

**SCHOOL OF COMPUTING**

**DEPARTMENT OF NETWORKING AND COMMUNICATION**

**Sketching and Painting an Image**

**18CSC305J**

**Artificial Intelligence**

**Team Member :**

**1. RA1911030010085 - Ankit Yadav**

**2. RA1911030010094 - Rahul Goel**

**3. RA1911030010119 - Rushil Saxena**

**Problem Statement:**

While editing photos, we usually apply filters to edit the photos directly and get a sketch representation or a pastel paint representation of the image.

In this project we design a program which discovers the working behind such existing filters and shows the intermediate steps as well before giving the sketch form and pastel paint form.

**Description:**

Since our project involves image processing we have used OpenCV and Adaptive Thresholding for its development. Rather than making it as a web application we are making an open source software which runs without the requirement of internet and the source code can be modified by user based on his requirements.

So we first take the image as an input, and break it into smaller parts using adaptive thresholding and assign a number to each part.

Then downscale the image and apply a bilateral filter to get a cartoon like effect.

We convert the image to grayscale and then we apply the media blur filter in order to get a blurred effect on the photo. This will help us in making the sketch.

Next step is to identify the edges in the image, which is basically used to get the sketch form of the image.

Then add this to the previously modified images to get a sketch effect. We use Adaptive Threshold for to do this step. Adaptive thresholding breaks the image into small regions and assigns a number to each region.

The painted image is an advanced form of the sketch image obtained.

We compile the final images obtained from the previous steps to get the final output as a painted image.

**Source 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('Sketch & Paint Your Image!')

top.configure(background='white')

label=Label(top,background='#CDCDCD', font=('calibri',20,'bold'))

def upload():

ImagePath=easygui.fileopenbox()

paint(ImagePath)

def paint(ImagePath):

# read the image(input)

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))

#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))

#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

paintImage = cv2.bitwise\_and(colorImage, colorImage, mask=getEdge)

ReSized6 = cv2.resize(paintImage, (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 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="Animmated\_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)

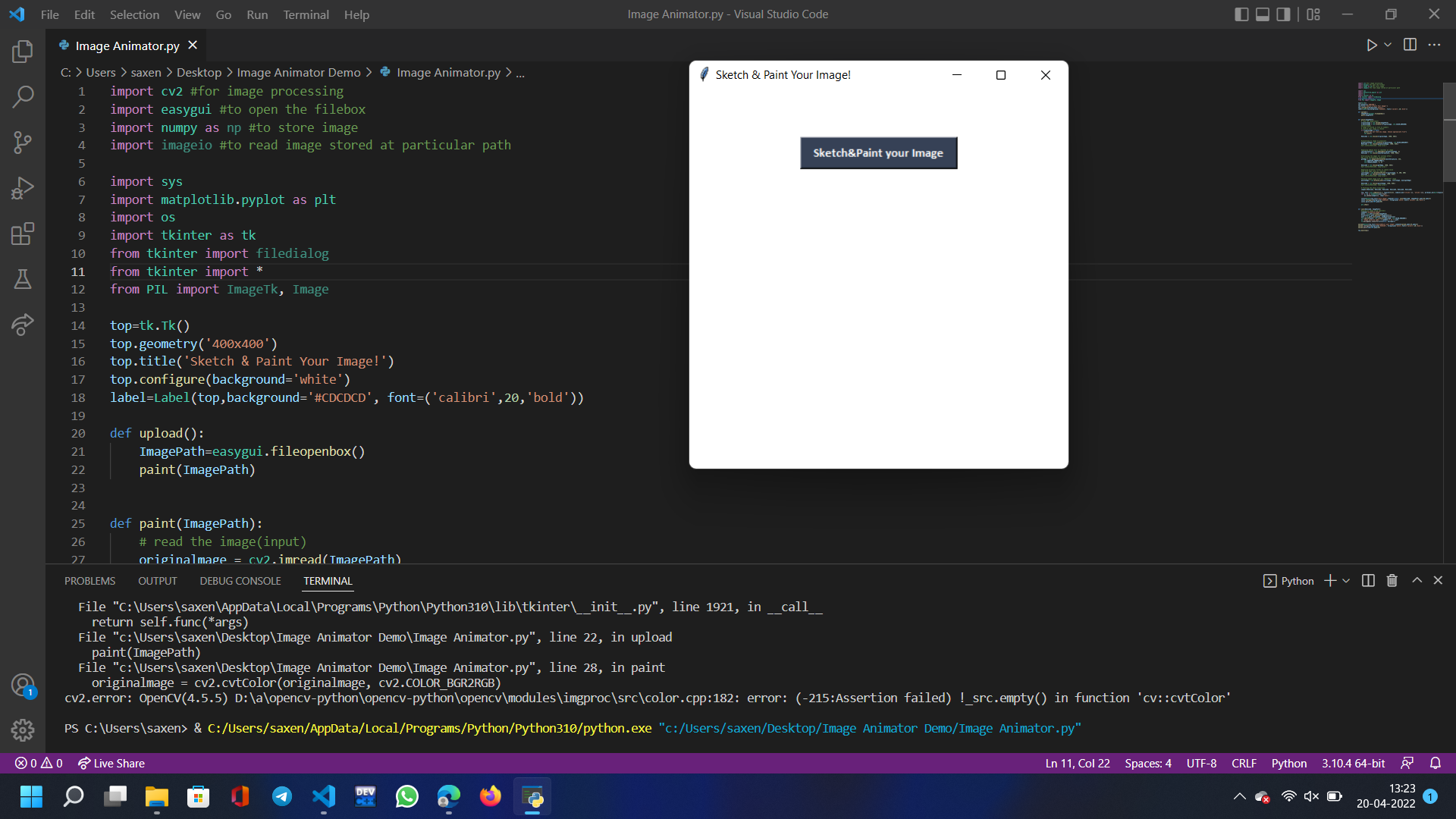
upload=Button(top,text="Sketch&Paint your Image",command=upload,padx=10,pady=5)

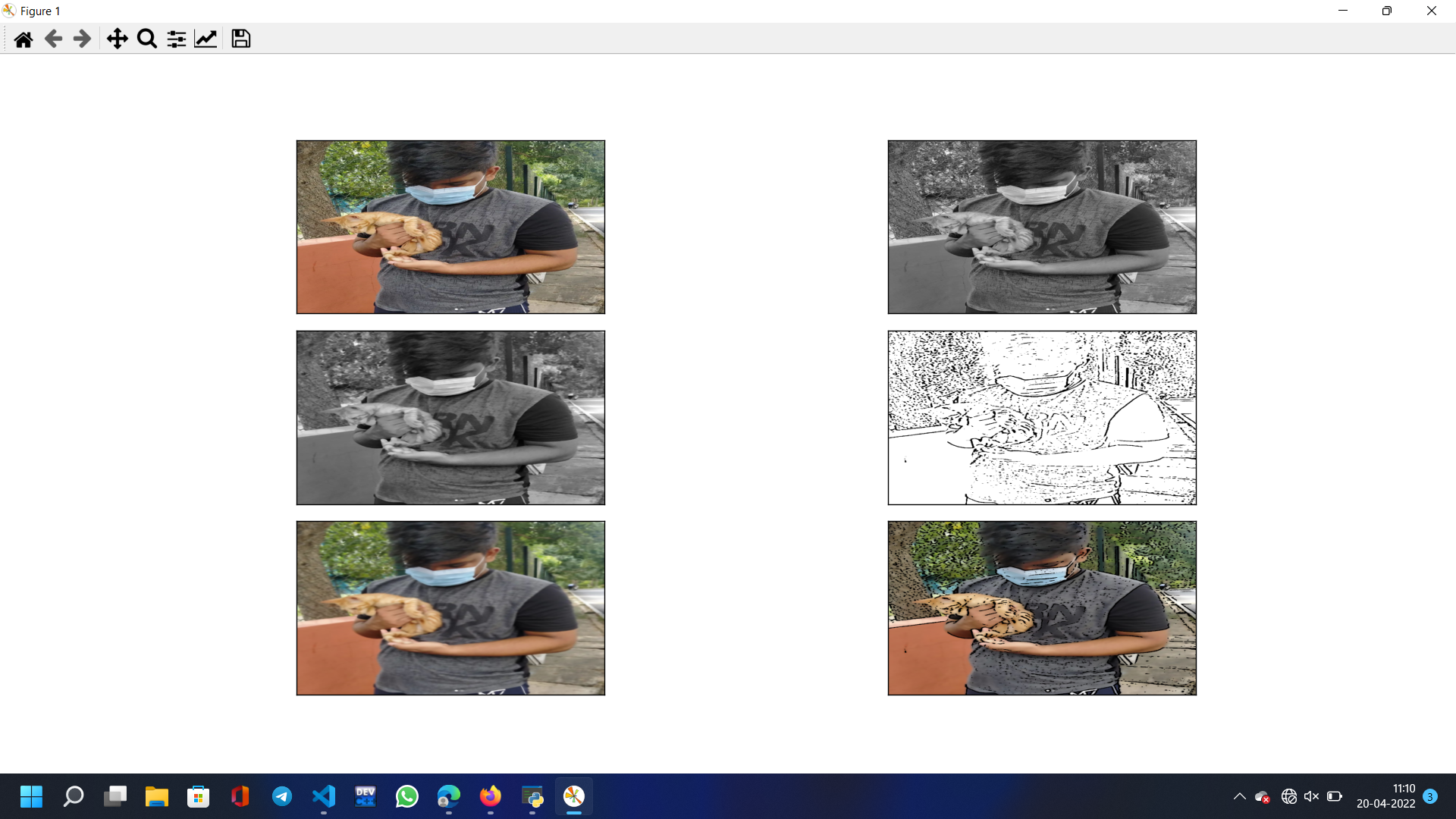
upload.configure(background='#364156', foreground='white',font=('calibri',10,'bold'))

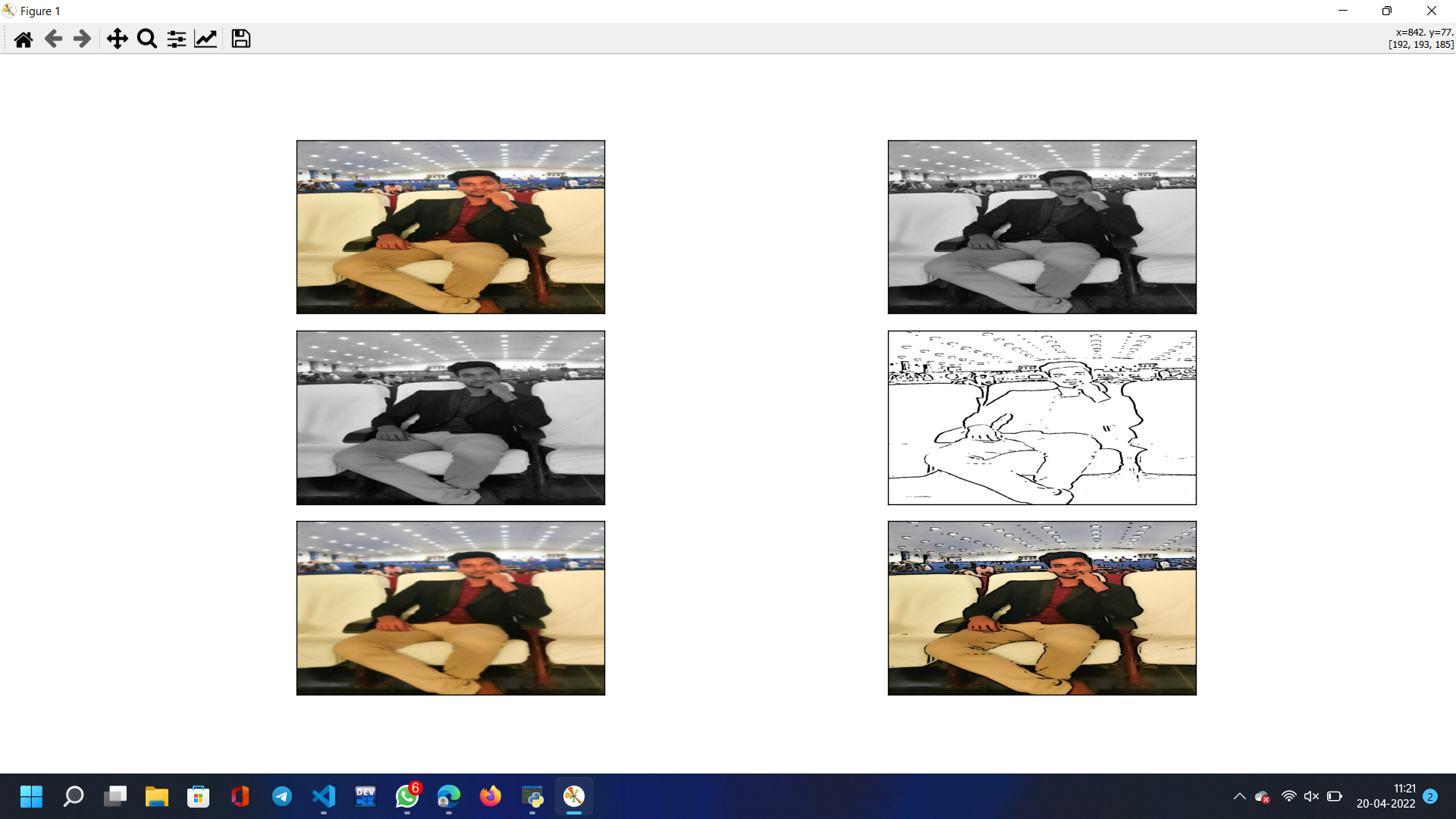
upload.pack(side=TOP,pady=50)

top.mainloop()

**Output:**

****

****

****

**Result:**

The input images were sketched and filtered successfully.