Cartoonify an Image

A Project Report Submitted in Fulfillment of the Requirements for the Award of Degree

Of

Bachelor of Technology

In

Computer Science and Engineering

By

Prince Kumar (Roll No.-11192558)

Parth Goel (Roll No. - 11192619)

Akanksha Raj (Roll No. - 11192663)

Abhishek Pundir(Roll No.-11192586)

Under the supervision of

Dr. Puneet Banga
(Assistant Professor)

M. M. Engineering College, Mullana, Ambala

Maharishi Markandeshwar (Deemed to be University), Mullana, Ambala, Haryana, India

May 2022

Department of Computer Science &Engineering

M. M.EngineeringCollege, Mullana, Ambala, Haryana, India

Candidate's Declaration

We hereby certify that the work which is being presented in the project entitled "**Cartoonify an image**" in fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering of M.M Engineering College, Mullana, Ambala, Haryana, India is an authentic record of our own work carriedout during a period from July 2021 to Nov 2021, under the supervision of **Dr. Puneet Banga (Assistant Professor).** The matter presented in this project report has not been submitted by us for the award of any degree of this or any other Institute/University.

Prince Kumar	**Parth Goel**	**Akanksha Raj**	**Abshishek Pindir**
(11192558)	**(11192619)**	**(11192663)**	**(11192586)**

This is to certify that the above statement made by the above candidates is correct to the best of my knowledge.

Date: 2 May 2022

Dr. Puneet Banga

(Assistant Professor)

Department of Computer Science &Engineering
M.M.EngineeringCollege, Mullana, Ambala, Haryana, India

Acknowledgement

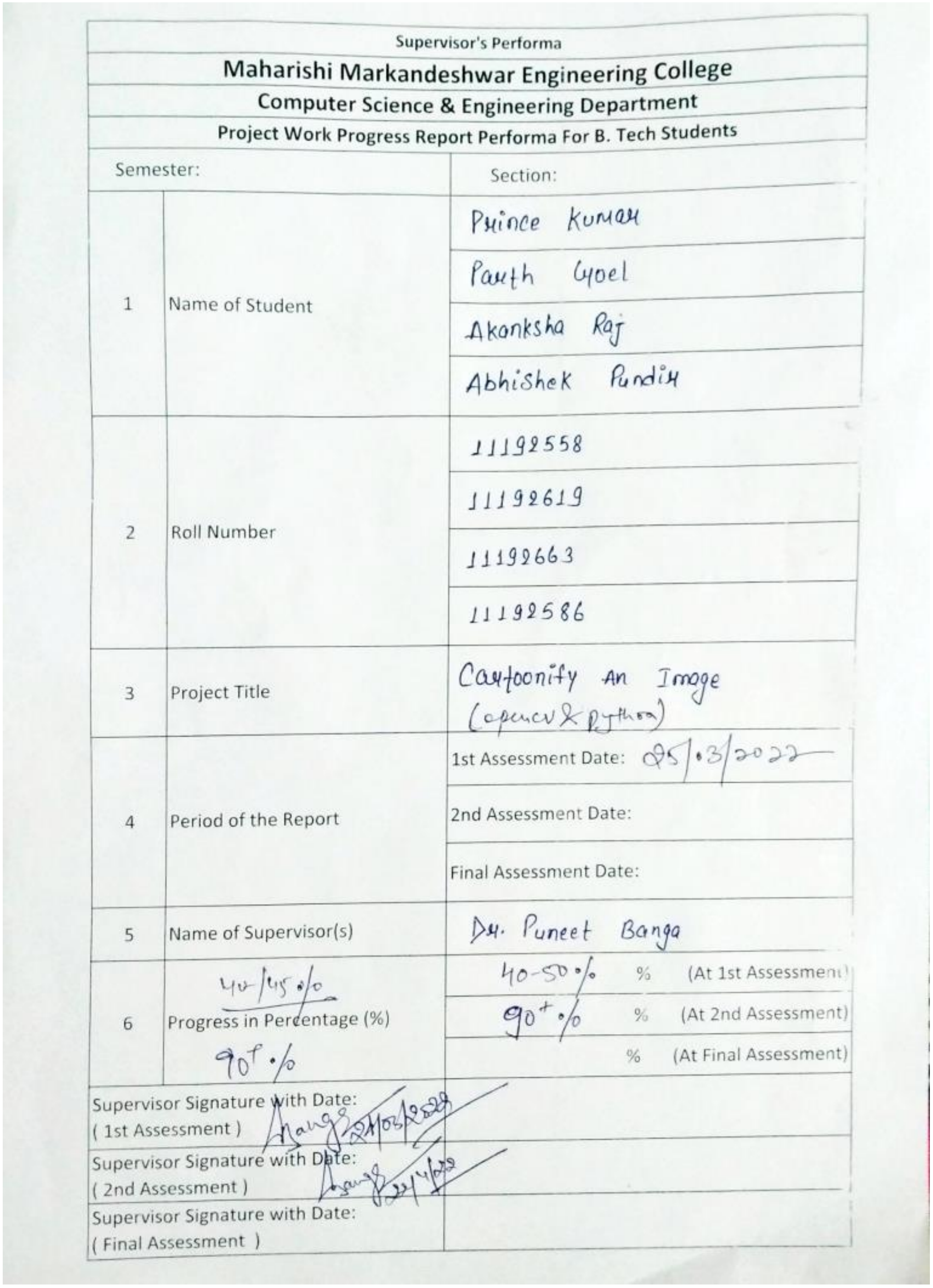
We wish to express my deep sense of indebtedness and sincerest gratitude to our guide, **Dr. Puneet Banga (Assistant Professor)** for his/her invaluable guidance and constructive criticism throughout this work. She/he has displayed unique tolerance and understanding at every step of progress and encouragesus. We deem it our privilege to have carried out our project work under his/her able guidance.

We would especially like to thank **Dr. Sandip Kumar Goel (Professor and Head, CSE Department, MMEC)** without whom, this work would not have been as it is now.

As a Final Personal Note, We are grateful to our parents, who are inspirational to us in their understanding,patience and constant encouragement.

Prince Kumar (Roll No. - 11192558)
Parth Goel (Roll No. - 11192619)
Akanksha Raj (Roll No.-11192663)
Abshishek Pundir(Roll No.-11192586)

TABLE OF CONTENT



Supervisor's Performa

Maharishi Markandeshwar Engineering College

Computer Science & Engineering Department

Project Work Progress Report Performa For B. Tech Students

Semester:		Section:
1	Name of Student	Prince Kumar
		Parth Goel
		Akanksha Raj
		Abhishek Pundir
2	Roll Number	11192558
		11192619
		11192663
		11192586
3	Project Title	Cartoonify An Image (opencv & python)
4	Period of the Report	1st Assessment Date: 25/03/2022
		2nd Assessment Date:
		Final Assessment Date:
5	Name of Supervisor(s)	Dr. Puneet Banga
6	40/45 % Progress in Percentage (%) 90+ %	40-50 % % (At 1st Assessment)
		90+ % % (At 2nd Assessment)
		% (At Final Assessment)
Supervisor Signature with Date: (1st Assessment) [illegible] 21/03/2022		
Supervisor Signature with Date: (2nd Assessment) [illegible] 22/4/22		
Supervisor Signature with Date: (Final Assessment)		

ABSTRACT

This paper 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. This paper presents an approach to cartoonize digital images into cartoon-like. The method used is different from as others used previously. This paper focuses on various techniques involved during the whole process, which, when used on layer by layer, gives an appropriately balanced output. We tend to explore different functions that can be integrated together in a particular pattern to get a filtered and composed output. The mathematical approach to different functions has also been explicated and the working is explained in detail. This system aims to use the filters' full functionality, which will help both in application and research, and can work as a framework to any computer vision-related systems and can be improved and embedded with other systems to work both as an independent module or as an integrated system.

Introduction

In this era of technology, cartoons do not need to be drawn manually. We have apps that can convert your images easily into cartoons. As we might know, sketching or creating a cartoon doesn't always need to be done manually. Nowadays, many apps can turn your photos into cartoons. But what if we tell you, that we can create our own effect with few lines of code? There is a library called OpenCV which provides a common infrastructure for computer vision applications and has optimized-machine-learning algorithms. It can be used to recognize objects, detect, and produce high-resolution images. To create a cartoon effect, we need to pay attention to two things; edge and color palette. 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. Those are what make the differences between a photo and a cartoon. 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. To adjust that two main components, there are four main steps that we will go through:

1. **Load image**

   The first main step is loading the image. Chose the image which we want to be transformed into a cartoon.

2. **Create edge mask**

   Commonly, a cartoon effect emphasizes the thickness of the edge in an image. We can detect the edge in an image by using the cv2 library.

3. **Reduce the color palette**

   The main difference between a photo and a drawing — in terms of color — is the number of distinct colors in each of them. A drawing has fewer colors than a photo. Therefore, we use color quantization to reduce the number of colors in the photo.

4. **Combine edge mask with the colored image**

   The final step is combining the edge mask that we created earlier, with the color - processed image. To do so, we use the cv2 bitwise function.

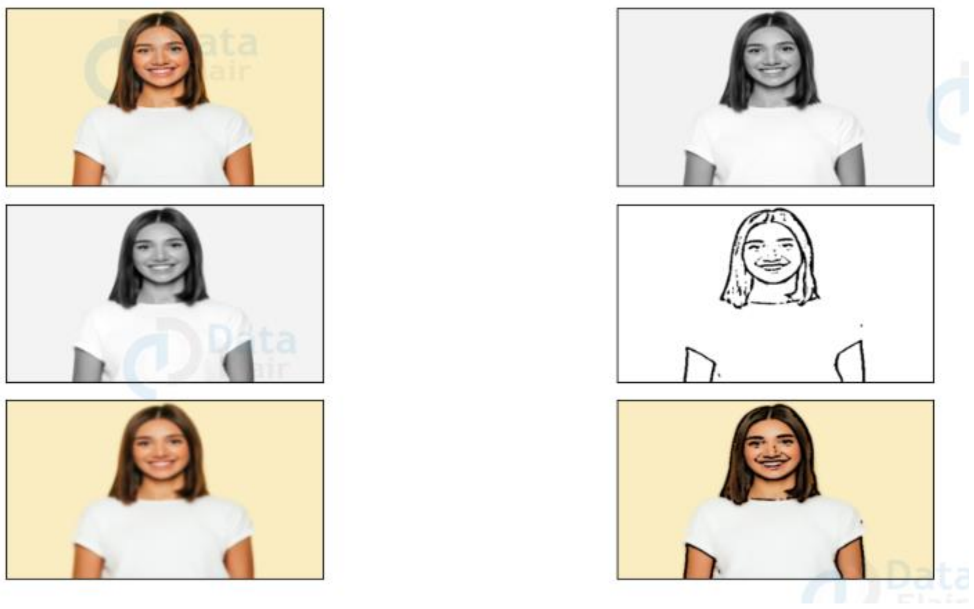


Figure 1: The output will look like the above image

1.1 Problem Statement

The cartoonization of an image depends on the type of method used in the system. Several ways are used for the same process. The most often use methods are– GAN (Generative Adversarial Networks), a machine learning framework that generated new data based on training data, and OpenCV library, which we are using on this system.

The current system works on OpenCV library and corresponding filters that are applied in the input image. Many filters can be used independently or can be added with others to create a custom filter. The most common and widely known filters are medianBlur(), GaussianBlur(), Laplacian(), BilateralFilter() and many more. These filters are best productive when used collectively with each other. Implementing a single filter with limited features will give output but certainly not an acceptable one.

In this system, there are certain combinations of filters used to carry out the cartoonization process.

1.2 Why the new system is to be developed?

The project, Image Cartoonization is creative, more flexible, and customizable in a sense. This web app provides users to make their personalized cartoonized images as per their choice based on different filters provided. Here, a user can upload or click a picture of his own and then manipulate various adjustments with given sliders before opting for the final image. Each filter has 2-3 sliders with clear instructions to have a clear understanding.

1.3 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.

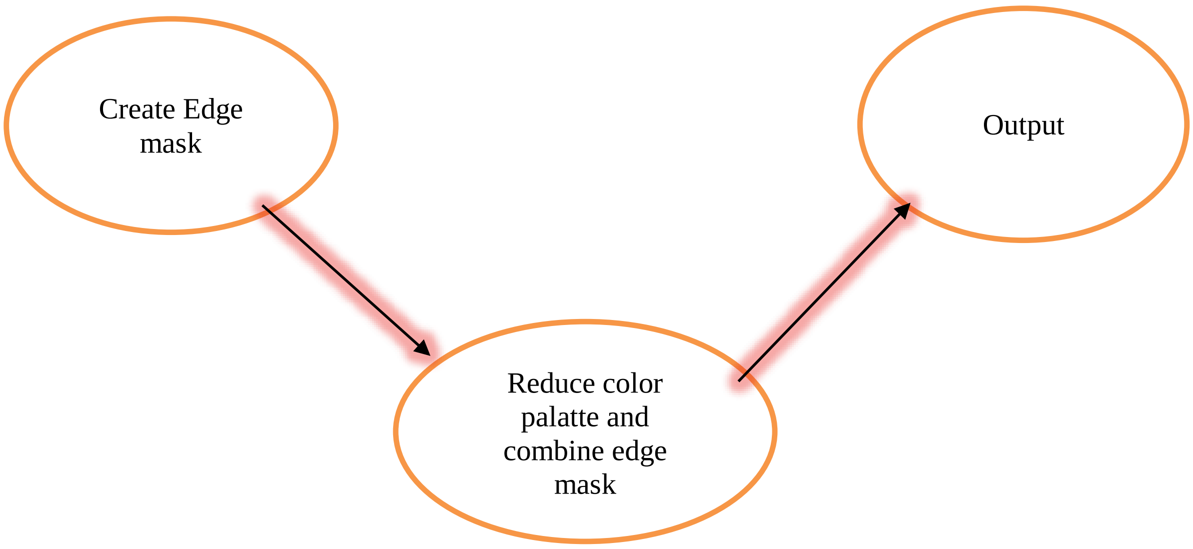


Figure 2: Steps for cartoonify an image

Following is our brief research on existing solutions:

A. Cartoon Effect The majority of photo editing websites offer the so called 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.

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.

1.4 Flow Chart

This phase includes two phases: The first phase is to compare the filteredsignal with the predefined dataset. The second phase is to get the recognized gesture in highest precision for better comparisons and error avoidance.

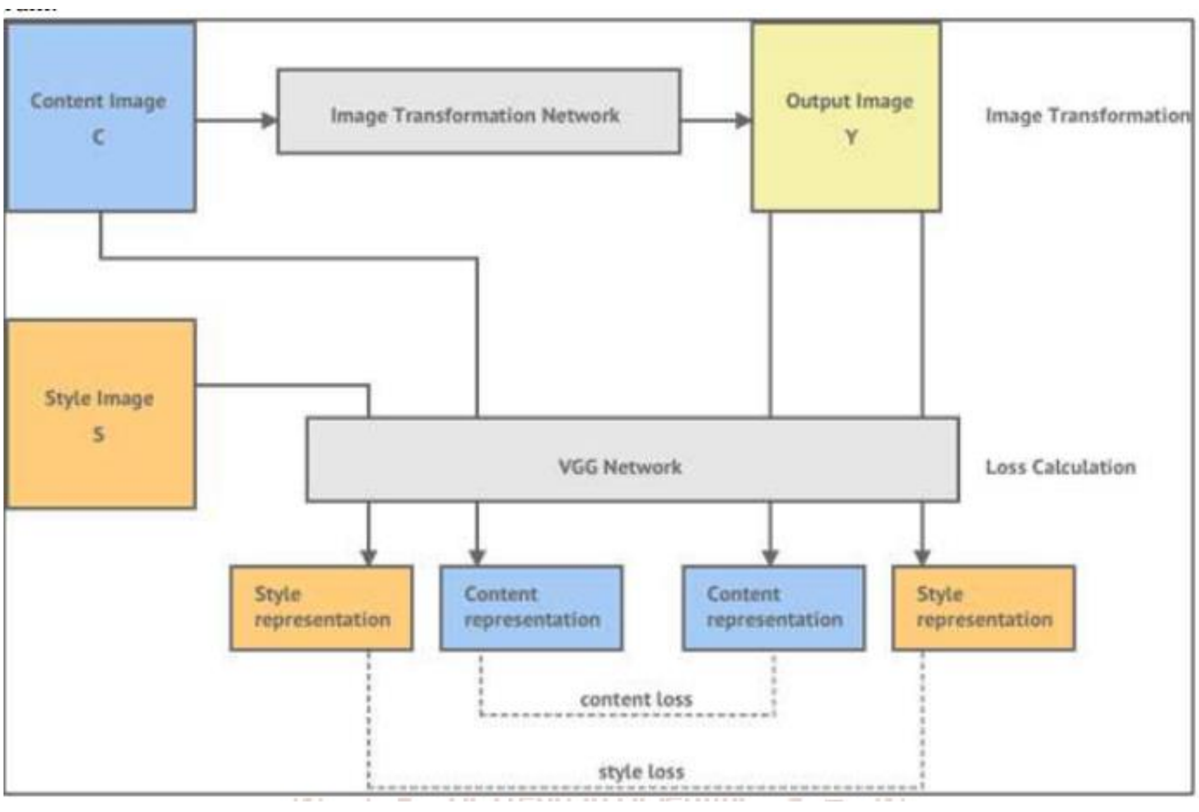


Figure 3: The above figures depicts the flowchart for cartoonifiny an image

1.5 Identification

The phase included two phases: The phase is to compared the filtered signal with the predefined Dataset. The second phase is to get the recognized gesturein highest precision for better comparison and error avoidance.

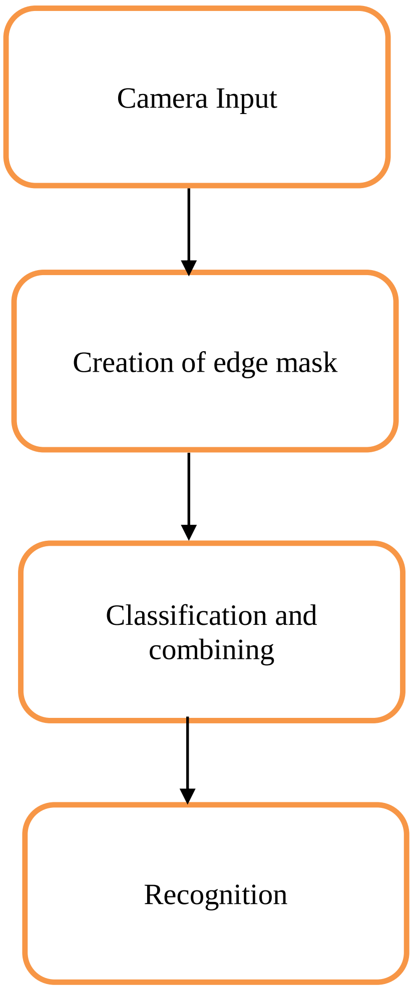


Figure 4: The above figure depicts the flow chart for the application

1.6 Methodology

The methodology basically involves two steps: Segmentation of the images based on color and Template based Feature Extraction. It revolves around the fact that Tintin (who is the character we are trying to identify), has a particular complexion and a marked feature as regards his hair. For the segmentation part, we have chosen John Smith's thesis. Regions having a specified color can be extracted by Histogram-Back-Projection. The thesis describes a variety of algorithms working on the basis of this method. Local histogramming achieves the best results. The process to produce the cartoon effect is divided into two branches- one for detecting and boldening the edges, and one for smoothing and quantizing the colors in the image. At the end, the resulting images are combined to achieve the effect.

System Description

USE OF CAMERA

The essential capacity of this framework is to perceive hand signal. Here,motion picture is taken utilizing the camera, the picture will be handled utilizing techniques for forms and in the wake of recognizing, the motionand a yield will be given on the showcase screen.

HARDWARE DETAILS

The experiment is performed on HP Pavilion having specifications:

- 4 Core CPU @ 2.30GHz
- RAM – 8GB
- Video Memory – 2 GBNVIDIA

PREPROCESSING STAGE

- Input Stage
- Processing Stage
- Output Stage

INPUT STAGE

- A high definition camera is used to capture images.
- The image captured must be as clear as possible to lower the occurrenceof error.
- The input image is transferred to the raspberry pi module for furtherprocessing.

PROCESSING STAGE

- It has the main role in gesture recognition.
- Major steps performed in this stage include:

OUTPUT STAGE

- The output is displayed on the screen.

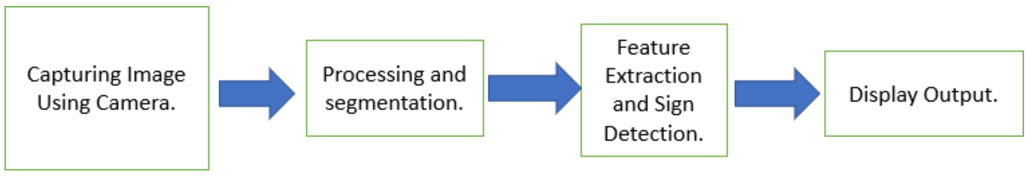

Capturing Image Using Camera.
Processing and segmentation.
Feature Extraction and Sign Detection.
Display Output.

Figure 5: Stages for Proposed Algorithm

<u>**Technology Used:-**</u>

Language:- PYTHON

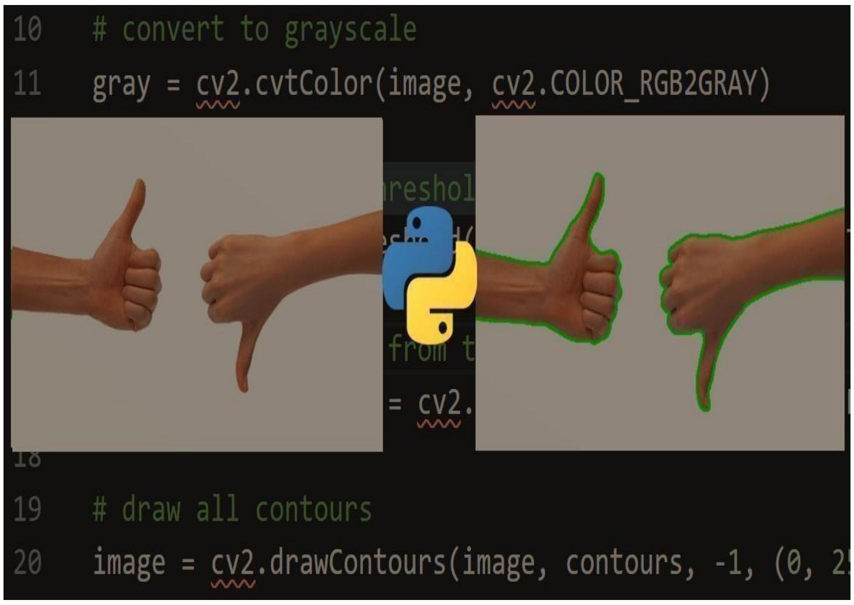
Library:- opencv

Module:- Numpy

LIBRARIES

OPEN CV

OpenCV is a Python library which is intended to tackle PC vision issues. It supports various languages and has been the most important library to deal with recognition type projects where frame or video needs to captured and dealt with whether it is object detection or motion detection.



```
# convert to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)

# draw all contours
image = cv2.drawContours(image, contours, -1, (0, 2
```

Figure 6 : Boundary drawn around the image.

OpenCV is a python library which is very helpful in calculating hand tracing and it is equipped with certain functions which are the steps towards creating a model having signal identification and those functions include video capturing, background removal, image editing like (resizing, rotation), plotting motion detection graph etc.

Figure 7(a): Recognition using cv2 and numpy

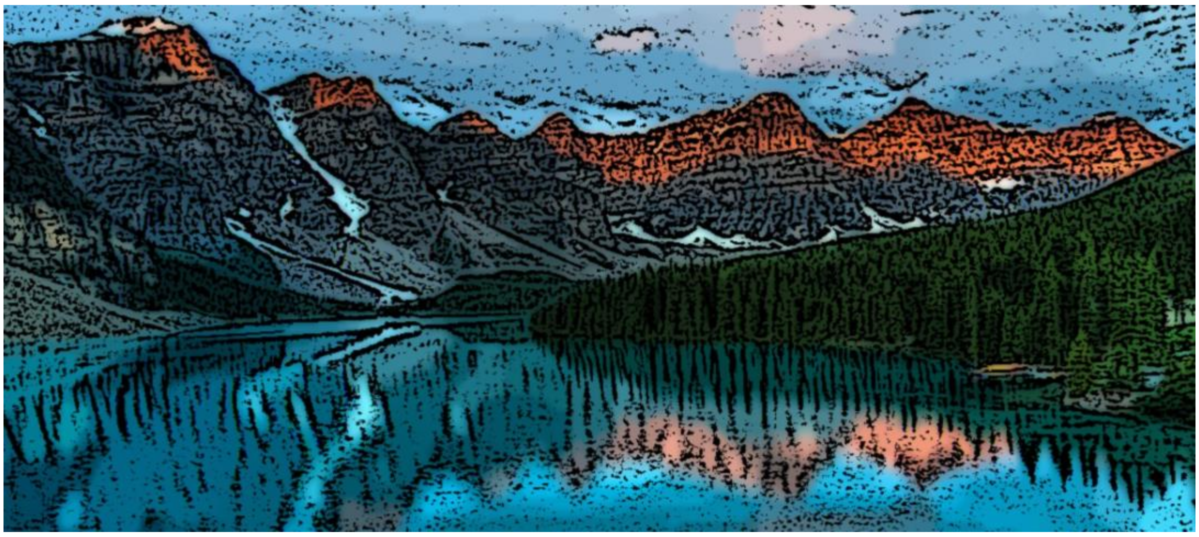


Figure 7(b): cv2 library are used

NumPy

NumPy is a Python module. The name NumPy represents :Numerical Python and it is utilized.It is an expansion module for Python, generally written in C. This guarantees the precompiled logical and numerical limits and functionalities of Numpy guarantee remarkable execution speed.

Numpy is mostly used for performing calculations using certain functions it provideslike multiply, divide, power etc. It is basically used for performing complex calculations with ease like dot product, summation etc. It is a very efficient module and makes the work easier.

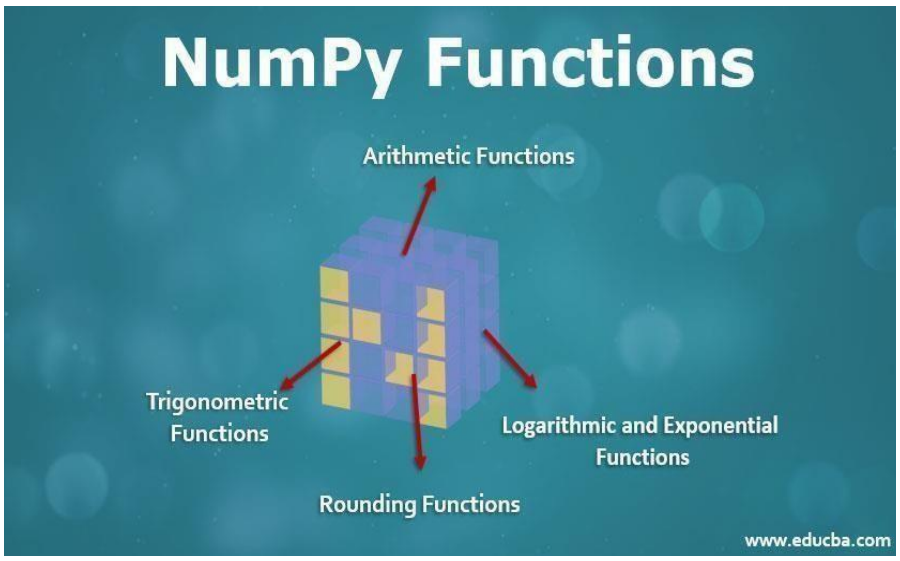
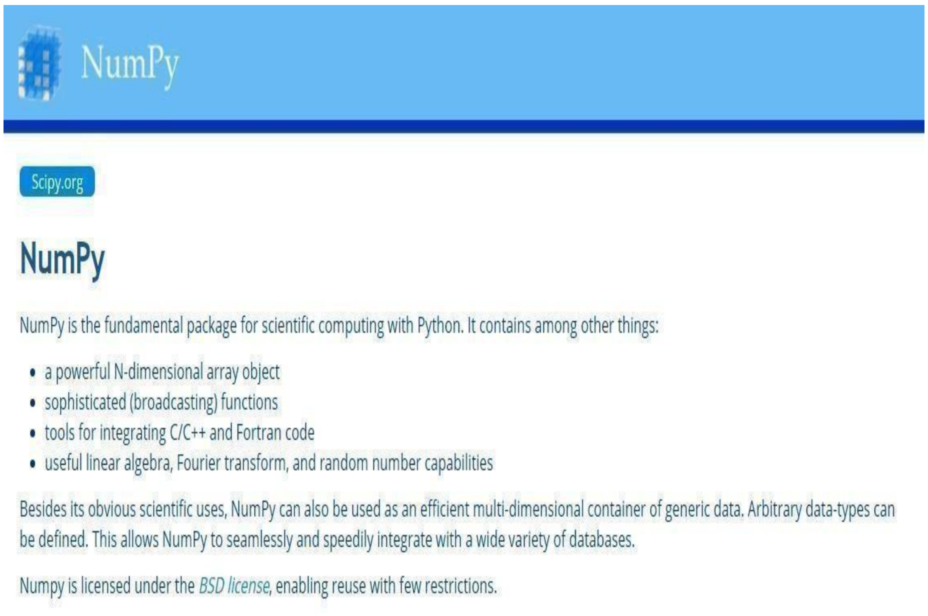


Figure 8 : NumPy Functions.


NumPy

Scipy.org

NumPy

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

Numpy is licensed under the *BSD license*, enabling reuse with few restrictions.

Figure 9: NumPy Description.

RESULT AND CONCLUSIONS

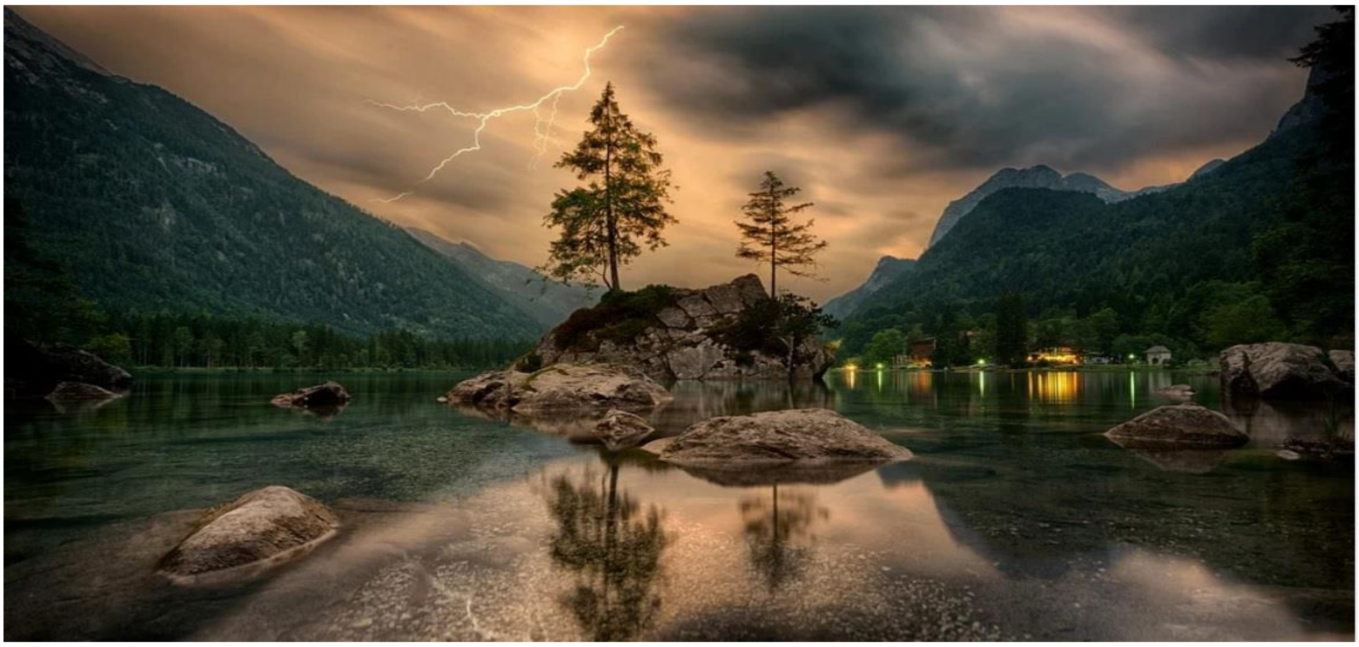


Figure 10(a): Original image

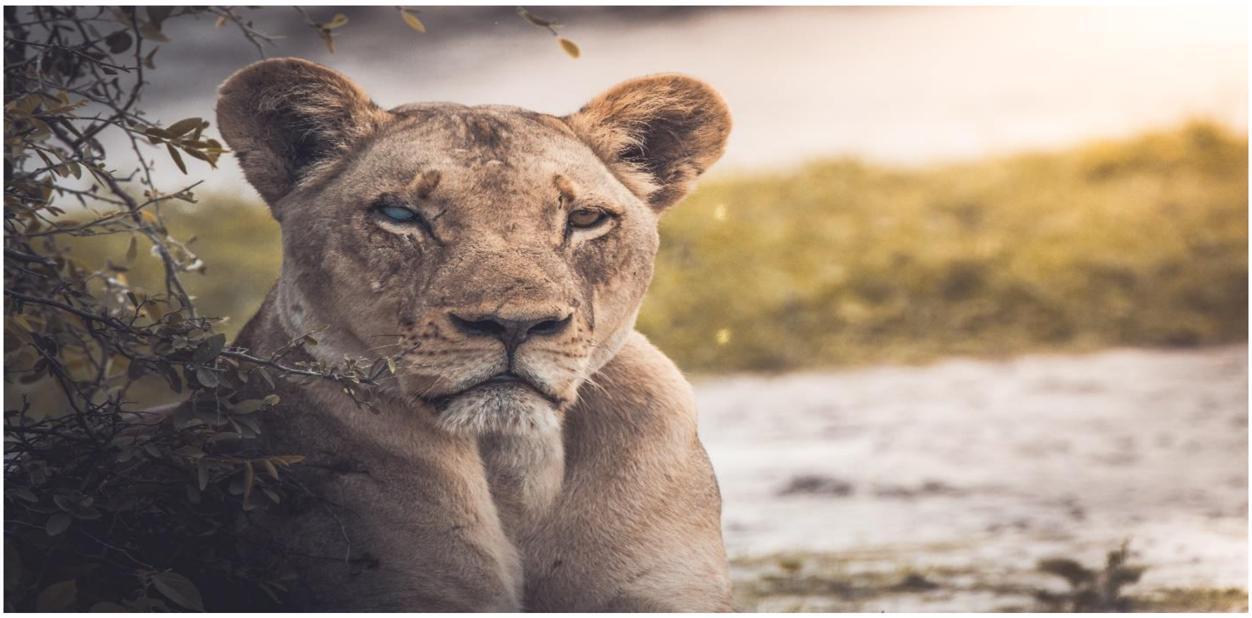


Figure 11(a): Original image

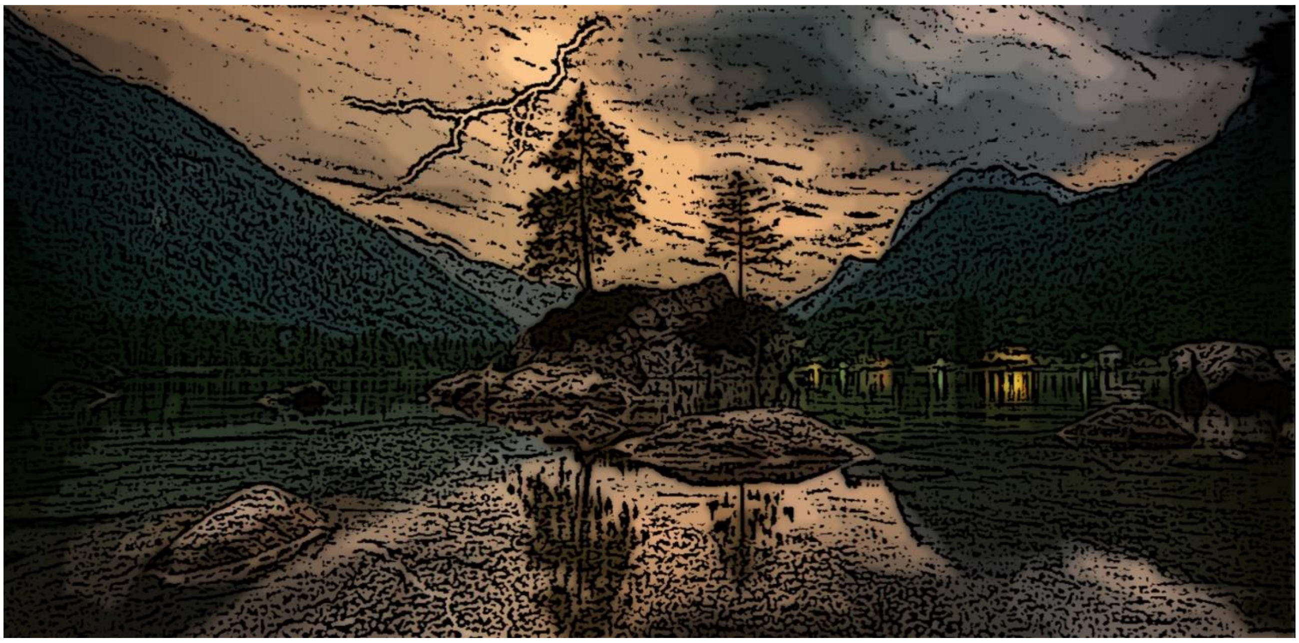


Figure 10(b): Cartoonified image

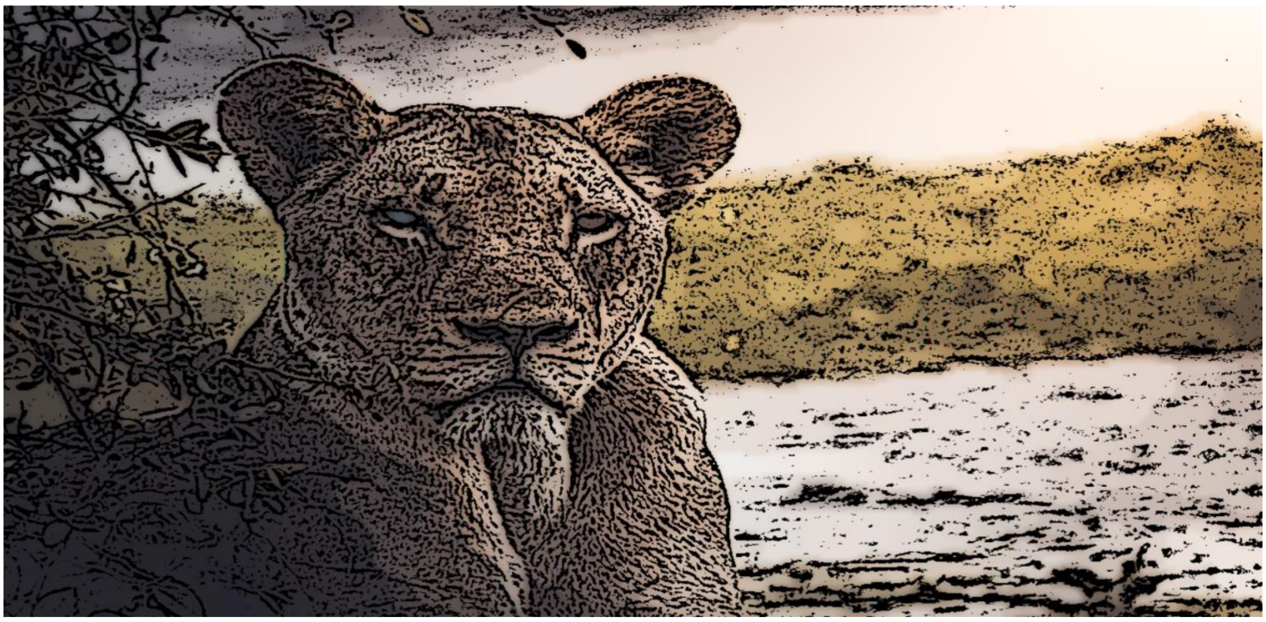


Figure 11(b): Cartoonified image

Objective of the project

The goal of this project is to create an application with a simple user interface allowing users to apply the cartoon algorithm to images of their choice.

The algorithm is designed to provide artistically and comically appealing results on as wide a range of pictures as possible, although it is conceded that not all inputs will yield equally satisfying results. Given the variety of input images, it would be unrealistic to expect a one-size-fits-all approach to produce consistent results.

The process to produce the cartoon effect is divided into two branches- one for detecting and **boldening** the edges, and one for smoothing and quantizing the colors in the image. At the end, the resulting images are combined to achieve the effect. Specific goals throughout the process were to achieve solid continuous contours while avoiding small line clutter in the image, and also to achieve large regions of homogenous color, with a reduced color palette.

Output images are evaluated on how well they meet these criteria. It is capable of producing exactly the effect specified above, and a wide range of input images will yield satisfactory results.

Conclusion

Thus we have shown that how image can be converted to cartoon. Hardware and software requirements of image to cartoon conversion are also shown in this paper. It is capable of producing exactly the effect specified above, and a wide range of input images will yield satisfactory results. The systematic working of image to cartoon conversion and respective algorithm and formulae is shown with neat diagram in this paper. Also we have stated challenges and problems one can face while cartoonifying the captured image. In this paper we have also discussed need and scope of cartoonifying the content image. This endeavor was expected to be a prototype to check the chance of seeing motion based correspondence. With this endeavor, we are able to make interactive memes and are able to cartoonify the image.

LIMITATION

As assessed over couple of presumption were made for the endeavor, at anyrate the framework has couple of obstructions not withstanding that, similarto its weakness to recognize and track hand if the foundation is in a general sense proportionate to skin shading, to see and trace insane conditions of lighting. In addition, the assertion compose requires client obstruction as decided already.

FUTURE SCOPE

User will be provided with a set of pretrained style images to choose from. Based on the chosen style and the content image provided by the user, the Resulting image with cartoon like effect is generated by the program. In this paper, we proposed an image cartoonization web app that transforms real-world photos into high-quality cartoon-style images. The animation industry is not going to stop now, and the 70 International Journal for Modern Trends in Science and Technology standards are getting higher by the day, so this system provides room for the addition of features and is easily adjustable to any other source code required for larger modules. Our system can easily be patched up with an automation system and will give expected results image less pixelated.

Working Code

```
from importlib.resources import path
import cv2
import numpy as np


# import uploadFile


import tkinter as tk
from tkinter import filedialog
from tkinter.filedialog import askopenfile
from PIL import Image, ImageTk
import matplotlib.pyplot as plt



# UPLOAD FILE
FILE_PATH="test.jpg"



my_w = tk.Tk()
my_w.geometry("600x350")  # Size of the window
my_w.title('Pick an image now')
my_font1=('times', 18, 'bold')
l1 = tk.Label(my_w,text='Now Pick a Photo',width=30,font=my_font1)
l1.grid(row=1,column=1)
b1 = tk.Button(my_w, text='Upload File', width=20,command = lambda:upload_file())
b1.grid(row=2,column=1)



def upload_file():
    global img

    f_types = [('Jpg Files', '*.jpg')]

    filename = filedialog.askopenfilename(filetypes=f_types)

    img = ImageTk.PhotoImage(file=filename)
    b2 =tk.Button(my_w,image=img) # using Button

    b2.grid(row=3,column=1)

    global FILE_PATH


    FILE_PATH=filename



my_w.mainloop()  # Keep the window open

class Cartoonizer:


        """Cartoonizer effect
                A class that applies a cartoon effect to an image.
                The class uses a bilateral filter and adaptive thresholding to create
                a cartoon effect.
        """


        #variable


        def _init_(self):

                pass


        def render(self, img_rgb):

img_rgb = cv2.imread(img_rgb)

img_rgb = cv2.resize(img_rgb, (1366,668))

                numDownSamples = 2          # number of downscaling steps

                numBilateralFilters = 50 # number of bilateral filtering steps


                # -- STEP 1 --

                # downsample image using Gaussian pyramid
```

```
                    img_color = img_rgb

for _ in range(numDownSamples):

                              img_color = cv2.pyrDown(img_color)

                    #cv2.imshow("downcolor",img_color)
                    #cv2.waitKey(0)
                    # repeatedly apply small bilateral filter instead of applying
                    # one large filter
                    for _ in range(numBilateralFilters):

                              img_color = cv2.bilateralFilter(img_color, 9, 9, 7)



                    #cv2.imshow("bilateral filter",img_color)

                    #cv2.waitKey(0)

                    # upsample image to original size

                    for _ in range(numDownSamples):

                              img_color = cv2.pyrUp(img_color)

                    #cv2.imshow("upscaling",img_color)

                    #cv2.waitKey(0)
```

# -- STEPS 2 and 3 --

```
                    # convert to grayscale and apply median blur
                    img_gray = cv2.cvtColor(img_rgb, cv2.COLOR_RGB2GRAY)
                    img_blur = cv2.medianBlur(img_gray, 3)
                    #cv2.imshow("grayscale+median blur",img_color)
                    #cv2.waitKey(0)

                    # -- STEP 4 --
                    # detect and enhance edges
                    img_edge = cv2.adaptiveThreshold(img_blur, 255,

          cv2.ADAPTIVE_THRESH_MEAN_C,

          cv2.THRESH_BINARY, 9, 2)
                    #cv2.imshow("edge",img_edge)
                    #cv2.waitKey(0)
```

```
        # -- STEP 5 --
        # convert back to color so that it can be bit-ANDed with color image
        (x,y,z) = img_color.shape
        img_edge = cv2.resize(img_edge,(y,x))
        img_edge = cv2.cvtColor(img_edge, cv2.COLOR_GRAY2RGB)
        cv2.imwrite("edge.png",img_edge)
        #cv2.imshow("step 5", img_edge)
        #cv2.waitKey(0)
        #img_edge = cv2.resize(img_edge,(i for i in img_color.shape[:2]))
        #print img_edge.shape, img_color.shape
        return cv2.bitwise_and(img_color, img_edge)

tmp_canvas = Cartoonizer()
file_name = FILE_PATH
res = tmp_canvas.render(file_name)

cv2.imwrite("Cartoon version.jpg", res)
cv2.imshow("Cartoon version", res)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

- **<u>license.md</u>**

MIT License

- **<u>README.md</u>**

```
# Cartoonify an image

Recognizing "Cartoonify an image" using OpenCV and Python.
```

Libraries needed

* cv2
* imutils
* numpy
* sklearn

Usage

1. `upload a file`
2. `Convert - image file`
3. `python recognize-image.py`

- **Upload.py**

```
FILE_PATH=None

my_w = tk.Tk()
my_w.geometry("600x350")  # Size of the window
my_w.title('Pick an image now')
my_font1=('times', 18, 'bold')
l1 = tk.Label(my_w,text='Now Pick a Photo',width=30,font=my_font1)
l1.grid(row=1,column=1)
b1 = tk.Button(my_w, text='Upload File', width=20,command = lambda:upload_file())
b1.grid(row=2,column=1)


def upload_file():
    global img
    f_types = [('Jpg Files', '*.jpg')]
    filename = filedialog.askopenfilename(filetypes=f_types)
    img = ImageTk.PhotoImage(file=filename)
    b2 =tk.Button(my_w,image=img) # using Button
    b2.grid(row=3,column=1)
    global FILE_PATH
    FILE_PATH=filename

my_w.mainloop()  # Keep the window open
```

- **Convert-image.py**

```
class Cartoonizer:
    """Cartoonizer effect
        A class that applies a cartoon effect to an image.
        The class uses a bilateral filter and adaptive thresholding to create
        a cartoon effect.
    """
    #variable
    def _init_(self):
        pass

    def render(self, img_rgb):
        img_rgb = cv2.imread(img_rgb)
        img_rgb = cv2.resize(img_rgb, (1366,668))
        numDownSamples = 2   # number of downscaling steps
        numBilateralFilters = 50 # number of bilateral filtering steps

        # -- STEP 1 --

        # downsample image using Gaussian pyramid
        img_color = img_rgb
        for _ in range(numDownSamples):
            img_color = cv2.pyrDown(img_color)

        #cv2.imshow("downcolor",img_color)
        #cv2.waitKey(0)
        # repeatedly apply small bilateral filter instead of applying
        # one large filter
        for _ in range(numBilateralFilters):
            img_color = cv2.bilateralFilter(img_color, 9, 9, 7)

        #cv2.imshow("bilateral filter",img_color)
        #cv2.waitKey(0)
        # upsample image to original size
        for _ in range(numDownSamples):
            img_color = cv2.pyrUp(img_color)
        #cv2.imshow("upscaling",img_color)
        #cv2.waitKey(0)

        # -- STEPS 2 and 3 --
        # convert to grayscale and apply median blur
        img_gray = cv2.cvtColor(img_rgb, cv2.COLOR_RGB2GRAY)
        img_blur = cv2.medianBlur(img_gray, 3)
        #cv2.imshow("grayscale+median blur",img_color)
        #cv2.waitKey(0)
```

```
  # -- STEP 4 --
     # detect and enhance edges
     img_edge = cv2.adaptiveThreshold(img_blur, 255,
                                    cv2.ADAPTIVE_THRESH_MEAN_C,
                                    cv2.THRESH_BINARY, 9, 2)
     #cv2.imshow("edge",img_edge)
     #cv2.waitKey(0)

     # -- STEP 5 --
     # convert back to color so that it can be bit-ANDed with color image
     (x,y,z) = img_color.shape

     img_edge = cv2.resize(img_edge,(y,x))
     img_edge = cv2.cvtColor(img_edge, cv2.COLOR_GRAY2RGB)
     cv2.imwrite("edge.png",img_edge)
     #cv2.imshow("step 5", img_edge)
     #cv2.waitKey(0)
     #img_edge = cv2.resize(img_edge,(i for i in img_color.shape[:2]))
     #print img_edge.shape, img_color.shape
     return cv2.bitwise_and(img_color, img_edge)


fmp_canvas = Cartoonizer()
file_name = FILE_PATH


     --
```

```
res = tmp_canvas.render(file_name)


cv2.imwrite("Cartoon version.jpg", res)

cv2.imshow("Cartoon version", res)

cv2.waitKey(0)

cv2.destroyAllWindows()
```