|  |  |
| --- | --- |
| **Name** | **Roll No.:** |
| 1. Medimi Pratyush | B229 (70022100434) |
| 2.Eeshita Gupta | B231 (70022100442) |
| 3. Swaraj Borkar | B232 (70022100443) |

**Project Report**

Image Processing GUI using Python and Tkinter

1. **Introduction**

The "Image Processing GUI" is a Python application built with the Tkinter library, which provides a graphical user interface for loading, processing, and displaying images. This project allows users to select an image from their computer, apply various image processing filters, and view the processed image in real-time.

**2. Project Objectives**

The primary objectives of this project are as follows:

- To create a user-friendly GUI application for image processing.

- To enable users to select an image file from their computer.

- To offer image processing options such as Grayscale conversion, Gaussian Blur, and Canny Edge detection.

- To display the original and processed images within the GUI.

**3. Technologies Used**

- Python: The core programming language used for this project.

- Tkinter: A Python library used for creating the graphical user interface.

- OpenCV (cv2): Used for image processing and manipulation.

- PIL (Pillow): Used for converting and displaying images in the Tkinter interface.

**4. Application Features**

The Image Processing GUI offers the following features:

4.1. Image Selection

Users can click the "Select Image" button to choose an image file from their computer. The selected image is then displayed within the GUI.

4.2. Image Processing

Users can choose from a dropdown menu of image processing filters, including:

- None: No filter is applied.

- Grayscale: Converts the image to grayscale.

- Blur: Applies Gaussian blur to the image.

- Canny Edge: Detects edges in the image using the Canny edge detection algorithm.

4.3. Image Display

The original and processed images are displayed within the GUI. The processed image is updated in real-time as the user selects different filters.

**5. Implementation**

The project is implemented in a Python script with the following key components:

1. Image Processing App Class: The main class that manages the GUI elements and image processing functionality.
2. load\_ image(): Allows users to select an image file and loads it for processing.
3. process\_ image(): Applies the selected image processing filter to the loaded image.
4. display\_ image(): Displays the image within the GUI.
5. **Code**

import tkinter as tk

from tkinter import filedialog

import cv2

from PIL import Image, ImageTk

class ImageProcessingApp:

    def \_\_init\_\_(self, root):

        self.root = root

        self.root.title("Image Processing GUI")

        # Create a file input button

        self.input\_button = tk.Button(root, text="Select Image", command=self.load\_image)

        self.input\_button.pack()

        # Create a dropdown for selecting filters

        self.filters = ["None", "Grayscale", "Blur", "Canny Edge"]

        self.selected\_filter = tk.StringVar(root)

        self.selected\_filter.set(self.filters[0])

        self.filter\_dropdown = tk.OptionMenu(root, self.selected\_filter, \*self.filters)

        self.filter\_dropdown.pack()

        # Create a process button

        self.process\_button = tk.Button(root, text="Process Image", command=self.process\_image)

        self.process\_button.pack()

        # Create an image display area

        self.image\_label = tk.Label(root)

        self.image\_label.pack()

        self.image = None

    def load\_image(self):

        file\_path = filedialog.askopenfilename()

        if file\_path:

            self.image = cv2.imread(file\_path)

            self.display\_image()

    def process\_image(self):

        if self.image is not None:

            filter\_choice = self.selected\_filter.get()

            if filter\_choice == "Grayscale":

                processed\_image = cv2.cvtColor(self.image, cv2.COLOR\_BGR2GRAY)

            elif filter\_choice == "Blur":

                processed\_image = cv2.GaussianBlur(self.image, (7, 7), 0)

            elif filter\_choice == "Canny Edge":

                processed\_image = cv2.Canny(self.image, 100, 200)

            else:

                processed\_image = self.image

            self.display\_image(processed\_image)

    def display\_image(self, img=None):

        if img is None:

            img = self.image

        if img is not None:

            img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

            img = Image.fromarray(img)

            img = ImageTk.PhotoImage(image=img)

            self.image\_label.configure(image=img)

            self.image\_label.image = img

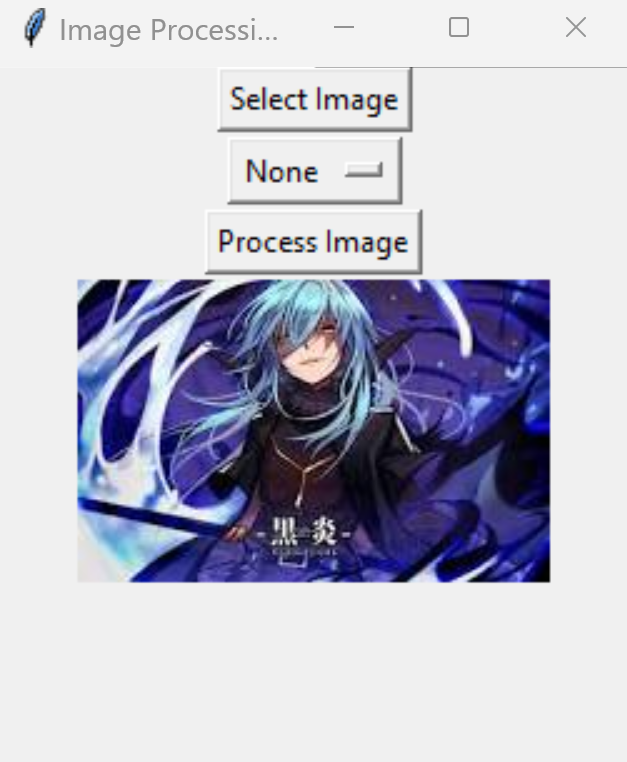
if \_\_name\_\_ == "\_\_main\_\_":

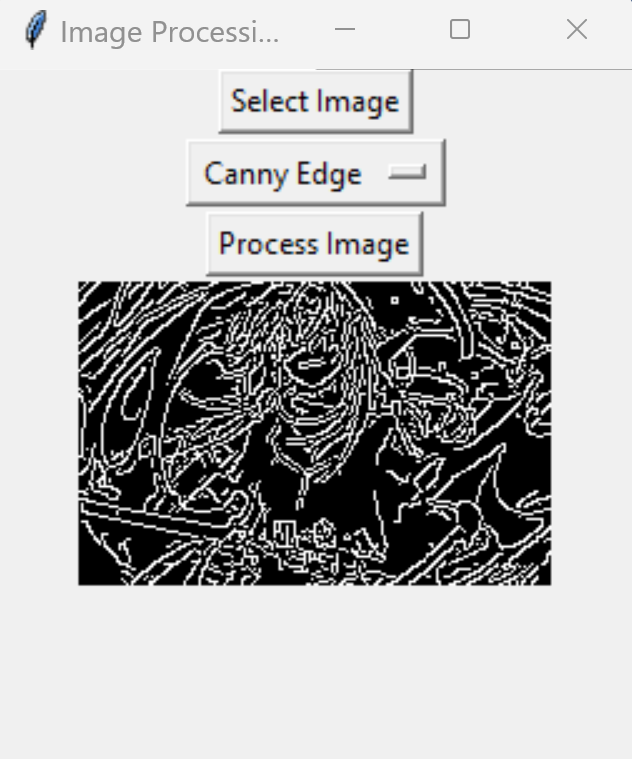
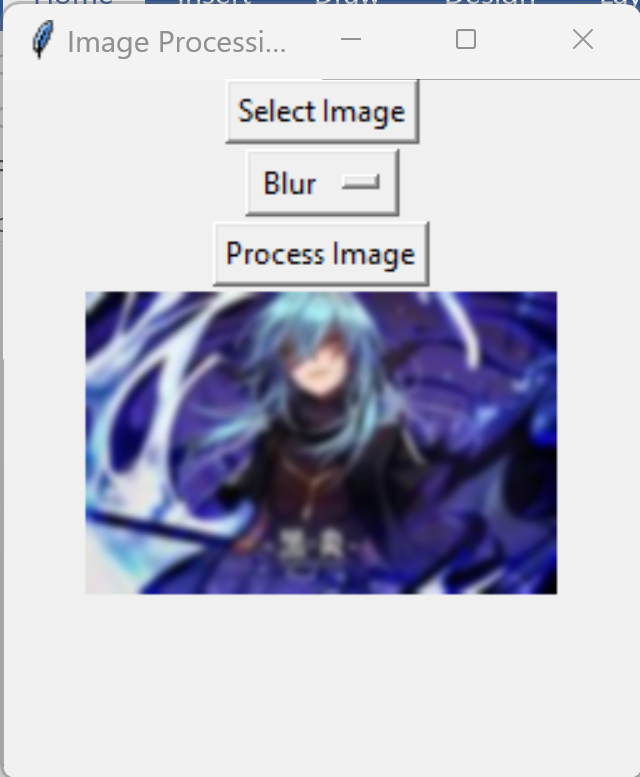
    root = tk.Tk()

    app = ImageProcessingApp(root)

    root.mainloop()

1. **Output**

****

****

**7.How to Run the Application**

To run the "Image Processing GUI" application:

1. Ensure you have Python installed.

2. Install the required libraries: `tkinter`, `opencv-python`, and `Pillow`.

3. Execute the Python script.

**8.Conclusion**

The "Image Processing GUI" project provides a simple yet effective interface for users to process and view images interactively. It demonstrates the integration of Tkinter for building a graphical user interface and OpenCV for image processing. This project can serve as a foundation for more advanced image processing applications or as a useful tool for those who need quick image manipulations without the need for extensive coding.

**9.Future Enhancements**

In future iterations, the project can be extended with the following enhancements:

1. Additional image processing filters and options.
2. Image save functionality.
3. Image comparison features (original vs. processed).
4. Batch processing of multiple images.

**10.Acknowledgments**

The project is made possible by the contributions of the open-source Python community and the libraries used. It showcases the power of Python in combining different technologies to create useful and interactive applications.

**11. References**

1. [Python Tkinter Documentation](https://docs.python.org/3/library/tkinter.html)
2. [OpenCV (cv2) Documentation](https://docs.opencv.org/master/)
3. [Pillow (PIL Fork) Documentation](https://pillow.readthedocs.io/en/stable/index.html)