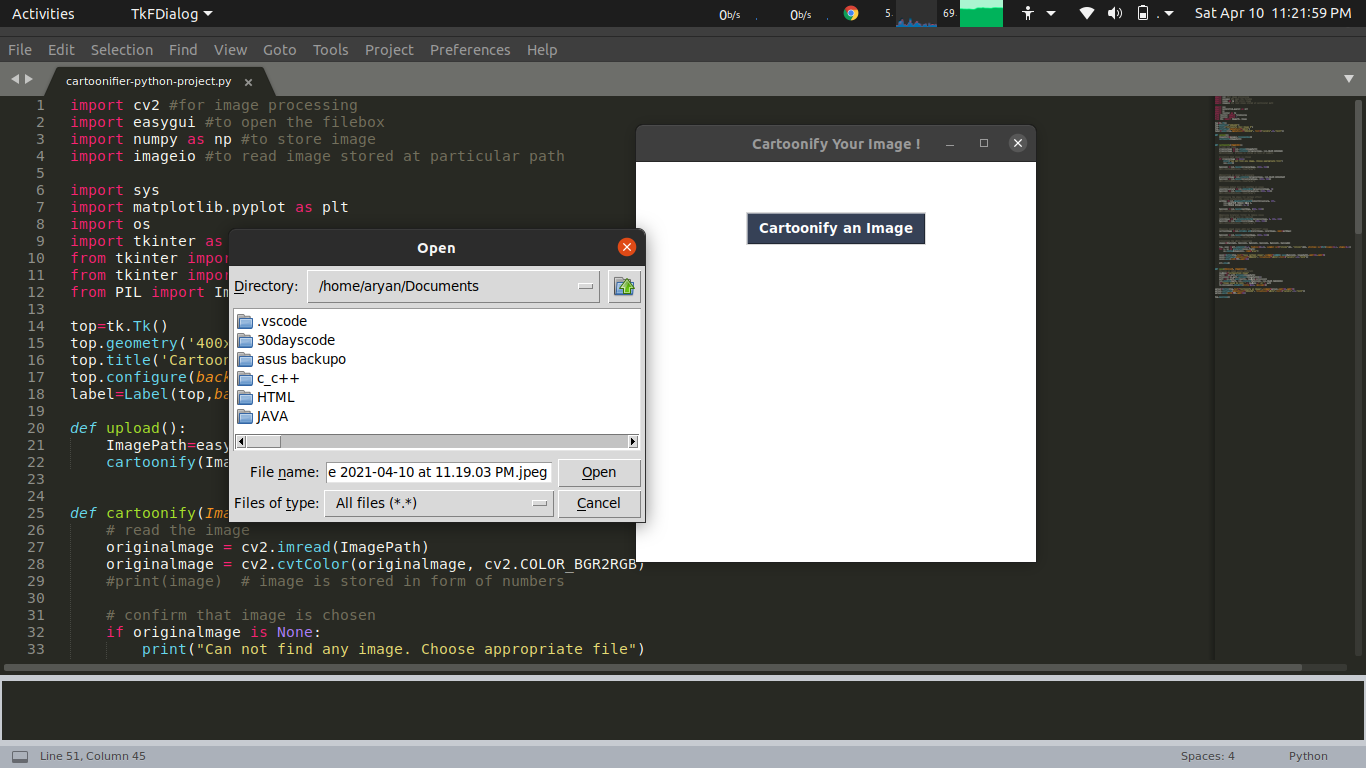
**SCREENSHOTS:**

**code:**



Output:







**CHAPTER3**

**CONCLUSION**

In this project, I applied edge detection and image filtering to achieve a cool cartoon effect .We can also used colour quantization method for this project.

### References:

[1] OpenCV – Python tutorials

[2] OpenCV: Computer Vision Projects with Python by Joseph Howse, Prateek Joshi, Michael Beyeler

[3] How to create a cartoon effect – Opencv with Python by Sergio Canu

**THANK YOU**