SECV3213-01 ASAS PEMPROSESAN IMEJ (FUNDAMENTAL OF IMAGE PROCESSING)

ASSIGNMENT 1

GROUP 4

Name	Matric No
Muhammad Akashah Bin Bahar	A21EC0066
Lew Chin Hong	A21EC0044
Yeo Chun Teck	A21EC0148

Lecturer: Dr Md. Sah Bin Hj. Salam

Introduction

The goal of Assignment 1 is to develop a simple but user-friendly image manipulation application that allows users to select several regions of Interest (ROIs) and perform different operations such as darkening, brightening, or blackening to the specified parts within the provided picture. The application leverages fundamental arithmetic and logic operations such as multiplication to modify the pixel intensities in the specified ROIs, providing users with a flexible tool for fine-tuned image modification. Through a straightforward interface, users can interactively select and manipulate distinct areas of the image, exploring the impact of different operations on visual outcomes. This assignment not only allows us to have hands-on experience with image processing techniques, but it also promotes comprehension of the underlying ideas involved in changing pixel values to obtain desired visual effects via basic arithmetic and logic operations.

Proposed Solution

Process Flow Diagram

Proposed Solution Explanation

To achieve the functions required, we choose to include the Tkinter library for creating a graphical user interface. In this assignment, this library is used to provide:

i. Select ROI button
   - The user can click this button to select ROI and press enter for selection confirmation

ii. ROI list box
   - The selected ROI(s) will be shown in this list box
   - The user needs to select the ROI(s) listed in this list box before performing the ROI brightness adjustment, ROI blackening or deletion operations

iii. Brightness Slider
   - This slider has a value from -100 to 100
     - 0 is the initial value. The greater the value, the brighter the image, and vice versa
   - The user can move this slider to adjust the brightness level of the selected ROI(s)

iv. Blacken ROI button
   - The user can click this button to blacken the selected ROI(s)

- This blackening function is achieved by directly adjusting the brightness level to -100

v. Delete selected ROI button
   - The user can click this button to delete the selected ROI(s)

Here is the process flow explanation

1. Load original image
   - Load the image “lenna.tif”
2. Initialize data structures
   - Create a list (rois) to store the selected ROI(s)
   - Create a dictionary (roi_brightness_levels) to store the brightness level for each ROI
   - Define a global variable (updating_brightness) to confirm brightness level updates
3. Select ROI(s)
   - The user clicks the “Select ROI” button to invoke the ROI selection that used cv2.selectROI()
4. Update ROI list
   - Update and store the information about selected ROI(s), including the coordinates and brightness level
   - Selected ROI(s) are stored in the ‘rois’ list while initial brightness levels are set in the ‘roi_brightness_levels’ dictionary
5. Create a copy of original image
6. Select ROI(s) from the list
   - The selected ROI(s) will be framed by the dashed line and will be chosen to perform further operation
7. Adjust Brightness Level
   - The user will move a slider to adjust the brightness level (from -100 to 100)
   - 0 is the initial value. The greater the value, the brighter the image, vice versa
8. Apply multiplication
   - The brightness level value obtained from the slider will be used as the scale of the cv2.multiply() function (This part will be explained later)
   - The user can directly blacken the ROI via another function, that function will directly set the brightness level to -100

9. Display Original Image and Processed Image

Here is a detailed explanation of brightness level adjustment and multiplication application

1. The Tkinter Scale widget, brightness_slider, is created to allow the user to adjust the brightness
2. It has a range from -100 to 100, representing a percentage change in brightness

```
# Create a slider for brightness adjustment
brightness_slider = tk.Scale(window, label="Brightness", from_=-100, to=100, orient="horizontal")
brightness_slider.pack()
```

3. Two events are bound to the slider: 'slider_click' for press and 'slider_release' for release.
4. As mentioned previously, we declare a global variable namely 'updating_brightness' that is used to confirm a brightness adjustment
5. These events are used to control the global variable' updating_brightness'

```
# Bind the slider to respond to mouse clicks and releases
brightness_slider.bind("<ButtonPress-1>", slider_click)
brightness_slider.bind("<ButtonRelease-1>", slider_release)
```

6. 'slider_click' sets 'updating_brightness' to True when the slider is clicked
7. 'slider_release' sets 'updating_brightness' to False when the slider is released

```
# Function to set updating_brightness to True when the slider is clicked
def slider_click(event):
    global updating_brightness
    updating_brightness = True

# Function to set updating_brightness to False when the slider is released
def slider_release(event):
    global updating_brightness
    updating_brightness = False
```

8. The '<Motion>' event is bound to the slider. This event is triggered when the slider is moved
9. The 'update_brightness_level' function is called during the motion event

```
# Bind the update_brightness_level function to the slider's event
brightness_slider.bind("<Motion>", update_brightness_level)
```

10. 'update_brightness_level' checks if 'updating_brightness' is True and updates the brightness level accordingly

```
# Function to update the brightness level for the selected ROI when the brightness slider is adjusted
def update_brightness_level(event):
        global updating_brightness
        if updating_brightness:
            selected_indices = roi_listbox.curselection()
            if selected_indices:
                brightness = brightness_slider.get()
                for i in selected_indices:
                    roi_brightness_levels[i] = brightness
                update_processed_image()
```

11. The ‘cv2.multiply()’ function is used to adjust the brightness of the selected ROI
12. The ‘brightness_level’ is obtained from the dictionary ‘roi_brightness_levels’, which stores brightness levels for each ROI.

```
# Get the brightness level for this ROI
brightness_level = roi_brightness_levels.get(i, 0)
```

13. The selected region's pixel values are multiplied by a scaling factor based on the ‘brightness_level’

```
# Apply brightness adjustment
adjusted_region = cv2.multiply(selected_region, (1, 1, 1, 1), scale=1 + brightness_level / 100.0)
```

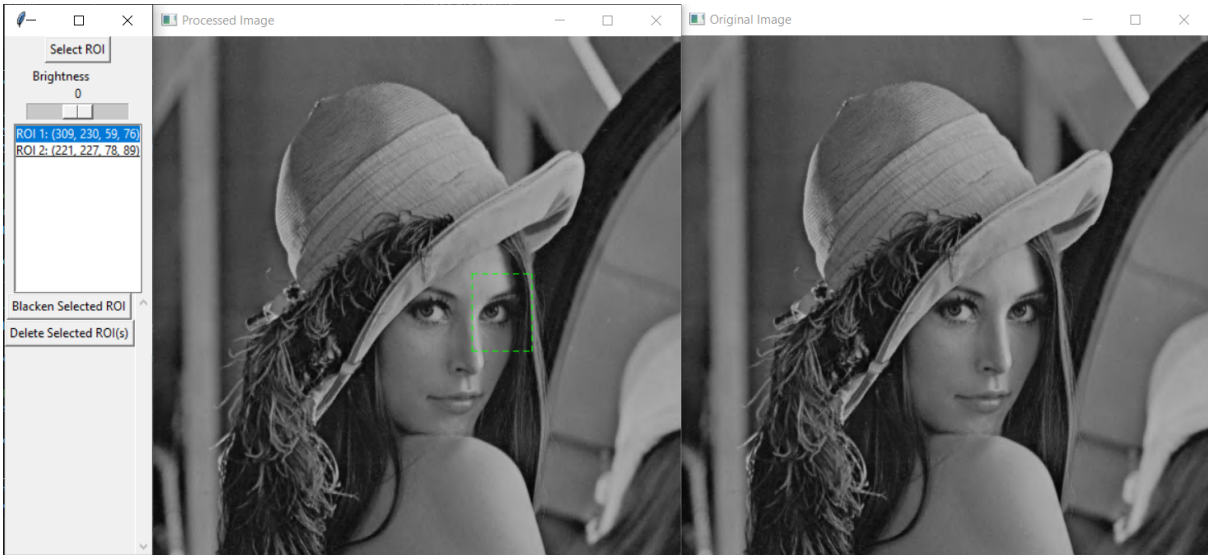
14. The adjusted region is then assigned back to the original image

```
image_copy[y:y + h, x:x + w] = adjusted_region
```

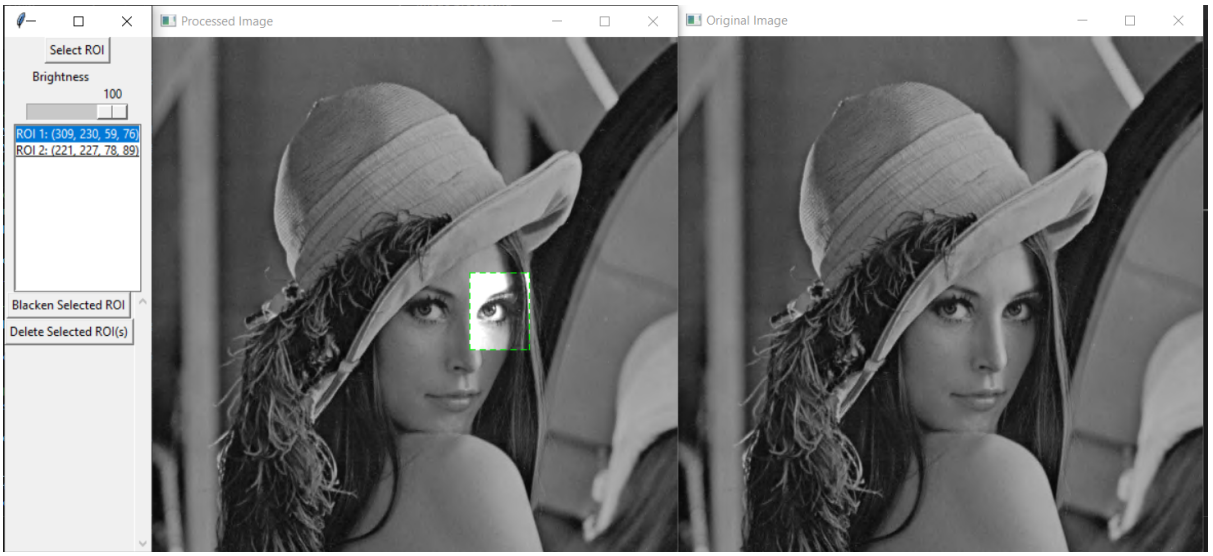
In short, the slider is associated with the ‘cv2.multiply()’ function through the ‘brightness_level’ variable. The slider events (<ButtonPress-1>, <ButtonRelease-1>, and <Motion>) are used to control when the brightness level should be updated, and this updated brightness level is then applied to adjust the brightness of the selected ROI.

Results of the solution

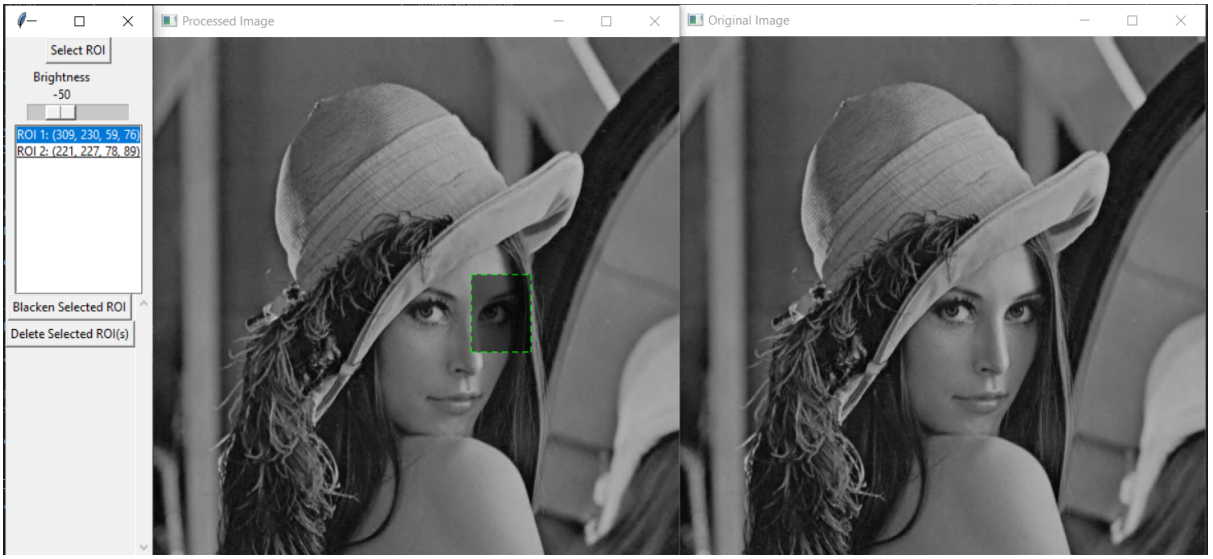
The green-dashed-line frame indicates the ROI selected (ROI with blue-highlighted in the ROI list)



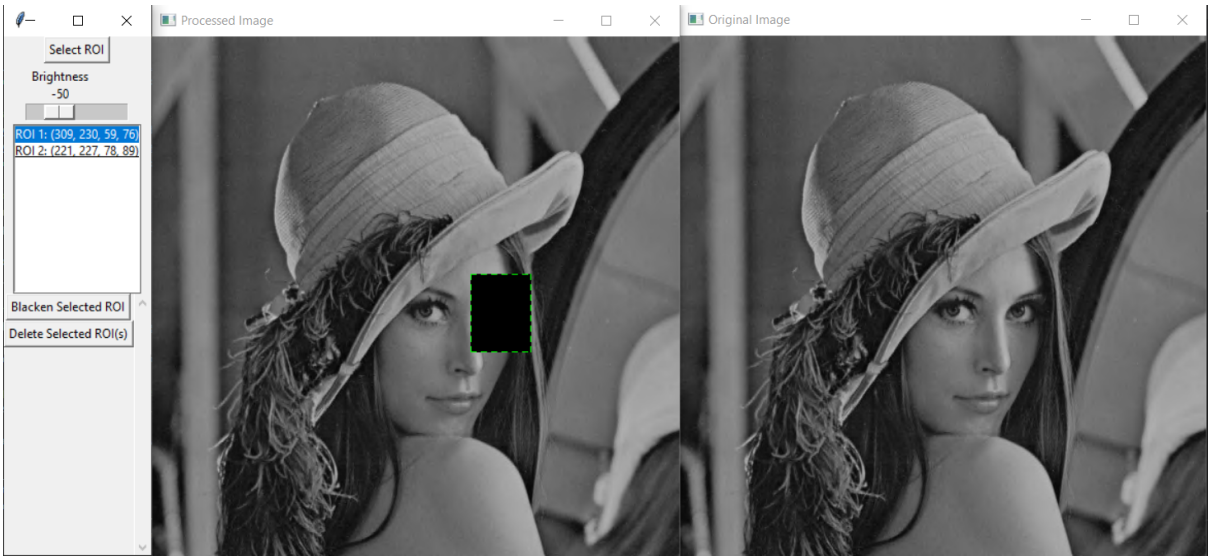
When brightness = 100



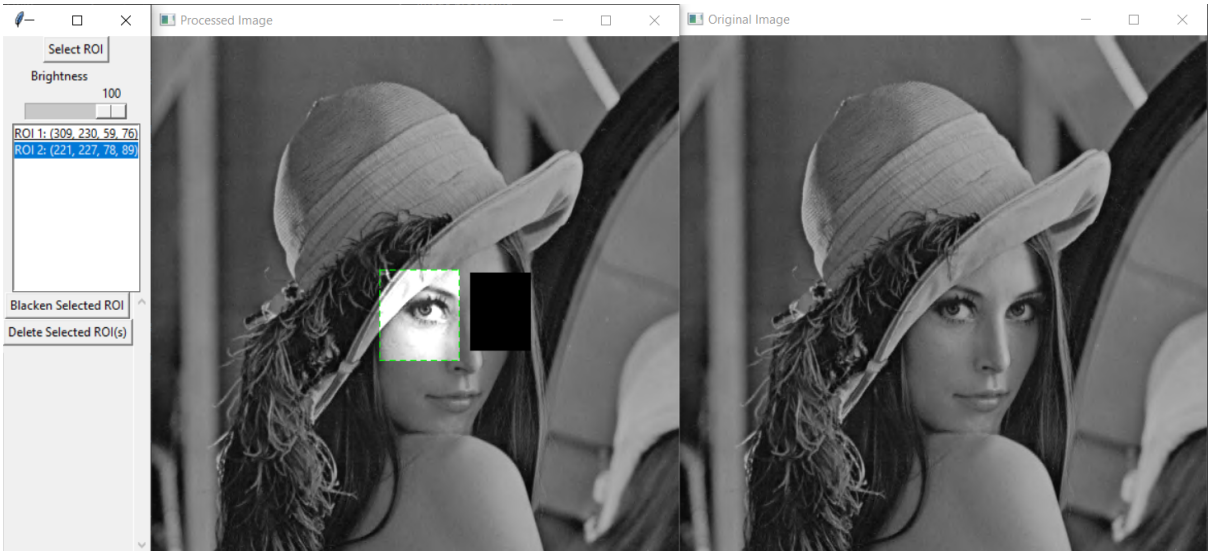
When brightness = -50



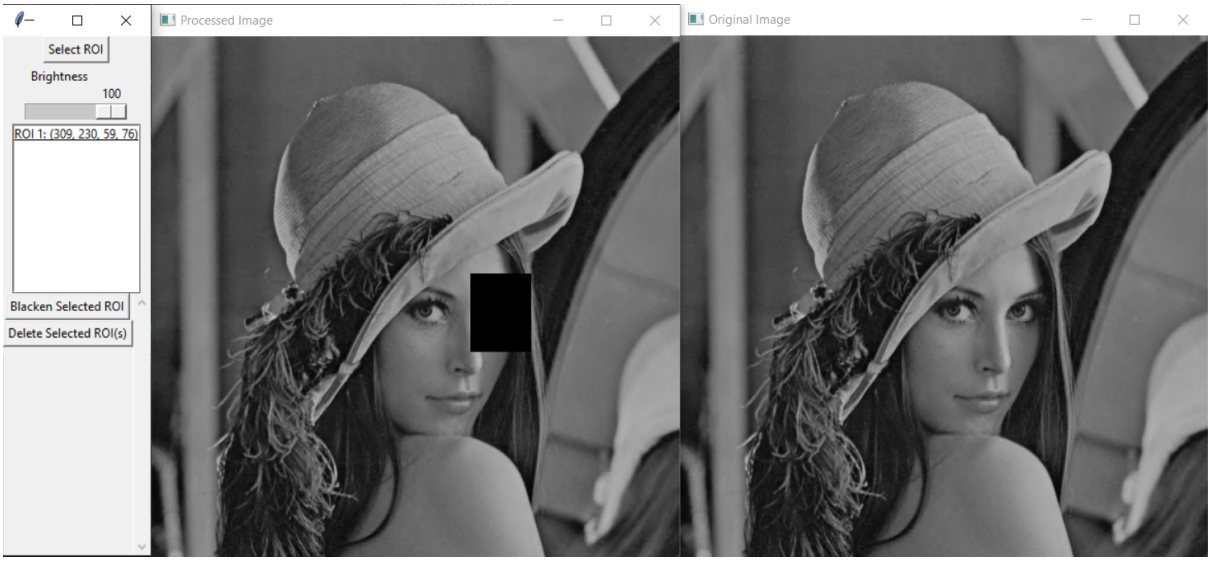
When the ‘Blacken Selected ROI’ button is pressed



Select another ROI (ROI 2) and adjust its brightness to 100



When the ‘Delete Selected ROI’ button is pressed for ROI 2



Conclusion

In conclusion, our proposed solution establishes a comprehensive graphical user interface (GUI) using the Tkinter library, facilitating the selection and manipulation of multiple Regions of Interest (ROIs) within an image. The code encompasses main functionalities, including the ability to select and delete ROIs interactively through buttons. Moreover, it incorporates a dynamic brightness adjustment feature facilitated by a Tkinter Scale widget (brightness_slider). This slider enables users to modify the brightness levels of selected ROIs, ranging from a darkening effect (-100) to a brightening effect (100). The code employs the cv2.multiply() function to implement these brightness adjustments to the chosen ROIs. The processed image, reflecting the cumulative impact of selected ROIs and their brightness alterations, is showcased in a separate window, enhancing user visualization. Furthermore, dashed frames are drawn around selected ROIs within the processed image also providing users with a clear visual indication. This solution prioritizes user interaction events, responding dynamically to button clicks, slider adjustments, and list selections. This establishes an engaging and interactive user experience. This solution, with its modular architecture, descriptive variable names, and well-organized structure, not only provides instant utility but also serves as a flexible basis for prospective modifications and extra functionality in the field of image processing applications.

Workload distribution among members

Member	Tasks				Task percentage
	System Planning	Code Contribution	Report Writing	Video Making	
Muhammad Akashah Bin Bahar	33%	33%	33%	33%	33
Lew Chin Hong	33%	33%	33%	33%	33
Yeo Chun Teck	34%	34%	34%	34%	34

Demo Video Link

https://youtu.be/BrrQOWXzD18?feature=shared