AJEENKYA D Y PATIL UNIVERSITY
THE INNOVATION UNIVERSITY
School of Engineering

A

Mini Project Report

on

"Cartoonify an Image with OpenCV in Python"

oof the degree of

Bachelor of Technology – B. Tech (ITDS)

by

Tauqeer Shaikh

URN NO: 2019-B-21012001C

Under the Guidance of

Siddharth Nanda

AJEENKYA
D Y PATIL UNIVERSITY
THE INNOVATION UNIVERSITY

December 2021

School of Engineering

Ajeenkya D Y Patil University, Pune

AJEENKYA D Y PATIL UNIVERSITY
THE INNOVATION UNIVERSITY
School of Engineering

Declaration of Originality

I, ***Om Bhagwan Kathe,*** URN ***2020-B-01022001***, hereby declare that this project entitled "***Cartoonify an Image with OpenCV in Python"*** presents my original work carried out as a bachelor student at School of Engineering, Ajeenkya D Y Patil University, Pune, Maharashtra. To the best of my knowledge, this project report contains no material previously published or written by another person, nor any material presented by me for the award of any degree or diploma of Ajeenkya D Y Patil University, Pune or any other institution. Any contribution made to this research by others, with whom I have worked at Ajeenkya D Y Patil University, Pune or elsewhere, is explicitly acknowledged in the project report. Works of other authors cited in this project report have been duly acknowledged under the sections "Reference" or "Bibliography". I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission.

I am fully aware that in case of any non-compliance detected in future, the Academic Council of Ajeenkya D Y Patil University, Pune may withdraw the degree awarded to me on the basis of the present project report.

Date: 09-12-2021

Place: Lohegaon, Pune

_____

Tauqeer Shaikh

Acknowledgement

I remain immensely obliged to **Prof. Siddharth Nanda** – Project Supervisor, for providing me with the idea of this topic, and for his invaluable support in garnering resources for me either by way of information or computers also his guidance and supervision which made this Internal Project happen.

I would like to thank **Prof. Siddharth Nanda,** Program Coordinator B.Tech, and **Dr. Biswajeet Champaty**, Head of Department for their invaluable support.

I would like to say that it has indeed been a fulfilling experience for working out this Internal Project.

Tauqeer Shaikh

AJEENKYA D Y PATIL UNIVERSITY
THE INNOVATION UNIVERSITY
School of Engineering

Dec 2021

CERTIFICATE

This is to certify that the project entitled **"Cartoonify an Image with OpenCV in Python"** is a bonafide work of "**Om Bhagwan Kathe" (URN-2020-B-01022001)** submitted to the Ajeenkya D Y Patil University, Pune in partial fulfilment of the requirement for the award of the degree of **"Bachelor of Technology (B.Tech) in Data Science"**.

_____

Prof. Siddharth Nanda

Project Supervisor

AJEENKYA D Y PATIL UNIVERSITY
THE INNOVATION UNIVERSITY
School of Engineering

Dec 2021

Supervisor's Certificate

This is to certify that the project entitled **"Cartoonify an Image with OpenCV in Python"** submitted by **Om Bhagwan Kathe, URN: 2020-B-01022001,** is a record of original work carried out by her under my supervision and guidance in partial fulfillment of the requirements of the degree of **Bachelor of Technology (B.Tech)** at **School of Engineering**, **Ajeenkya D Y Patil University, Pune, Maharashtra 412105**. Neither this project report nor any part of it has been submitted earlier for any degree or diploma to any institute or university in India or abroad.

_____

Prof. Siddharth Nanda

Project Supervisor

ABSTRACT

Image Processing – In the field of the research processing of an image consisting of identifying an object in an image, identify the dimensions, no of objects, changing the images to blur effect and such effects are highly appreciated in this modern era of media and communication.There are multiple properties in the Image Processing. Each of the property estimates the image to be produced more with essence and sharper image.Each Image is examined to various grid. Each picture element together is viewed as a 2-D Matrix. With each of the cell store different pixel values corresponding to each of the picture element.

This project represents different techniques of converting image to cartoon. Using any one of below mentioned techniques it is possible to convert all types of captured images to cartoon such as images of person, mountains, trees, flora and fauna etc. There are several other techniques for image to cartoon conversion such as using photoshop, adobe illustrator, windows MAC, paint.net and much more.

LIST OF FIGURES

FIGURE 1 CODE FOR IMPORTING THE REQUIRED MODEL **15**

FIGURE 2 CODE FOR BILDING A FILEBOX TO CHOOSE A PARTICULAR FILE **16**

FIGURE 3 CODE FOR HOW IS AN IMAGE STORED **16**

FIGURE 4 CODE FOR TRANSFORMING AN IMAGE TO GRAYSCALE **17**

FIGURE 5 OUTPUT AFTER TRANSFORMING AN IMAGE TO GRAYSCALE **17**

FIGURE 6 CODE FOR SMOOTHENING OF GRAYSCALE IMAGE 18

FIGURE 7 OUTPUT AFTER SMOOTHEING OF GRAYSCALE IMAGE 18

FIGURE 8 CODE FOR RETRIVING THE EDGES OF AN IMAGE 18

FIGURE 9 OUTPUT AFTER RETRIVING THE EDGES OF AN IMAGE 19

FIGURE 10 CODE FOR PREPARING A MASK IMAGE **19**

FIGURE 11 OUTPUT AFTER PREPARING A MASK IMAGE **20**

FIGURE 12 CODE FOR GIVING A CARTOON EFFECT **20**

FIGURE 13 OUTPUT AFTER GIVING A CARTOON EFFECT **21**

FIGURE 14 CODE FOR PLOTTING ALL THE TRANSITIONS TOGETHER **21**

FIGURE 15 OUTPUT AFTER PLOTTING ALL THE TRANSITIONS TOGETHER **22**

FIGURE 16 CODE FOR FUNCTIONALLY OF SAVE BUTTON **22**

FIGURE 17 CODE FOR MAKING THE MAIN WINDOW **23**

FIGURE 18 CODE FOR MAKING THE CARTOONIFY BUTTON IN THE MAIN WINDOW **23**

FIGURE 19 CODE FOR MAKING THE C BUTTON IN THE MAIN WINDOW 24

FIGURE 20 CODE FOR MAKING MAIN FUNCTION TO BUILD THE TKINTER WINDOW 24

FIGURE 21 1ST SCRENSHOT OF CODE 25

FIGURE 22 2ND SCREENSHOT OF CODE 25

FIGURE 23 3RD SCREENSHOT OF CODE 26

FIGURE 24 4TH SCREENSHOT OF CODE 26

FIGURE 25 1ST OUTPUT SCREEN 27

FIGURE 26 2ND OUTPUT SCREEN 27

FIGURE 27 3RD OUTPUT SCREEN 28

TABLE OF CONTENTS

DECLARATION OF ORIGINALITY		2
ACKNOWLEDGEMENT		3
CERTIFICATE		4
SUPERVISOR'S CERTIFICATE		5
ABSTRACT		6
LIST OF FIGURES		7-8
CHAPTER 1 INTRODUCTION		10
1.1	INTRODUCTION TO PROJECT	11
1.2	NEED OF PROJECT	12-13
1.3	PROPOSED SYSTEM	14
CHAPTER 2 CODE AND IMPLEMENTATION		15
2.1	WHAT IS OPEN CV?	15
2.2	STEPS TO DEVELOP IMAGE CARTOONFIER	15
2.3	IMPORTING THE REQUIRED MODULES	15
2.4	BUILDING A FILEBOX TO CHOOSE A PARTICULAR FILE	16
2.5	HOW IS AN IMAGE STORED?	16
2.6	TRANSFORMING AN IMAGE TO GRAYSCALE	17
2.7	SMOOTHENING A GRAYSCALE IMAGE	18
2.8	RETRIVING THE EDGES OF AN IMAGE	18
2.9	PREPARING A MASK IMAGE	19
2.10	GIVING A CARTOON EFFECT	20
2.11	PLOTTING ALL THE TRANSACTIONS TOGETHER	21
2.12	FUNCTIONALLY OF SAVE SAVE BUTTON	22
2.13	MAKING THE MAIN WINDOW	23
2.14	MAKING THE CARTOONIFY BUTTON WINDOW	23
2.15	MAKING OF SAVE AND FUNCTION BUTTON	24
2.16	FINAL CODE ON PYCHARM	25-26
2.17	FINAL OUTPUT	27-28
CHAPTER 3 CONCLUSION		29
CHAPTER 4 BIBLIOGRAPGY		30

CHAPTER 1 INTRODUCTION

1.1 INTRODUCTION TO PROJECT

Social media is extensively used these days. And standing out in this online crowd has always been a to-do on every user's list on these social media platforms. Be it images, blog posts, artwork, tweets, memes, opinions and what not being used to seek attention of followers or friends to create influence or to connect with them on such social platforms.

We aim to provide one such creative solution to their needs, which is applying cartoon like effects to their images. Users can later share these images on any social media platforms, messengers, keep it for themselves, share it with loved ones or do whatever they like with it.

Nowadays almost everyone is registered in social networks. We keep online status updated every day, share photos and comments, follow our friends' news. To have a nice profile is a matter of prestige.

You can use a photo of your own in a profile image, create an amusing avatar or turn your photo into a cartoon. With a pool of web applications available online, an image conversion to cartoon takes few clicks.

Advanced technology has become the integral part of our life. To satisfy the need of the society, almost in each work, we use the technology. In current era computer science is major subject.

It has many real life applications such as cloud computing, NSPP,remote monitoring, Wireless sensor network, uncertainty, internet of things, Neural network, artificial intelligence, FSPP, TP, internet Security, and so on.

Technology is the mode by which user can store, fetch, communicate and utilize the information. The image processing plays a major role in all computers related applications.

The image processing appears in many real-life applications, e.g., home security, banking system, education sector, defense system, Railway, and so on. In this manuscript we discuss about the cartooning of image. Each of these methodologies

offers a rapid contribution to human interest. Each confined methodology helps in filtering the picture element that forms to an image.

There are various factors that enables to produce the essence of an image. The concerns are contrasting and appropriate color mixing, matching between any two pixels connecting two cells, accurate placing of objects together combined to form image features. In the recent times there happened to be drastic changes in ample fields. The uplift of these fields enhances in betterment of the society.

In the field of medicine, these processing of images enable to extract the fullest accuracy of the images. Image Processing is widely processed in the medical field such as in the MRI/ET scans. The amount of research in the image processing has helped to acquire early detection of tumors. There plays a vital role in the field of image processing and in the field of Biology.

This research bound to save livelihood as early detection can be identified and effective treatment can be started off. These extended concepts have enabled to build better security systems which ensure safety.

The security/surveillance systems have managed to build systems depending on the image processing algorithms, The recent technology of fingerprint unlock, face detection unlock has resulted in developing an efficient security. These Biometric systems perhaps have been now installed on to smaller devices as well for the simpler usage.

With the recent success apprehended by the social media is duly with the techniques installed to enhance the user experience. E.g. – Facebook confines with the auto tag mechanism to automatically suggest the person's name and not by manually tagging each person on the image.

The basic concept in this algorithm involves the technique of converting the RGB colour image to an accurate, cartooned image without multiple filtrations or blurred image without proper facilitation of edge detection.

This user interface allows to apply the animation effects. This naturally provides an artistic effect and comics as well with wide range of pictures.

1.2 NEED OF PROJECT

Creating a cartoon like effect is time and space consuming. Existing solutions to provide cartoon like effect to images are complex. Some solutions involve installing complex photo editing software like photoshop and other involve performing some task by user.

Our research shows a website to carry out the task of Applying effects is more suitable, space efficient and takes minimum user efforts, for example toony photos is an existing website to perform such task but it is difficult to use as user has to markdown points & lines on the image to apply effects which is not user friendly also the options are limited. Hence there is a dire need for a website which is user friendly and performs the task of applying effects to images very well.

Following is our brief research on existing solutions:

A. Cartoon Effect The majority of photo editing websites offer the socalled Cartoon Effect. The main advantages of online photo to cartoon effect apps are simplicity and quickness. You'll have to upload a photo from your computer or from the web, find Cartoon Effect in the tool set or choose between styles or variants of this funny photo effect (like in case of www.picturetopeople.org, Kuso Cartoon) and press the button Apply (or Go).

   The image processing varies from several seconds up to 1-2 minutes. However, as all quick online solutions these apps have drawbacks. A lot of photo online photo editing tools are rather humdrum because they are deprived of enhancement features. In these apps cartoonization is limited to 1-click operation. Besides, sometimes colors may become blurred and it leads to an unsatisfactory result. Such apps as www.converttocartoon.com, Photo.to, AnyMaking and others belong to this group. At the same time there are online photo editors with more advanced tools.

   They have a variety of adjustment options. For example, BeFunky helps you modify sketch brightness, contrast, smoothness and other details.

B. Pencil Sketch Another means of cartoonization is making a pencil sketch out of your digital photograph. Whenever you apply Cartoon effect your images turn bright and cheerful.

   If you want to render a solid atmosphere and achieve respectability in your online profile pencil sketch creation will suit your needs better. The image manipulation procedure is just the same as described for the cartoon effect.

Upload a photo, select the desired effect, push the button Apply and you are done. The application does its job instantly by itself. PhotoSketcher, Fotosketcher, Dumpr, Tuxpi photo editor and many other applications give you an opportunity to convert your snaps to life-like pencil sketches.

Besides, you can decorate your profile photo with a cute photo frame and even create a photo with your favorite cartoon character. Amaze your nearest and dearest, friends and coworkers with a cool profile photo, stand out from the crowd and attract more followers and fans in social networks. You know, the first impression is the strongest.

How to turn photo into cartoon online or on Windows/Mac Moreover, sharing a photo cartoon on social media could attract more attention when others just post standard photos. We are going to share how to turn photo to cartoon on Windows, Mac, and online in this tutorial.

With these photo editors and our guides, you can create cartoon at any time, even if you have not learnt any knowledge about painting. If you are ready, let’s start right now.

1.3 PROPSED SYSTEM

We propose to use neural style transfer which is a machine learning algorithm, which involves two images, first is the input image from the user and second is the style image which is used to apply the style on the input image. Following are the examples of images generated using neural style transfer.

We propose to create a website, which consists of image upload functionality using which the user can upload his image, the uploaded image is then processed by server using Neural style transfer algorithm and the resulting image is presented to the user on the website. Which then user can download & share. Neural fast style transfer is used by Apps such as https://deepart.io, Prisma, Artisto etc.

We decided to choose this approach over traditional image filters (e.g. using image filters such as median & bilateral filters to posterize an Image) as Neural fast style transfer is quite new and challenging technique which uses machine learning & image processing to produce various styled images based on variety of input & style images. The algorithm can be implemented in Python/JavaScript/Lua to perform neural style transfer.

We will use Python to implement the backend and the front end of the website will be in HTML, CSS & JS. Basically, in Neural Style Transfer we have two images- style and content. We need to copy the style from the style image and apply it to the content image. By, style we basically mean, the patterns, the brushstrokes, etc. we will provide a set of style images which a user can use to apply different kinds of Cartoon like effects to his image

CHAPTER 2 CODE AND IMPLEMENTATION

2.1 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.

2.2 STEPS TO DEVELOP IMAGE CARTOONIFIER :

2.3 STEP 1: IMPORTING THE REQUIRED MODULES

We will import the following modules:

- **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.

CODE:

```
1.  import cv2 #for image processing
2.  import easygui #to open the filebox
3.  import numpy as np #to store image
4.  import imageio #to read image stored at particular path
5.  import sys
6.  import matplotlib.pyplot as plt
7.  import os
8.  import tkinter as tk
9.  from tkinter import filedialog
10. from tkinter import *
11. from PIL import ImageTk, Image
```

Figure 1 Code for importing the require models.

2.4 STEP 2: BUILDING A FILE BOX TO CHOOSE A PARTICULAR FILE

In this step, we will build the main window of our application, where the buttons, labels, and images will reside. We also give it a title by title() function.

CODE:

```
""" fileopenbox opens the box to choose file
and help us store file path as string """
def upload():
    ImagePath=easygui.fileopenbox()
    cartoonify(ImagePath)
```

Figure 2 Code for Building a file box to choose a particular file.

EXPLANATION:-

The above code opens the file box, i.e the pop-up box to choose the file from the device, which opens every time you run the code. fileopenbox() is the method in easyGUI module which returns the path of the chosen file as a string.

2.5 STEP 3: HOW IS AN IMAGE STORED?

Now, just think, how will a program read an image? For a computer, everything is just numbers. Thus, in the below code, we will convert our image into a numpy array.

CODE:

```
#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')
```

Figure 3 Code for how is an image stored

EXPLANATION:

Imread is a method in cv2 which is used to store images in the form of numbers. This helps us to perform operations according to our needs. The image is read as a numpy array, in which cell values depict R, G, and B values of a pixel

Beginning with image transformations:

To convert an image to a cartoon, multiple transformations are done. Firstly, an image is converted to a Grayscale image. Yes, similar to the old day's pictures.! Then, the Grayscale image is
Let's start with these transformations to convert an image to its cartoon image.

2.6 STEP 4: TRANSFORMING AN IMAGE TO GRAYSCALE

smoothened, and we try to extract the edges in the image. Finally,we form a color image and mask it with edges. This creates a beautiful cartoon image with edges and lightened color of the original image.

CODE:

```
#converting an image to grayscale
grayScaleImage = cv2.cvtColor(originalmage, cv2.COLOR_BGR2GRAY)
ReSized2 = cv2.resize(grayScaleImage, (960, 540))
#plt.imshow(ReSized2, cmap='gray')
```

Figure 4 Code for Transforming an image to grayscale

EXPLANATION :

- cvtColor(image, flag) is a method in cv2 which is used to transform an image into the colour- space mentioned as 'flag'. Here, our first step is to convert the image into grayscale. Thus, we use the BGR2GRAY flag. This returns the image in grayscale. A grayscale image is store d as grayScaleImage.

- After each transformation, we resize the resultant image using the resize() method in cv2 and display it using imshow() method. This is done to get more clear insights into every single transformation step.

OUTPUT:-

Figure 5 Output for grayscale image

2.7 STEP 5: SMOOTHENING A GRAYSCALE IMAGE

CODE:-

```
#applying median blur to smoothen an image
smoothGrayScale = cv2.medianBlur(grayScaleImage, 5)
ReSized3 = cv2.resize(smoothGrayScale, (960, 540))
#plt.imshow(ReSized3, cmap='gray')
```

Figure 6 Code for Smoothening a grayscale image

EXPLANATION:-

To smoothen an image, we simply apply a blur effect. This is done using medianBlur() function. Here, the center pixel is assigned a mean value of all the pixels which fall under the kernel. In turn, creating a blur effect. This code will show the following output.

OUTPUT:-

Figure 7 Output after smoothening a grayscale image

2.8 STEP 6: RETRIEVING THE EDGES OF AN IMAGE

CODE:-

```
#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')
```

Figure 8 Code for Retrieving the Edges of an image

EXPLANATION:-

- Cartoon effect has two specialties: 1.Highlighted Edges & 2. Smooth Colours

- In this step, we will work on the first specialty. Here, we will try to retrieve the edges and highlight them. This is attained by the adaptive thresholding technique The threshold value is the mean of the neighborhood pixel values area minus the constant C. C is a constant that is subtracted from the mean or weighted sum of the neighborhood pixels. Thresh_binary is the type of threshold applied, and the remaining parameters determine the block size.

OUTPUT:-

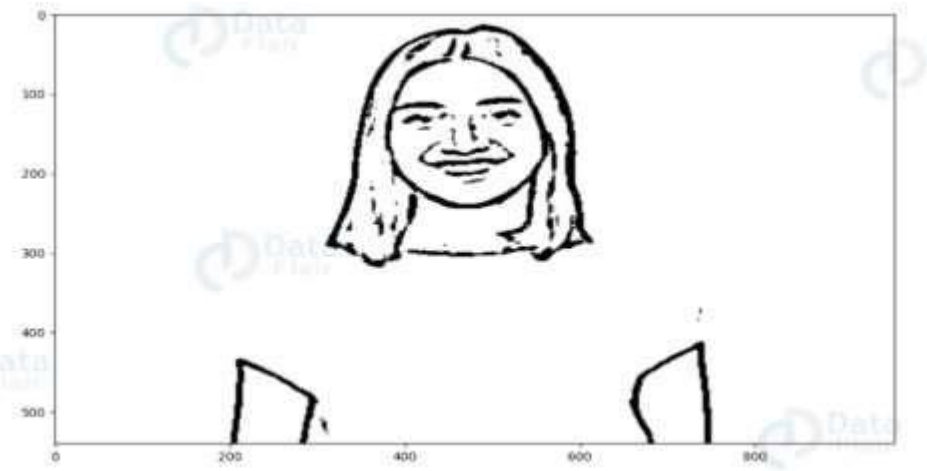


Figure 9 Output after Retrieving an Edges of an image

2.9 STEP 7: PREPARING A MASK IMAGE

CODE:-

```
#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')
```

Figure 10 Code for Preparing a Mask image

EXPLANATION:-

- In the above code, we finally work on the second specialty. We prepare a lightened color image that we mask with edges at the end to produce a cartoon image. We use bilateralFilter which removes the noise. It can be taken as smoothening of an image to an extent.

- The third parameter is the diameter of the pixel neighborhood, i.e, the number of pixels around a certain pixel which will determine its value. The fourth and Fifth parameter defines signmaColor and sigmaSpace. These parameters are used to give a sigma effect, i.e make an image look vicious and like water paint, removing the roughness in colors.

- Yes, it's similar to BEAUTIFY or AI effect in cameras of modern mobile phones.

OUTPUT:-

Figure 11 Output after preparing a mask image

2.10 STEP 8: GIVING A CARTOON EFFECT

CODE:-

```
#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')
```

Figure 12 Code for Giving a Cartoon Effect

EXPLANATION:-

- So, let's combine the two specialties. This will be done using MASKING. We perform bitwise and on two images to mask them. Remember, images are just numbers?

- Yes, so that's how we mask edged image on our "BEAUTIFY" image.

- This finally CARTOONIFY our image!

OUTPUT:-

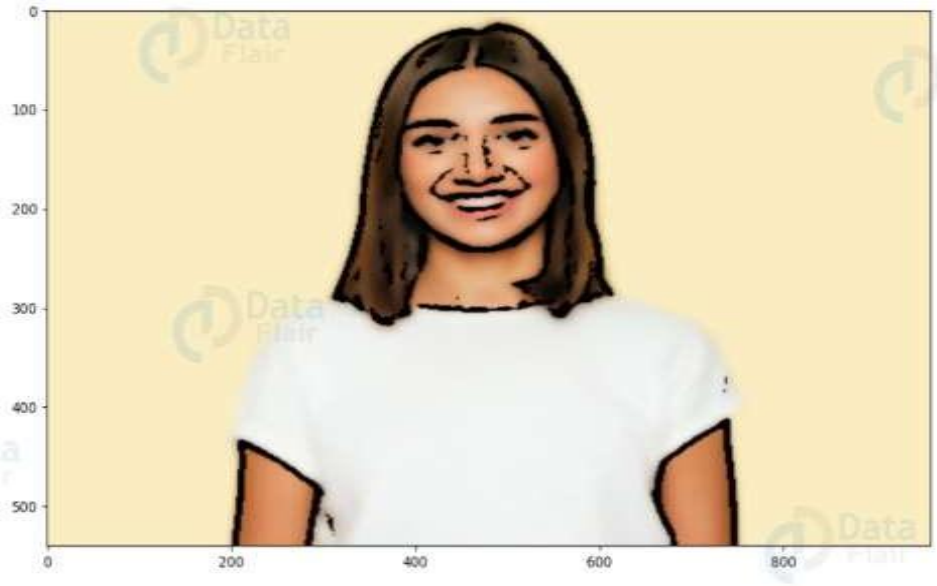


Figure 13 Output after giving a cartoon effect

2.11 STEP 9: PLOTTING ALL THE TRANSITIONS TOGETHER

CODE:-

```
# 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')
//save button code
plt.show()
```

Figure 14 Code for Plotting all the transitions together

EXPLANATION:-

- To plot all the images, we first make a list of all the images. The list here is named “images” and contains all the resized images. Now, we create axes like subl=plots in a plot and display one-one images in each block on the axis using imshow() method.

- plt.show() plots the whole plot at once after we plot on each subplot.

OUTPUT:-

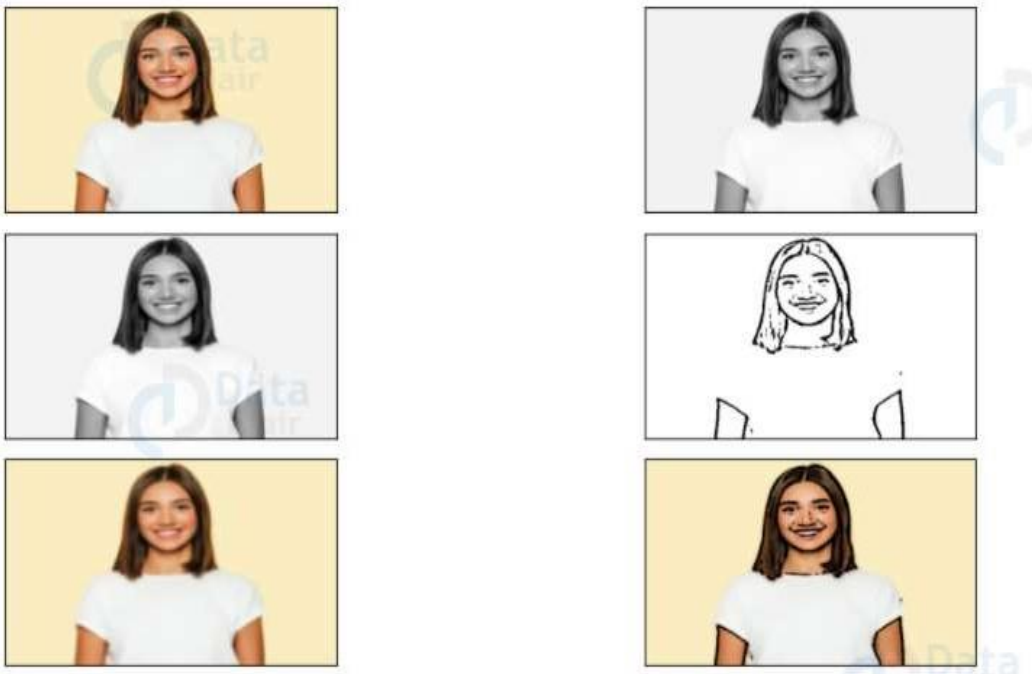


Figure 15 Output after plotting all the transaction together

2.12 STEP 10: FUNCTIONALLY OF SAVE BUTTON

CODE:-

```
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)
```

Figure 16 Code for Functionally of save Button

EXPLANATION:-

- Here, the idea is to save the resultant image. For this, we take the old path, and just change the tail (name of the old file) to a new name and store the cartoonified image with a new name in the same folder by appending the new name to the head part of the file.

- For this, we extract the head part of the file path by os.path.dirname() method. Similarly, os.path.splitext(ImagePath)[1] is used to extract the extension of the file from the path.

- Here, newName stores "Cartoonified_Image" as the name of a new file. os.path.join(path1, newName + extension) joins the head of path to the newname and extension. This forms the complete path for the new file.

- imwrite() method of cv2 is used to save the file at the path mentioned. cv2.cvtColor(ReSized6, cv2.COLOR_RGB2BGR) is used to assure that no color get extracted or highlighted while we save our image. Thus, at last, the user is given confirmation that the image is saved with the name and path of the file.

2.13 STEP 11: MAKING THE MAIN WINDOW

CODE:-

```
top=tk.Tk()
top.geometry('400x400')
top.title('Cartoonify Your Image !')
top.configure(background='white')
label=Label(top,background='#CDCDCD', font=('calibri',20,'bold'))
```

Figure 17 Making the main window

2.14 STEP 12: MAKING THE CARTOONIFY BUTTON IN THE MAIN WINDOW

CODE:-

```
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)
```

Figure 18 Code for Making the cartoonify button in the main window

2.15 STEP 13: MAKING A SAVE BUTTON IN THE MAIN WINDOW

CODE:-

```
1.  save1=Button(top,text="Save cartoon image",command=lambda: save(ImagePath,
    ReSized6),padx=30,pady=5)
2.  save1.configure(background='#364156', foreground='white',font=('calibri',10,'bold'))
3.  save1.pack(side=TOP,pady=50)
```

Figure 19 Code for making a save button in the main window

Step 14: main function to build the tkinter window

CODE:-

```
1.  top.mainloop()
```

Figure 20 Code for making a Main function to build the tkinter window

2.16 FINAL CODE ON PYCHARM :-

main.py code.py

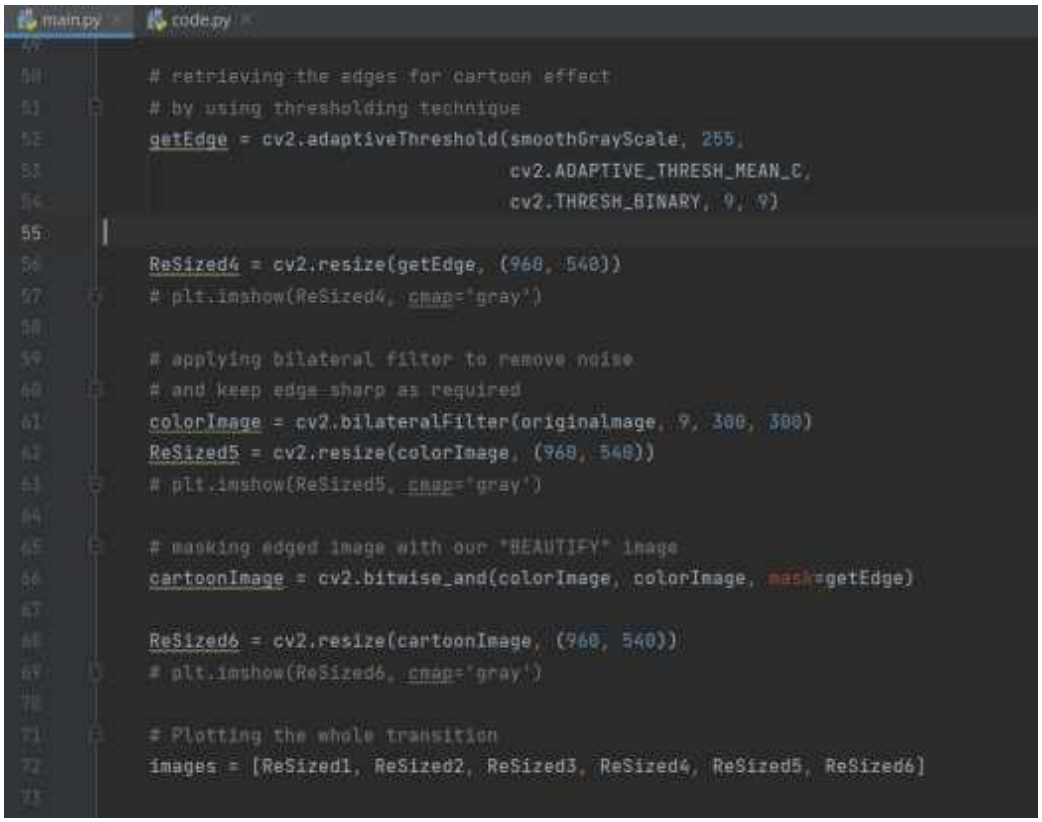
```
import cv2  # for image processing
import easygui  # to open the filebox
import numpy as np  # to store image
import imageio  # to read image stored at particular path

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)
```

Figure 21 1st Screenshot of Code

main.py code.py

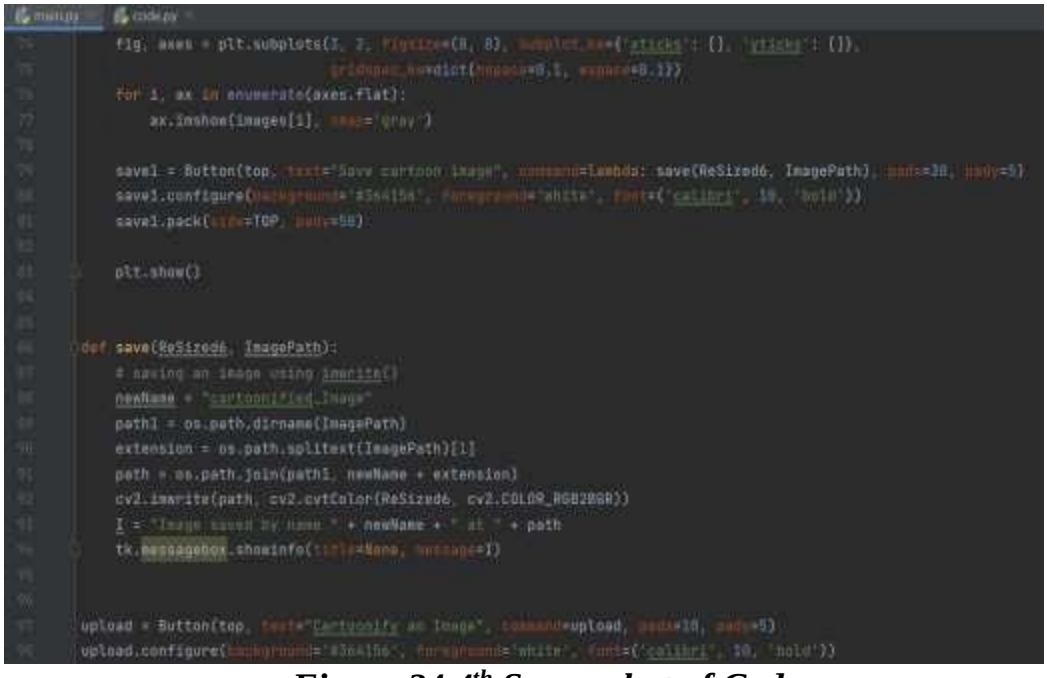
```
def cartoonify(ImagePath):
    # read the image
    originalmage = cv2.imread(ImagePath)
    originalmage = cv2.cvtColor(originalmage, cv2.COLOR_BGR2RGB)
    # print(image)  # image is stored in form of numbers

    # 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
```

Figure 22 2nd Screenshot of Code



```
main.py   code.py

    # 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]
```

Figure 23 03rd Screenshot of Code



```
    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'))
```

Figure 24 4th Screenshot of Code

OUTPUT:-

Step 1:- Run the Code and Click on Cartoonify on image Button.

Figure 25 1st Output screen

Step 2:- Select the Photo which you want to Cartoonify.

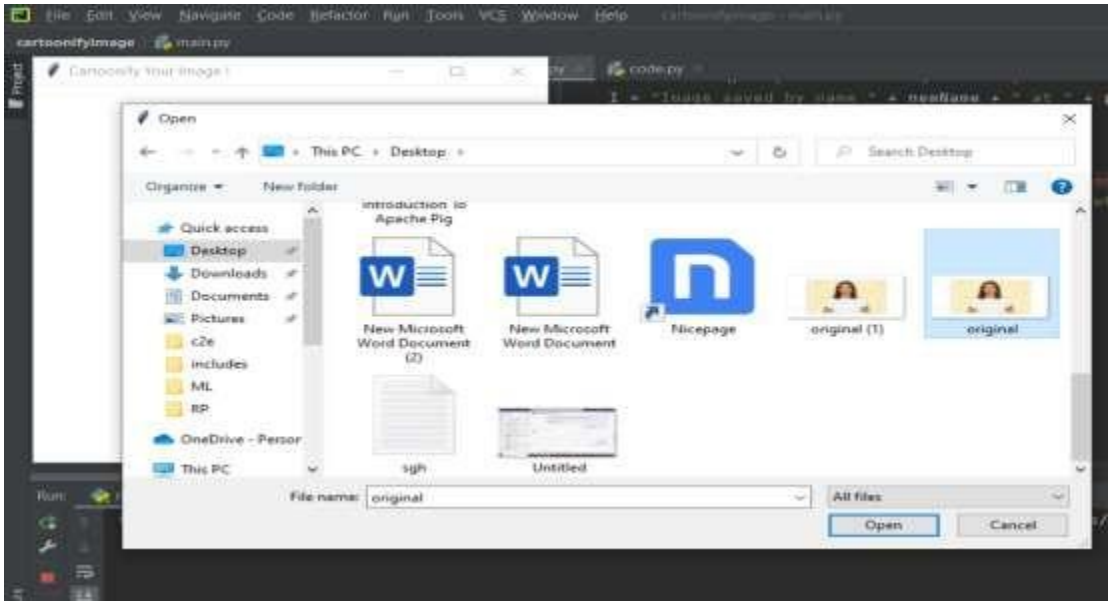


Figure 26 2nd Output screen

Step 3:- After that You will get the Cartoonify image of your Image and you can save it.

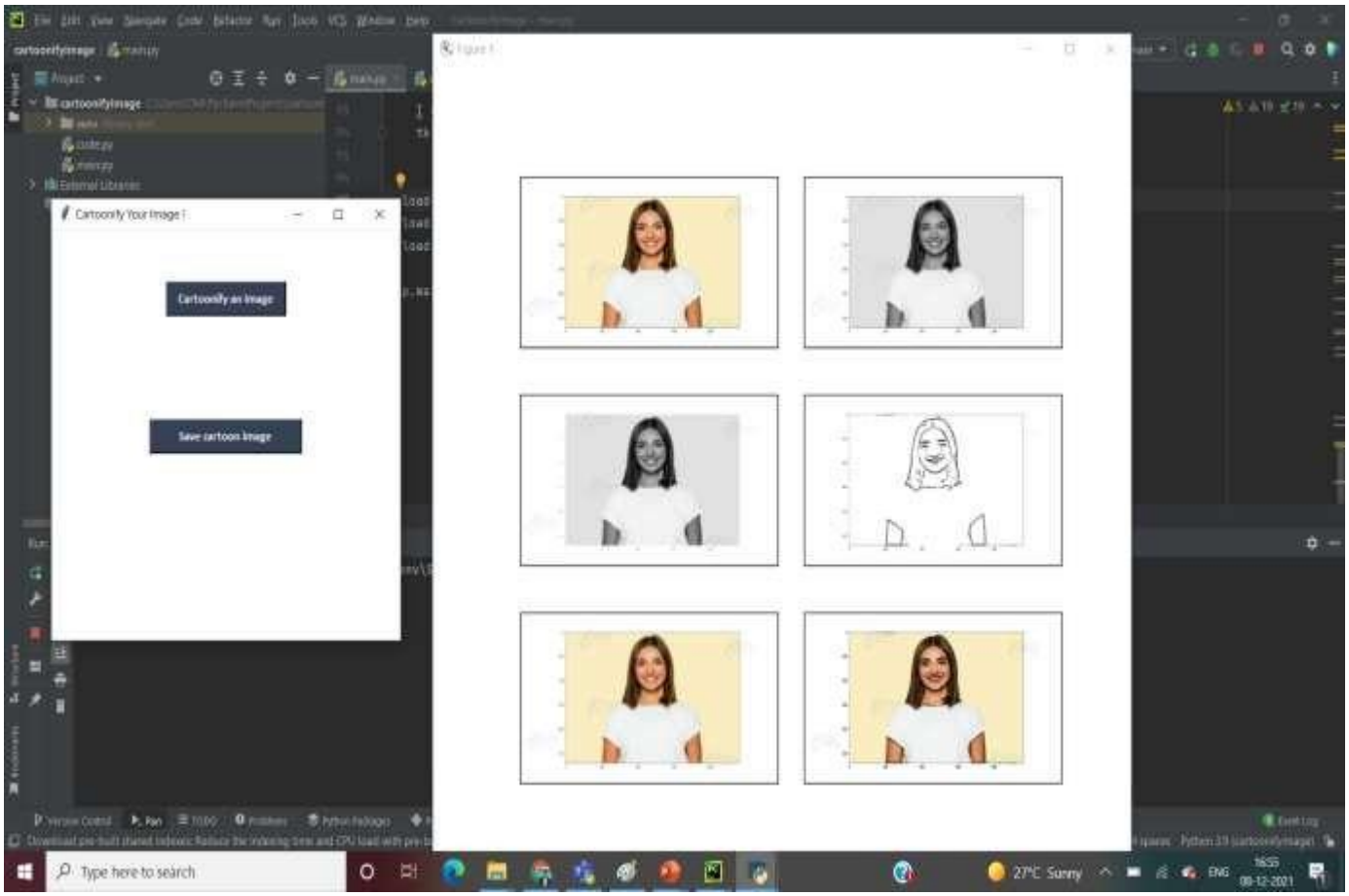


Figure 27 03rd Output screen

CHAPTER 3 CONCLUSION

All things considered, the Toonify algorithm is a success. It is capable of producing exactly the effect specified above, and a wide range of input images will yield satisfactory results. However, the algorithm is not perfect, and it does not respond predictably across all inputs.

Given the variety of input images, it would be unrealistic to expect a one-size-fits-all approach to produce consistent results. In the future, an adaptive algorithm would probably be better suited to providing a consistent effect. Such an algorithm would recognize image types (portrait, indoor scene, outdoor scene for instance) and then change the edge detection parameters to better suit that particular image. Additionally, the algorithm may rely on an image derived from the hue, rather than the luminance, for edge detection.

This might aid the algorithm in drawing contours around regions of different colors. Currently, the algorithm seems best suited for images that have a few large regions, each with relatively low color diversity. Typically the images that fail are those with a high amount of local color variation and detail.

CHAPTER 4 BIBLIOGRAPGY

- ***Baggio, Daniel L, Mastering OpenCV with Practical Computer Vision Projects Packt Publishing Ltd, 2012***
- ***https://data-flair.training/blogs/cartoonify-image-opencv-python/***
- ***Introduction to Machine Learning By: John Muel***
- ***Image Style Transfer Using Convolutional Neural Networks, 2016 - Leon A. Gatys, Alexander S. Ecker, Matthias Bethge***