Project Report

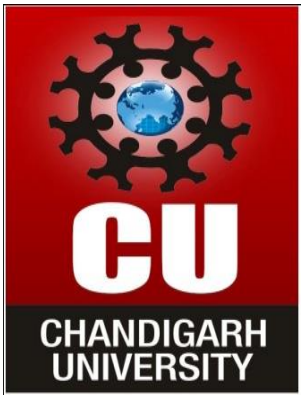
On

Image Cartoonifier

Submitted for the requirement of

Project course

BACHELOR OF ENGINEERING
COMPUTER SCIENCE & ENGINEERING



Submitted to:
Dheresh Soni

Submitted By:
Shubham Bhardwaj
18BCS6785

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
CHANDIGARH UNIVERSITY, GHARUAN

November 2021

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.

INTRODUCTION

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.

Software and Hardware Requirement

Hardware Requirement

Processor: i5-processor (Minimum)

RAM: 16GB

Graphic Card: Nvidia or AMD Graphic Card with 2GB(minimum)

Software Requirement

Operating System: Windows 7/Windows 8 or 8.1/Windows 10

Python 3.8

Jupyter Notebook

WORKING/ALGORITHM

The process to create a cartoon effect image can be initially branched into 2 divisions –

1) To detect, blur and bold the edges of the actual RGB color image.

2) To smooth, quantize and the conversion of the RGB image to grayscale.

The results involved in combining the image and help achieve the desired result.

1.1 Identifying Edges

Finding smooth outline that represents or bounds the shape of the image is an important property to achieve a quality image. All Edge processing tasks are:

- MEDIAN FILTER – This filter helps in reducing the noise created during the downscaling the image and later converting the original image to cartoon image by applying the bilateral filter. Any extreme specks are smoothened over.
- EDGE DETECTION – At first the noise of the image is removed within the image .Later the smoothened image is filtered using horizontal and vertical direction by dividing the cells of the picture element(both x and y dimensions.)
- MORPHOLOGICAL OPERATIONS – This serves the purpose to Bolden and smoothen the outline of the edges variably. The pixels that are highlighted but seems far are removed. Hence the edge lines reduce to thinner outline.
- EDGE FILTERING – Two divisions of the constituent regions, any region that pertain below a a certain threshold is removed. Small outline identified by the detection method is removed from the final image.

1.2 Color to RGB Image

The most important aspect is to eliminate the color regions and apply cartoon effects. Through this algorithm, the colors are smoothened on multiple filtrations so as to create a equal color regions.

- BILATERAL FILTERING – The important role of this filter is to smooth the images without creating any sort of noise also while preserving the edges. Filtering is performed by reading an image from the file and storing it in a matrix object. Initially creating an empty matrix to store the result and applying bilateral filter. This totally depends on the kernel size and testing by running more no of iterations.
- QUANTIZE COLOURS – The last step of the conversion involves the step of reducing the number of colors in each pixel.

1.3 Recombine

The final task is to overlay the edges onto the color image is when both the color and edge image processing are complete.

IMPLEMENTATION

- **CODE**

```
import cv2

import easygui

import numpy as np

import imageio

import sys

import matplotlib.pyplot as plt

import os

import uuid

import time

import tkinter as tk

from tkinter import filedialog

from tkinter import *

import tkinter.messagebox

from PIL import ImageTk, Image


top=tk.Tk()

top.geometry('500x500')

top.title('Image Cartoonifier!')

top.configure(background='yellow')
```

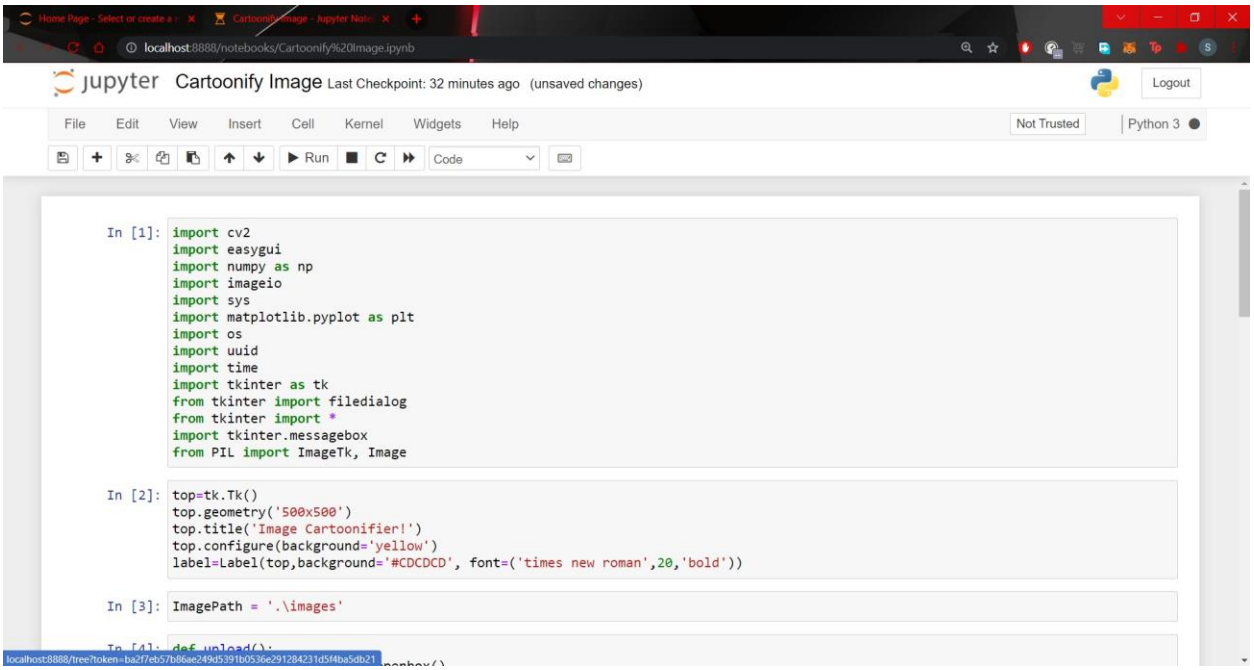
```
label=Label(top,background='#CDCDCD', font=('times new
roman',20,'bold'))
ImagePath = '.\images'

def upload():
    ImagePath=easygui.fileopenbox()
    cartoonify(ImagePath)

def new_image():
    cap = cv2.VideoCapture(0)
    time.sleep(5)
    for imgnum in range(1):
        ret, frame = cap.read()
        newName="Live"
        path1 = os.path.dirname(ImagePath)
        extension=os.path.splitext(ImagePath)[1]
        path = os.path.join(path1, newName+extension)
        cv2.imwrite(newName+extension, frame)
        cv2.imshow('frame', frame)
        time.sleep(2)
        if cv2.waitKey(1) & 0xFF == ord('q'):
```

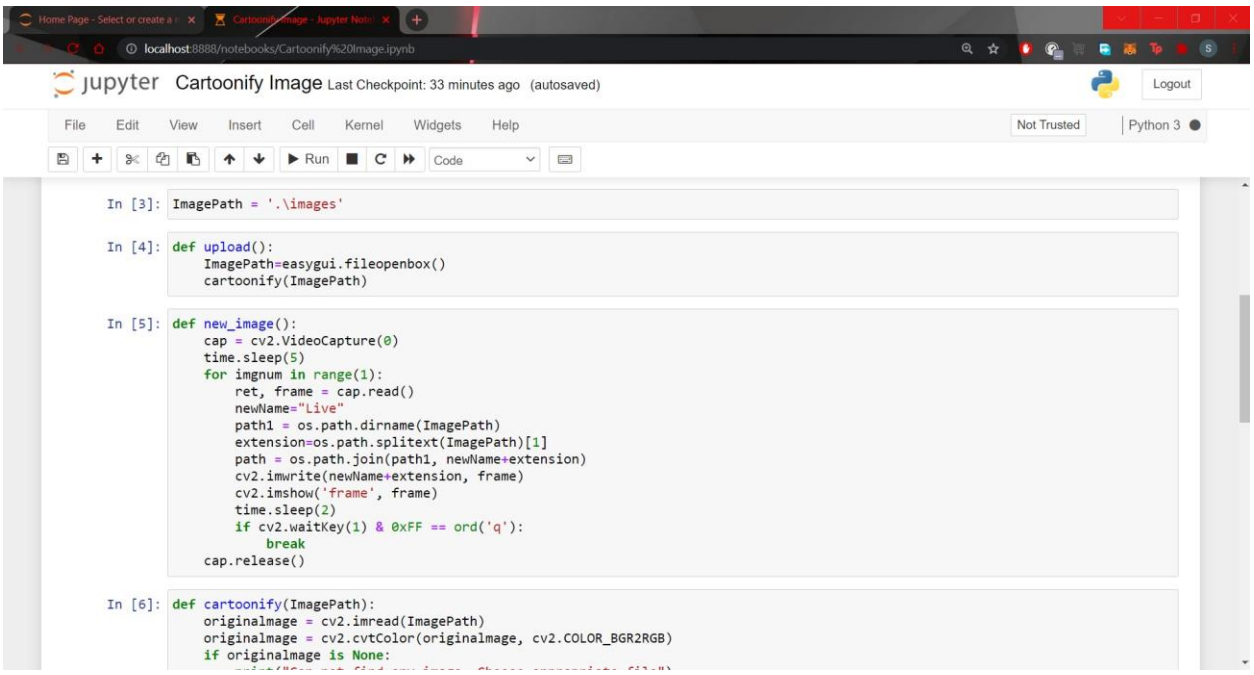
```
        break
    cap.release()


def cartoonify(ImagePath):
    originalmage = cv2.imread(ImagePath)
    originalmage = cv2.cvtColor(originalmage, cv2.COLOR_BGR2RGB)
    if originalmage is None:
        print("Can not find any image. Choose appropriate file")
        sys.exit()
    ReSized1 = cv2.resize(originalmage, (960, 540))
    grayScaleImage= cv2.cvtColor(originalmage, cv2.COLOR_BGR2GRAY)
    ReSized2 = cv2.resize(grayScaleImage, (960, 540))
    smoothGrayScale = cv2.medianBlur(grayScaleImage, 5)
    ReSized3 = cv2.resize(smoothGrayScale, (960, 540))
    getEdge = cv2.adaptiveThreshold(smoothGrayScale, 255,
        cv2.ADAPTIVE_THRESH_MEAN_C,
        cv2.THRESH_BINARY, 9, 9)
    ReSized4 = cv2.resize(getEdge, (960, 540))
    colorImage = cv2.bilateralFilter(originalmage, 9, 300, 300)
    ReSized5 = cv2.resize(colorImage, (960, 540))
    cartoonImage = cv2.bitwise_and(colorImage, colorImage, mask=getEdge)
```

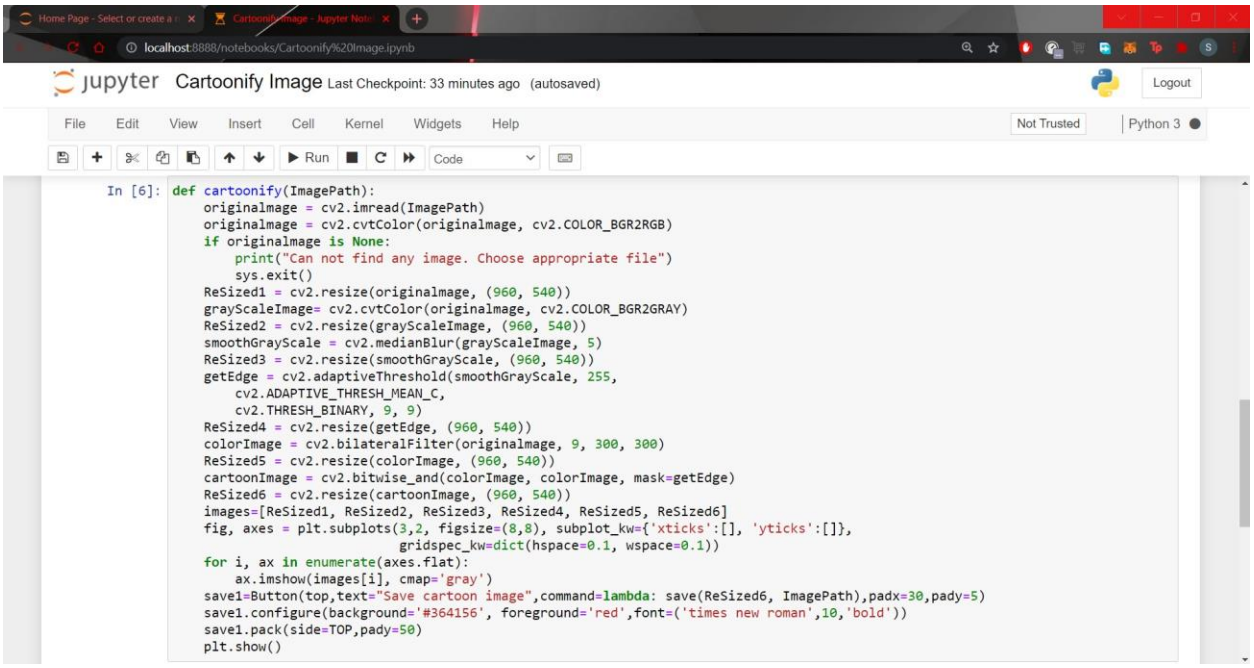
```
    ReSized6 = cv2.resize(cartoonImage, (960, 540))
    images=[ReSized1, ReSized2, ReSized3, ReSized4, ReSized5, ReSized6]
    fig, axes = plt.subplots(3,2, figsize=(8,8), subplot_kw={'xticks':[], 'yticks':[]},
                             gridspec_kw=dict(hspace=0.1, wspace=0.1))
    for i, ax 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='red',font=('times new
roman',10,'bold'))
    save1.pack(side=TOP,pady=50)
    plt.show()

def save(ReSized6, ImagePath):
    newName="NewImage"
    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)
```

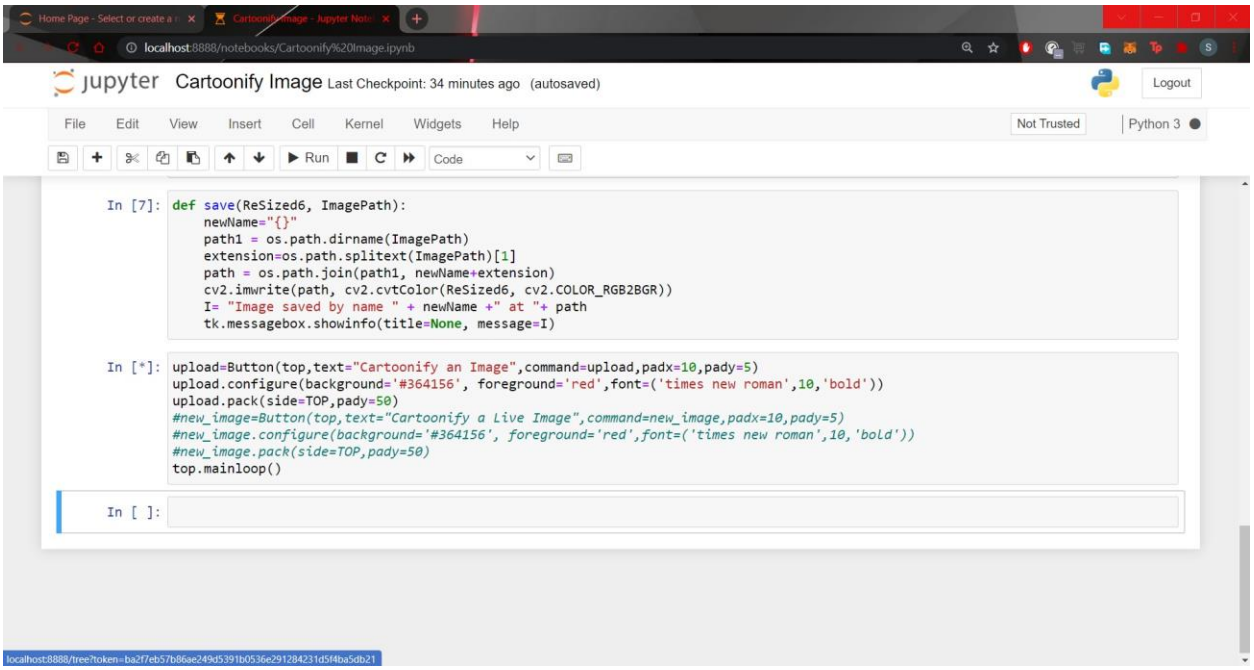
```
upload=Button(top,text="Cartoonify an
Image",command=upload,padx=10,pady=5)
upload.configure(background='#364156', foreground='red',font=('times new
roman',10,'bold'))
upload.pack(side=TOP,pady=50)
#new_image=Button(top,text="Cartoonify a Live
Image",command=new_image,padx=10,pady=5)
#new_image.configure(background='#364156', foreground='red',font=('times
new roman',10,'bold'))
#new_image.pack(side=TOP,pady=50)
top.mainloop()
```

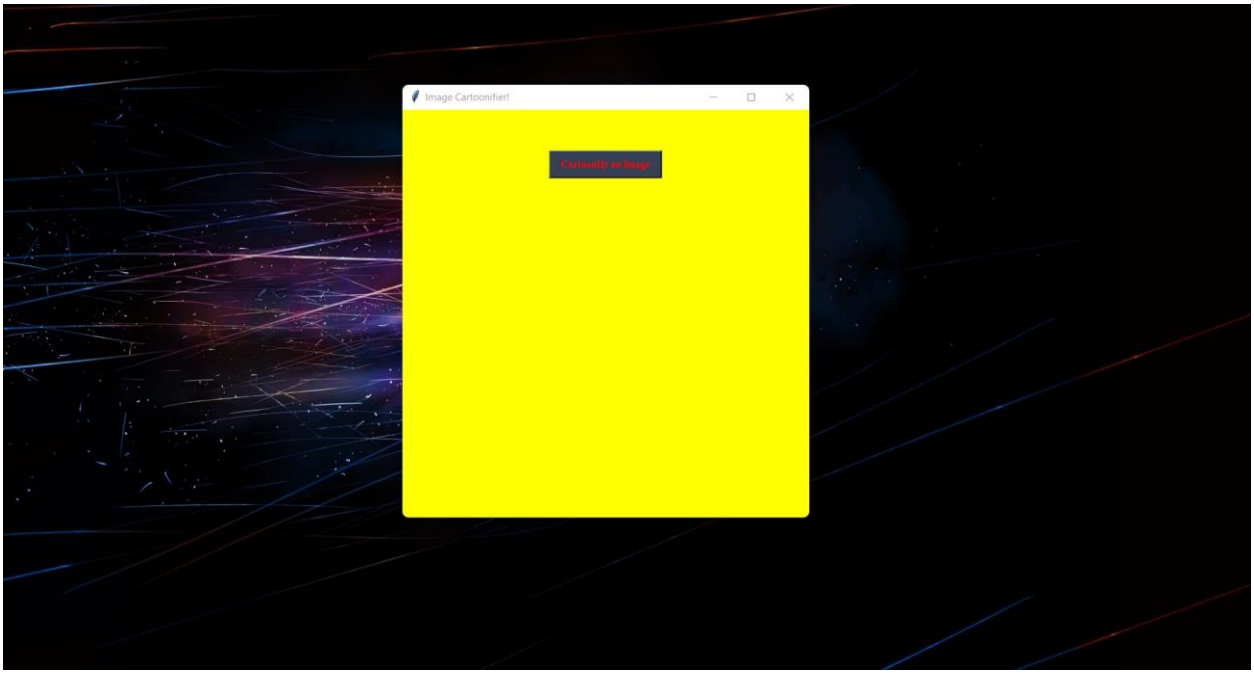
- **SCREENSHOTS**



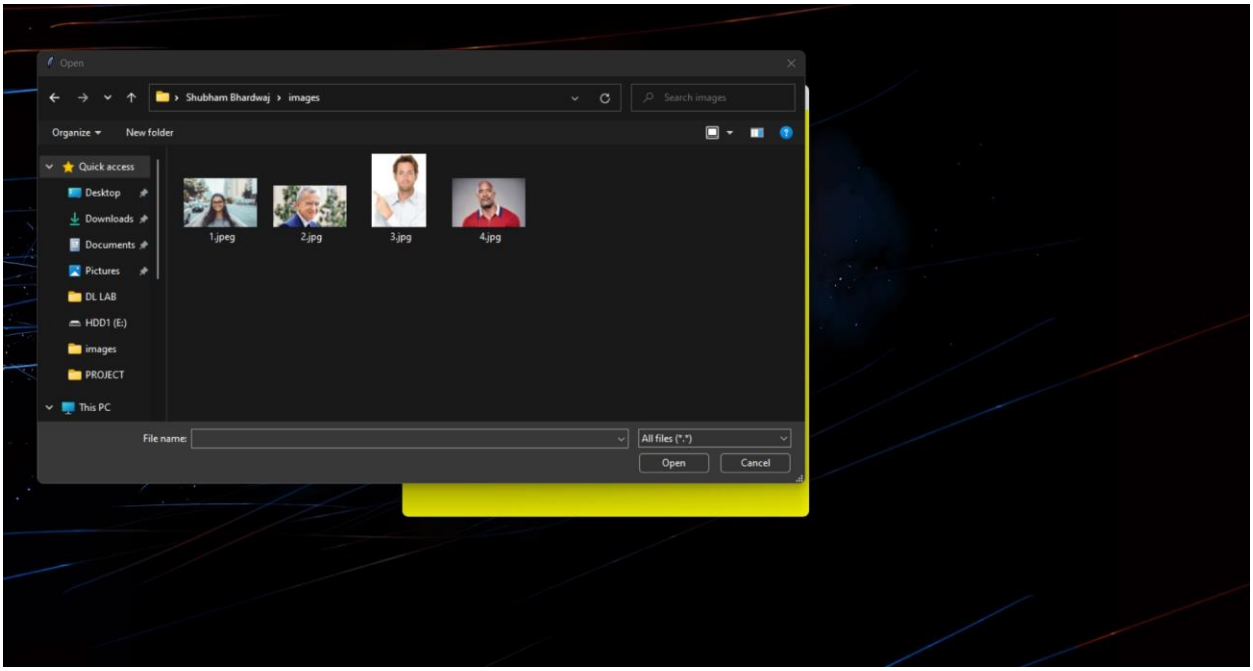



Home Page - Select or create a n
Cartoonify Image - Jupyter Note
localhost:8888/notebooks/Cartoonify%20Image.ipynb
jupyter Cartoonify Image Last Checkpoint: 33 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
In [6]:
def cartoonify(ImagePath):
originalmage = cv2.imread(ImagePath)
originalmage = cv2.cvtColor(originalmage, cv2.COLOR_BGR2RGB)
if originalmage is None:
print("Can not find any image. Choose appropriate file")
sys.exit()
ReSized1 = cv2.resize(originalmage, (960, 540))
grayScaleImage= cv2.cvtColor(originalmage, cv2.COLOR_BGR2GRAY)
ReSized2 = cv2.resize(grayScaleImage, (960, 540))
smoothGrayScale = cv2.medianBlur(grayScaleImage, 5)
ReSized3 = cv2.resize(smoothGrayScale, (960, 540))
getEdge = cv2.adaptiveThreshold(smoothGrayScale, 255,
cv2.ADAPTIVE_THRESH_MEAN_C,
cv2.THRESH_BINARY, 9, 9)
ReSized4 = cv2.resize(getEdge, (960, 540))
colorImage = cv2.bilateralFilter(originalmage, 9, 300, 300)
ReSized5 = cv2.resize(colorImage, (960, 540))
cartoonImage = cv2.bitwise_and(colorImage, colorImage, mask=getEdge)
ReSized6 = cv2.resize(cartoonImage, (960, 540))
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='red',font=('times new roman',10,'bold'))
save1.pack(side=TOP,pady=50)
plt.show()

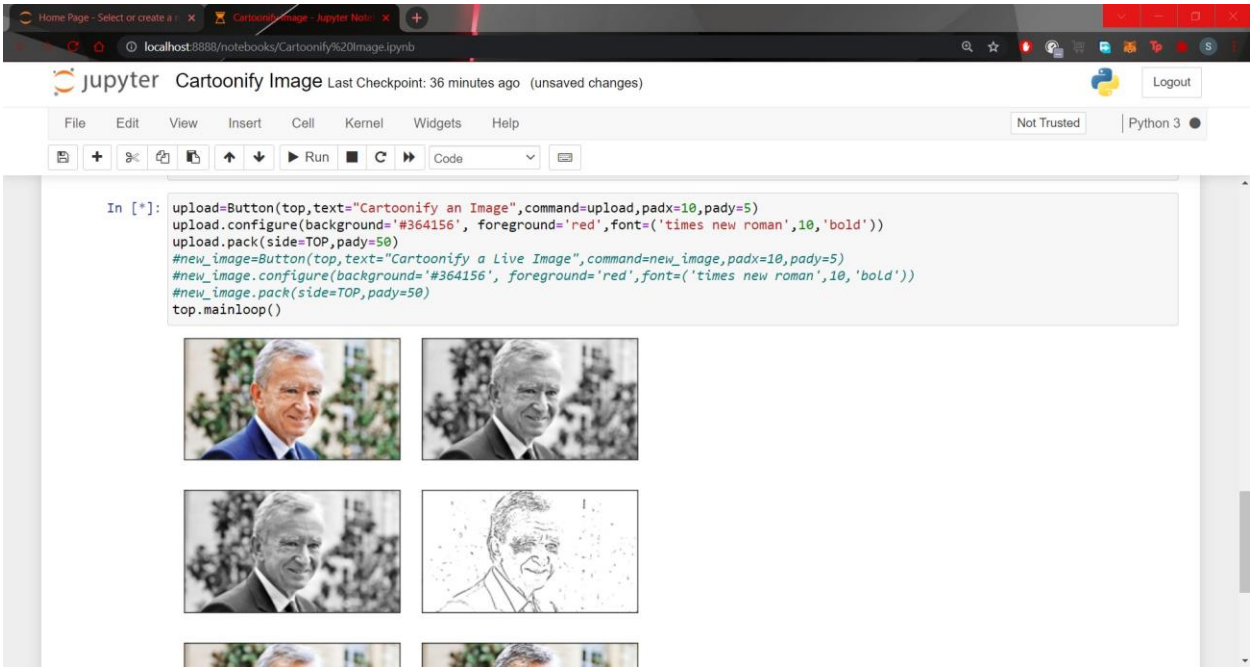

Home Page - Select or create a n
Cartoonify Image - Jupyter Note
localhost:8888/notebooks/Cartoonify%20Image.ipynb
jupyter Cartoonify Image Last Checkpoint: 34 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
In [7]:
def save(ReSized6, ImagePath):
newName="{}"
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)
In [*]:
upload=Button(top,text="Cartoonify an Image",command=upload,padx=10,pady=5)
upload.configure(background='#364156', foreground='red',font=('times new roman',10,'bold'))
upload.pack(side=TOP,pady=50)
#new_image=Button(top,text="Cartoonify a Live Image",command=new_image,padx=10,pady=5)
#new_image.configure(background='#364156', foreground='red',font=('times new roman',10,'bold'))
#new_image.pack(side=TOP,pady=50)
top.mainloop()
In [ ]:
localhost:8888/tree?token=ba2f7eb57b86ae249d5391b0536e291284231d5f4ba5db21

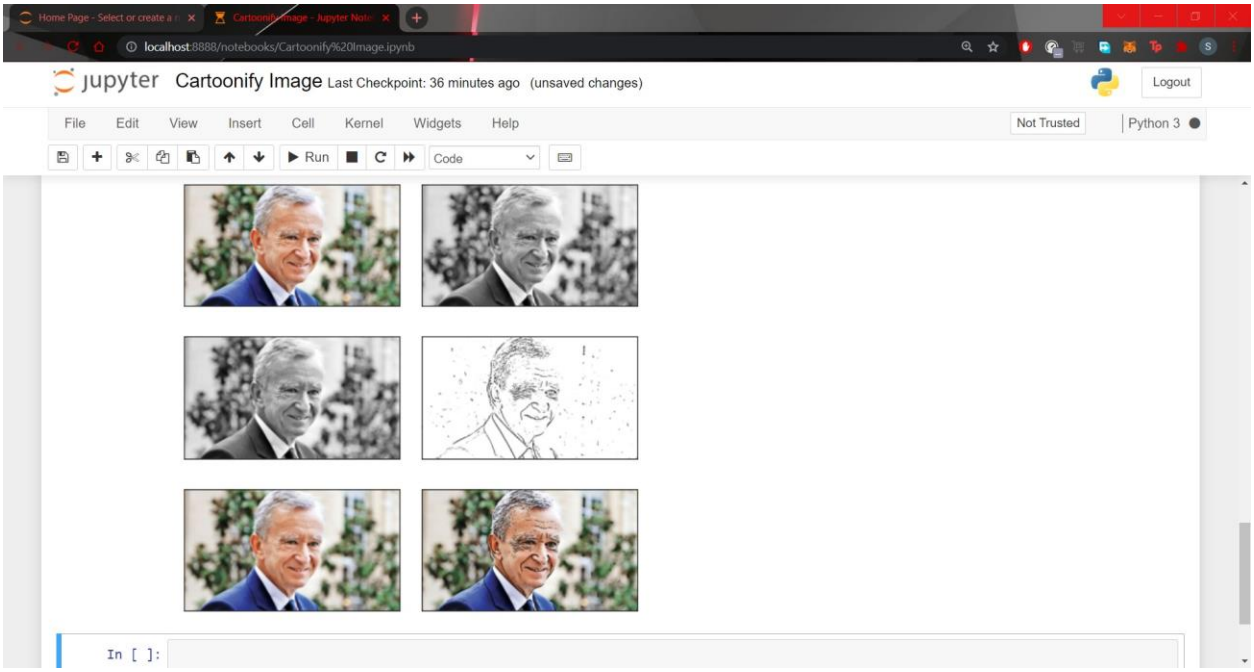


Click on button “Cartoonify an Image” to upload the required image.

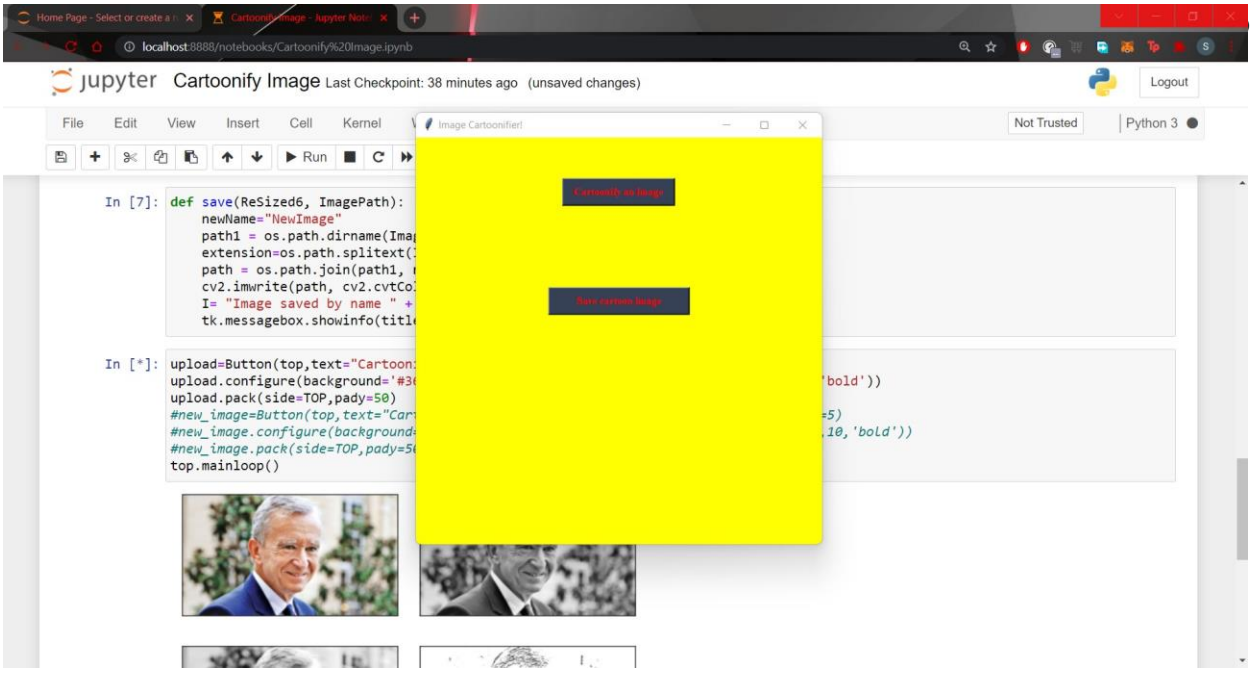


Select the image to be modified.

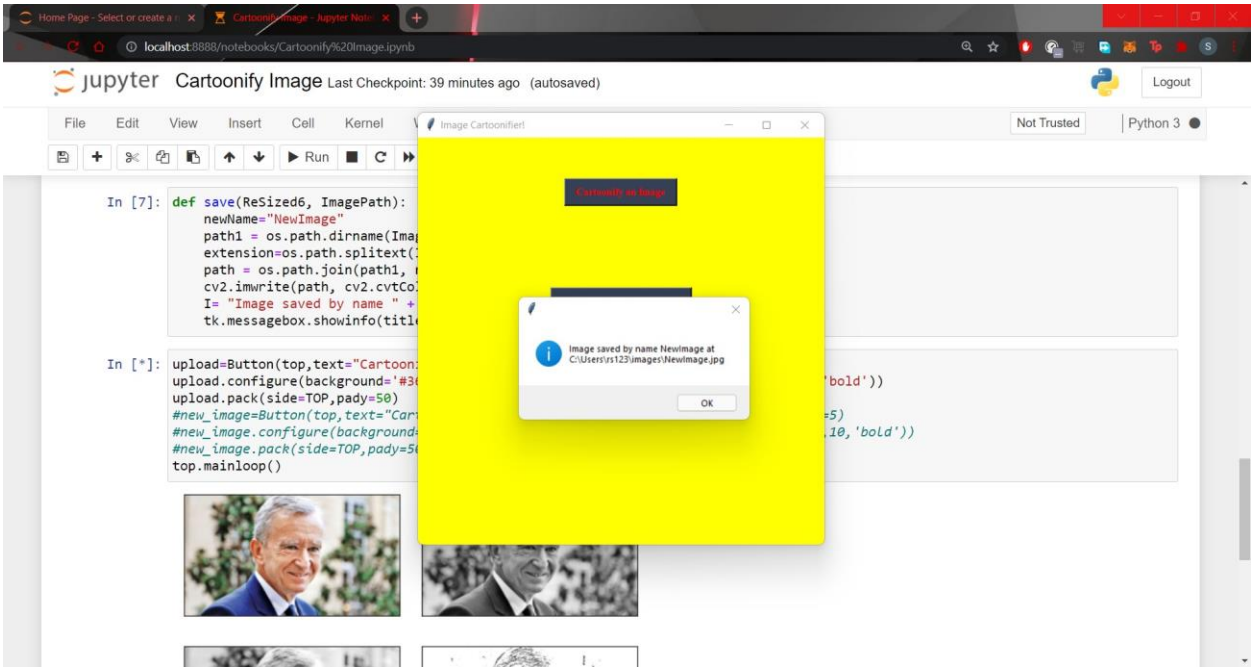


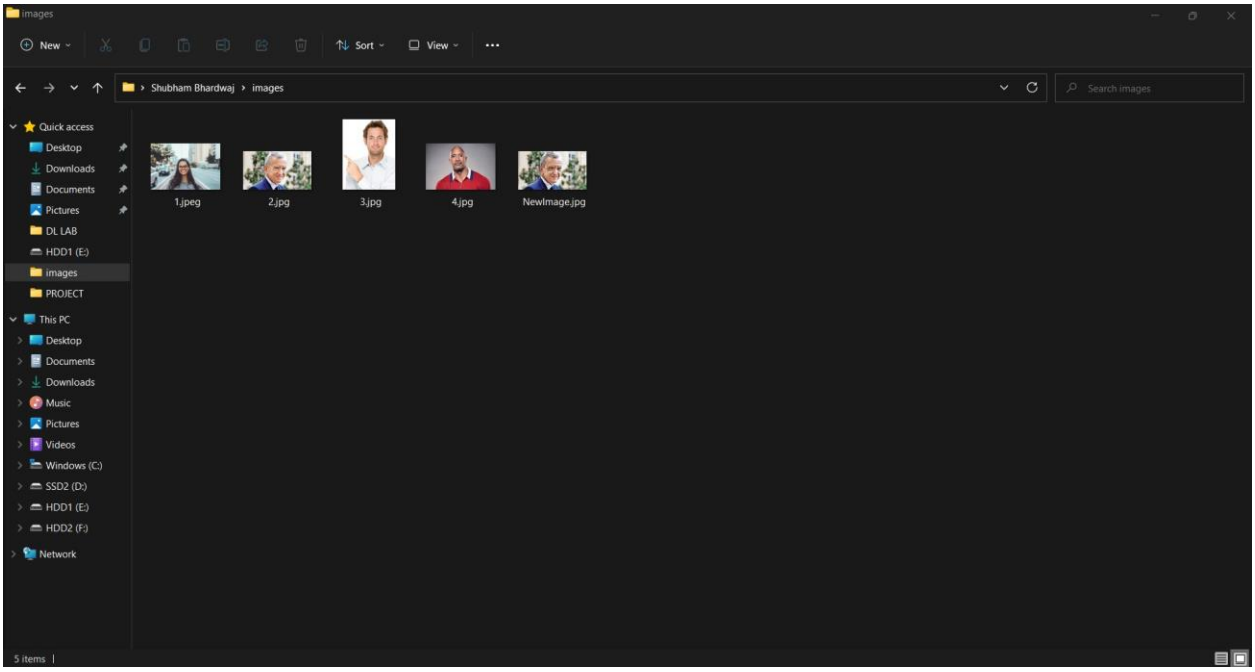


The selected image is uploaded in the program and is modified to look like a cartoon by highlighting the edges of the face.



Click on button “Save cartoon image” to save the modified image and message will appear giving location of image saved and name of image.





Before modifications to the image.

Image after modifications to it.

CONCLUSION

First of all, the basic tools to handle the titled problems of the thesis are incorporated. It includes origin and history of image processing, different types of uncertain environment, existing methods for cartoon imaging. Amid the previous three decades, the topic of image processing has gained vital name and recognition among researchers because of their frequent look in varied and widespread applications within the field of various branches of science and engineering. As an example, image processing is helpful to issues in signature recognition, digital video processing, Remote Sensing and finance. Conclusion and Future Directions

Firstly, we use high-resolution camera to take picture of the internal structure of the wire. Secondly, we use OpenCV image processing functions to implement image pre-processing. Thirdly we use morphological opening and closing operations to segment image because of their blur image edges. The main attraction of the paper is to solve different types of images having one object, two object and three object which can't be solved by any of the exiting methods but can be solved by our proposed method.

Bibliography

- https://www.wikipedia.org/
- https://github.com/
- https://www.youtube.com/
- https://opencv.org/