Open CV Operations

Operations to be Performed : Canny Edge Detection , Blurring Image , Contour Detection , Gray Scale

```
import tkinter as tk
from tkinter import filedialog
import cv2
from PIL import Image, ImageTk
```

- Tkinter is used for GUI , Handling File Dialogs .
- For Processing Images we use Libraries Like Open CV and PIL
- OpenCV stands for Open Computer Vision and PIL stands for Python Imaging Library (Image Processing) for processing Images

```
class ImageProcessingGUI:
    def __init__(self, root):
        self.root = root
        self.root.title("Image Processing GUI")
        self.original_img = None
        self.label_heading = tk.Label(root, text="Choose an Image",
font=("Helvetica", 18, "bold"))
        self.label_heading.pack(pady=10)
        self.btn_open_image = tk.Button(root, text="Open Image",
command=self.open_image)
        self.btn_open_image.pack(pady=5)
        self.image_label = tk.Label(root)
```

- Here we define a class called Image Processing GUI , The __init__ method initializes the GUI by creating the main window root and the window Title is set as Image Processing GUI m Original Image is set as None
- Choose an Image will be a Heading Label , button called Open Image will call open_image method when clicked , The Image Label is used to display Selected Image.

```
def open_image(self):
    file_path = filedialog.askopenfilename(filetypes=[("Image files", "*.png;*.jpg;*.jpeg")])
    if file_path:
        self.original_img = cv2.imread(file_path)
        img = Image.open(file_path)
        img = img.resize((400, 300))
        img = ImageTk.PhotoImage(img)
        self.image_label.config(image=img)
        self.image_label.image = img

        # Process image
        self.process_image()
```

- This is open image method , triggered when open image button is clicked
- It allows a file Dialog asking users to choose image file which supports formats like PNG,JPG,JPEG.

- Once a Image is selected , it reads image using OpenCV and converts to resized PIL Image Format , to display it on Image Label

```
def process_image(self):
    if self.original_img is not None:
        processed_img = self.original_img.copy()

        # Perform image processing operations
        gray_img = cv2.cvtColor(processed_img, cv2.COLOR_BGR2GRAY)
        blurred_img = cv2.GaussianBlur(processed_img, (5, 5), 0)
        edges_img = cv2.Canny(processed_img, 100, 200)

        # Check OpenCV version for findContours function
        contours_version = cv2.__version__.split('.')[0]
        if int(contours_version) >= 4:
            contours, _ = cv2.findContours(edges_img, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
        else:
            _, contours, _ = cv2.findContours(edges_img, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

        cv2.drawContours(processed_img, contours, -1, (0, 255, 0), 2)
```

- The Process Image Method is Called , after User Selects an Image , It performs various Image Processing operations on selected Image , converting into gray scale , applying blur , finding edges and contours and draws on processed Image

```
def show_processed_images(self, gray_img, blurred_img, edges_img, contour_img):
    # Resize images to medium size for display
    gray_img = cv2.resize(gray_img, (400, 300))
    blurred_img = cv2.resize(blurred_img, (400, 300))
    edges_img = cv2.resize(edges_img, (400, 300))
    contour_img = cv2.resize(contour_img, (400, 300))

    # Convert images to RGB for display with tkinter
    gray_img = cv2.cvtColor(gray_img, cv2.COLOR_BGR2RGB)
    blurred_img = cv2.cvtColor(blurred_img, cv2.COLOR_BGR2RGB)
    edges_img = cv2.cvtColor(edges_img, cv2.COLOR_BGR2RGB)
    contour_img = cv2.cvtColor(contour_img, cv2.COLOR_BGR2RGB)

    # Convert images to PIL format
    gray_img = Image.fromarray(gray_img)
    blurred_img = Image.fromarray(blurred_img)
    edges_img = Image.fromarray(edges_img)
    contour_img = Image.fromarray(contour_img)
```

- Show processed_image take proccessed Image as Arguments , resize them to medium size , converts to RGB Format, Displays each processed Image , By calling Show Image Method

```
def show_image_with_heading(self, heading, image):
    image = image.resize((400, 300))
    image = ImageTk.PhotoImage(image)

    window = tk.Toplevel()
    window.title(heading)
    label = tk.Label(window, image=image)
    label.image = image
    label.pack(pady=10)
```

- Show image with Heading displays each processeed Image in new window
- This code allows users to choose an image and displays it along with the processed images (gray scale, blurred, edges, contours) in separate windows, each with its respective heading.