PAMANTASAN NG LUNGSOD NG MAYNILA
(University of the City of Manila)
Intramuros, Manila

Elective 3

Laboratory Activity No. 2
Image Representation, Color Models, and Image Operations

Score

Submitted by:

ALAMBRA, Joseph Nathaniel
ARAGON, Patrick Laurence M.
BANAL, Daryll L.
SENTASAS, David Bryan L.
TUTANES, Allen Christopher O.

SATURDAY (7:00 – 4:00 pm) / CPE 0332.1-1

Date Submitted
30-07-2024

Submitted to:
Engr. Maria Rizette H. Sayo

I. Objectives

This laboratory activity aims to implement the principles and techniques of image acquisition, representation, color models through MATLAB/Octave and open CV using Python

1. Acquire the image.
2. Acquire image representation.
3. Acquire image color models.
4. Modify image representation.
5. Flip Image.

II. Methods

A. Perform a task given in the presentation

- Copy and paste your MATLAB code

```
% Read an image
img = imread("C:/Users/Elitebook 840 G7/Documents/3rd Year - 3rd Sem/Elective
(Laboratory)/flower.jpg");

% Display the image
figure(1);
imshow(img);
title('Original Image');

% Get image dimensions (rows, columns, color channels)
[rows, cols, channels] = size(img);
disp(['Image size: ', num2str(rows), ' x ', num2str(cols), ' x ', num2str(channels)]);

% Check color model (grayscale or RGB)
if channels == 1
   disp('Color Model: Grayscale');
else
   disp('Color Model: RGB');
end

% Access individual pixels (example: center pixel)
center_row = floor(rows/2) + 1;
center_col = floor(cols/2) + 1;
center_pixel = img(center_row, center_col, :);
disp(['Center pixel value: ', num2str(center_pixel(:)')]);

% Basic arithmetic operations (add constant value to all pixels)
brightened_img = img + 50;
figure(2);
```

imshow(brightened_img);
title('Image Brightened');

% Basic geometric operation (flipping image horizontally)
flipped_img = fliplr(img);
figure(3);
imshow(flipped_img);
title('Image Flipped Horizontally');

B. Supplementary Activity

- Write a Python program that will implement the output in Method A.

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Read an image
img = cv2.imread("C:/Users/Elitebook 840 G7/Documents/3rd Year - 3rd
Sem/Elective (Laboratory)/flower.jpg")

# Display the image
plt.figure(1)
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.title('Original Image')

# Get image dimensions (rows, columns, color channels)
rows, cols, channels = img.shape
print(f'Image size: {rows} x {cols} x {channels}')

# Check color model (grayscale or RGB)
if channels == 1:
    print('Color Model: Grayscale')
else:
    print('Color Model: RGB')

# Access individual pixels (example: center pixel)
center_row = rows // 2
center_col = cols // 2
center_pixel = img[center_row, center_col]
print(f'Center pixel value: {center_pixel}')

# Basic arithmetic operations (add constant value to all pixels)
brightened_img = cv2.add(img, 50)
plt.figure(2)
plt.imshow(cv2.cvtColor(brightened_img, cv2.COLOR_BGR2RGB))
plt.title('Image Brightened')

# Basic geometric operation (flipping image horizontally)
flipped_img = cv2.flip(img, 1)
plt.figure(3)
plt.imshow(cv2.cvtColor(flipped_img, cv2.COLOR_BGR2RGB))
```

```
plt.title('Image Flipped Horizontally')

plt.show()
```

III. Results

Image Attribute and Color Model

MATLAB Results

```
Image size: 500 x 500 x 3
Color Model: RGB
Center pixel value: 237   112     2
```

Octave Results

```
Image size: 500 x 500 x 3
Color Model: RGB
Center pixel value: 237   112     2
>>
```

PyCharm Results

```
Image size: 500 x 500 x 3
Color Model: RGB
Center pixel value: [  2 112 237]
```

Steps:

1. Copy/crop and paste your results. Label each output (Figure1, Figure2, Figure3)

picture file: flower.jpg

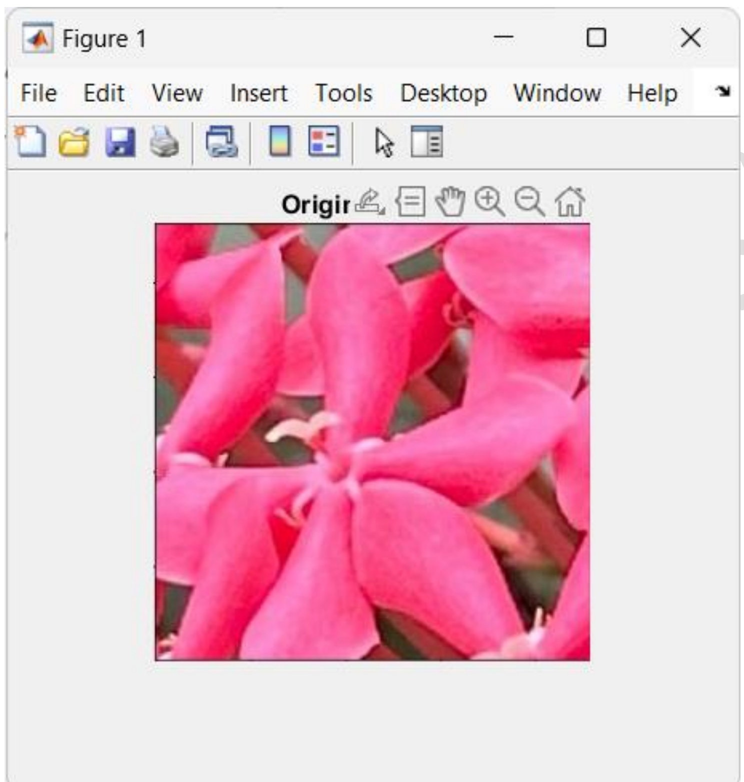


Figure 1: Flower

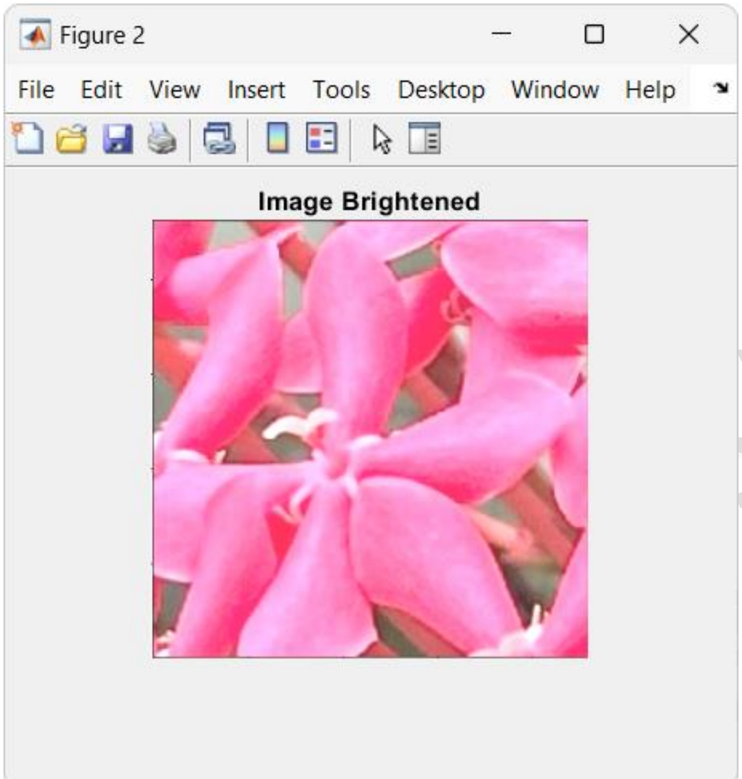


Figure 2: Image Brightened of Flower

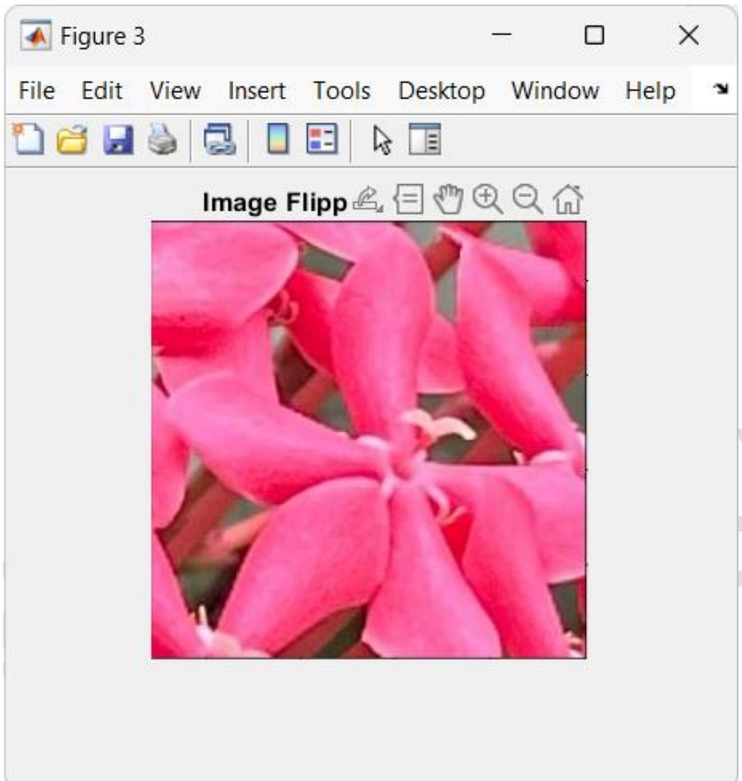
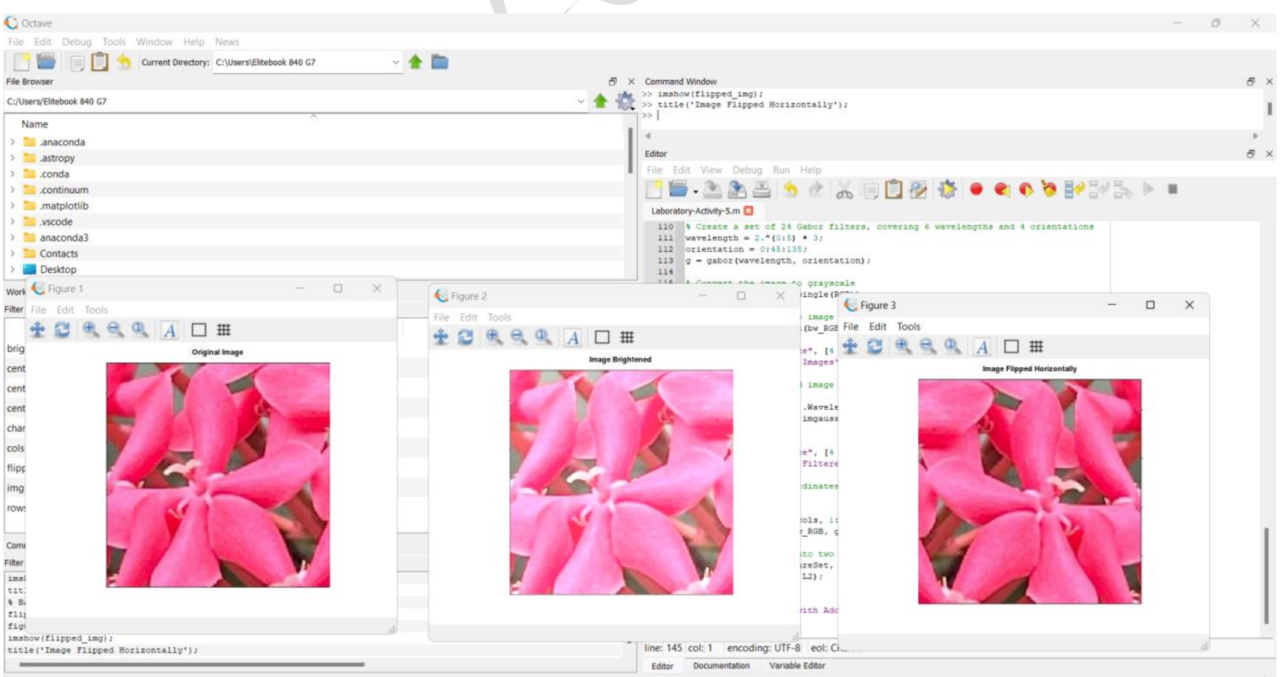
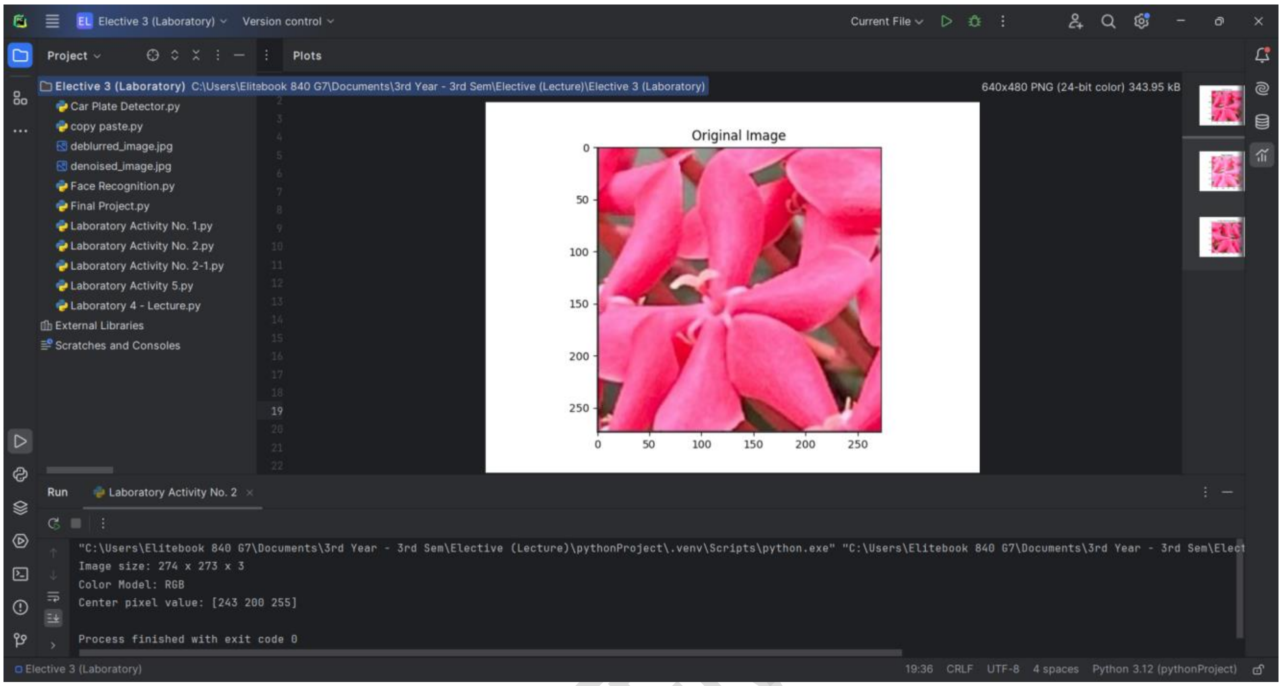


Figure 3: Image of Flower Flipped Horizontally

Octave Screenshot



PyCharm Screenshot



These codes perform the following:

1. Reads an image using imread.
2. Displays the image using imshow.
3. Gets the image dimensions (rows, columns, color channels) using size and displays them.
4. Checks the color model (grayscale or RGB) based on the number of channels.
5. Accesses the value of a specific pixel (center pixel in this case).Performs a basic arithmetic operation (adding a constant value to all pixels) to brighten theimage.
6. Performs a basic geometric operation (flipping the image horizontally) using fliplr.

Parameter Modification

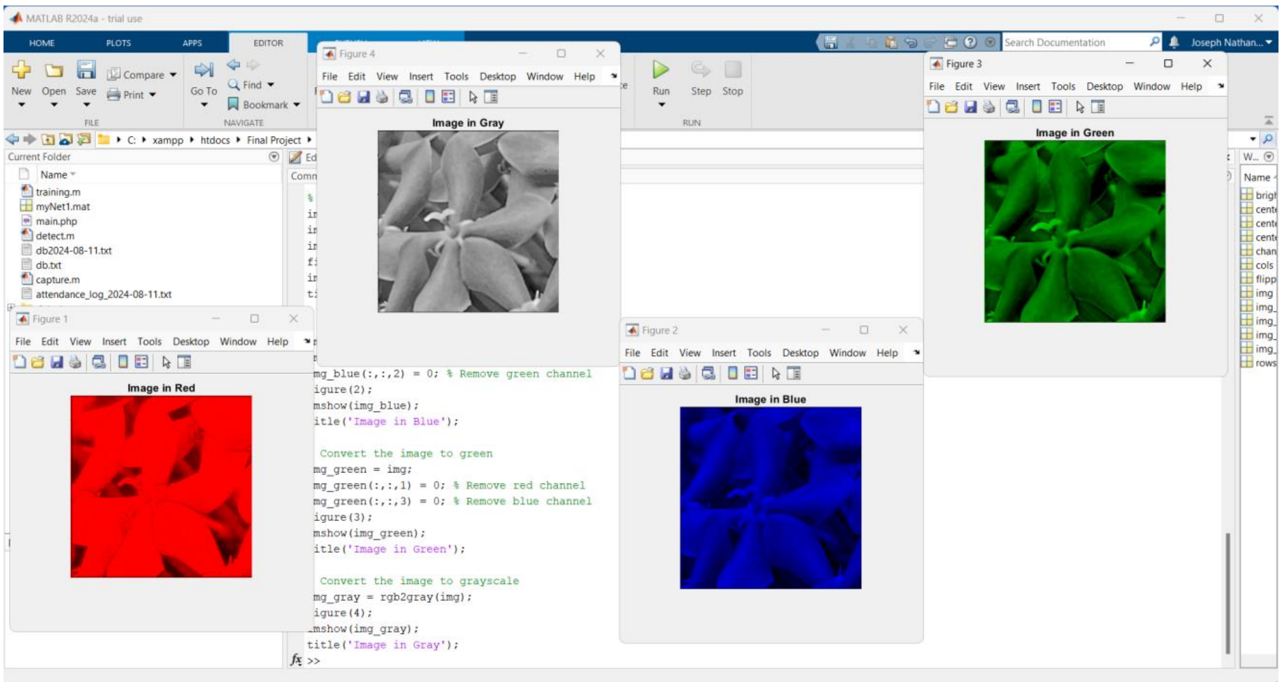
<You can modify it to explore other functionalities>

- Try displaying individual color channels for RGB images (e.g., imshow(img(:,:,1)) for red channel).

Source Code

```
% Read an image
img = imread("C:/Users/Elitebook 840 G7/Documents/3rd Year - 3rd Sem/Elective
(Laboratory)/flower.jpg");

% Convert the image to red
img_red = img;
img_red(:,:,2) = 0; % Remove green channel
img_red(:,:,3) = 0; % Remove blue channel
figure(1);
imshow(img_red);
title('Image in Red');

% Convert the image to blue
img_blue = img;
img_blue(:,:,1) = 0; % Remove red channel
img_blue(:,:,2) = 0; % Remove green channel
figure(2);
imshow(img_blue);
title('Image in Blue');

% Convert the image to green
img_green = img;
img_green(:,:,1) = 0; % Remove red channel
img_green(:,:,3) = 0; % Remove blue channel
figure(3);
imshow(img_green);
title('Image in Green');

% Convert the image to grayscale
img_gray = rgb2gray(img);
figure(4);
imshow(img_gray);
title('Image in Gray');
```

Output

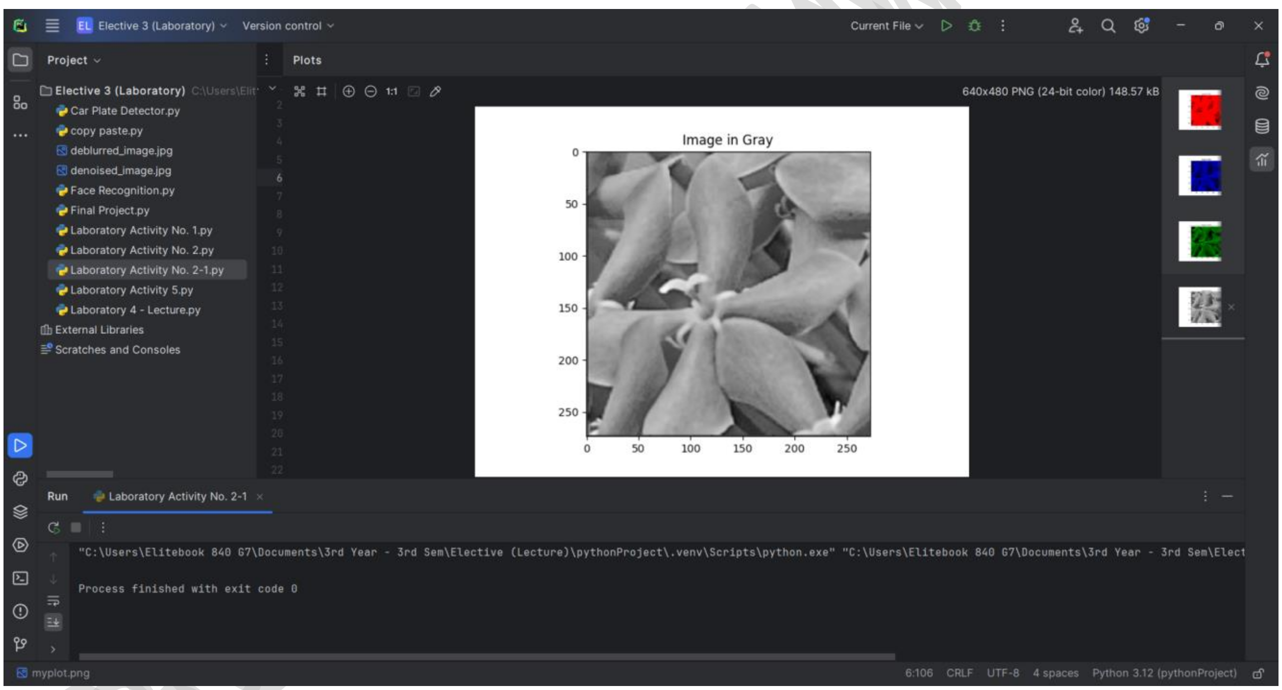


Python Source Code Equivalent

```
import cv2
import matplotlib.pyplot as plt

# Read an image
img = cv2.imread('/content/drive/MyDrive/Colab/flower.jpg')

# Convert the image to RGB for display with matplotlib
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

# Convert the image to red
img_red = img_rgb.copy()
img_red[:, :, 1] = 0  # Remove green channel
img_red[:, :, 2] = 0  # Remove blue channel
plt.figure(1)
plt.imshow(img_red)
plt.title('Image in Red')

# Convert the image to blue
img_blue = img_rgb.copy()
img_blue[:, :, 0] = 0  # Remove red channel
img_blue[:, :, 1] = 0  # Remove green channel
plt.figure(2)
plt.imshow(img_blue)
plt.title('Image in Blue')
```

```
# Convert the image to green
img_green = img_rgb.copy()
img_green[:, :, 0] = 0  # Remove red channel
img_green[:, :, 2] = 0  # Remove blue channel
plt.figure(3)
plt.imshow(img_green)
plt.title('Image in Green')

# Convert the image to grayscale
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plt.figure(4)
plt.imshow(img_gray, cmap='gray')
plt.title('Image in Gray')

plt.show()
```

Outputs



- Experiment with different arithmetic operations (subtraction, multiplication).

Source Code

```
% Read an image
img = imread("C:/Users/Elitebook 840 G7/Documents/3rd Year - 3rd Sem/Elective (Laboratory)/flower.jpg");

% Increase image size 5 times
img = imresize(img, 5);

% Display the original image
figure(1);
imshow(img);
title('Original Image');

% Get image dimensions (rows, columns, color channels)
[rows, cols, channels] = size(img);
disp(['Image size: ', num2str(rows), ' x ', num2str(cols), ' x ',
num2str(channels)]);

% Check color model (grayscale or RGB)
if channels == 1
    disp('Color Model: Grayscale');
else
    disp('Color Model: RGB');
end

% Access individual pixels (example: center pixel)
center_row = floor(rows/2) + 1;
center_col = floor(cols/2) + 1;
center_pixel = img(center_row, center_col, :);
disp(['Center pixel value: ', num2str(center_pixel(:)')]);

% Convert the image to red
img_red = img;
img_red(:,:,2) = 0; % Remove green channel
img_red(:,:,3) = 0; % Remove blue channel
figure(2);
imshow(img_red);
title('Image in Red');

% Display the image size, color model, and center pixel value for the red image
disp('Red Image:');
disp(['Image size: ', num2str(rows), ' x ', num2str(cols), ' x ',
num2str(channels)]);
disp(['Center pixel value: ', num2str(img_red(center_row, center_col, :))]);

% Convert the image to blue
img_blue = img;
img_blue(:,:,1) = 0; % Remove red channel
img_blue(:,:,2) = 0; % Remove green channel
figure(3);
```

```
imshow(img_blue);
title('Image in Blue');

% Display the image size, color model, and center pixel value for the blue
image
disp('Blue Image:');
disp(['Image size: ', num2str(rows), ' x ', num2str(cols), ' x ',
num2str(channels)]);
disp(['Center pixel value: ', num2str(img_blue(center_row, center_col, :))]);

% Convert the image to green
img_green = img;
img_green(:,:,1) = 0; % Remove red channel
img_green(:,:,3) = 0; % Remove blue channel
figure(4);
imshow(img_green);
title('Image in Green');

% Display the image size, color model, and center pixel value for the green
image
disp('Green Image:');
disp(['Image size: ', num2str(rows), ' x ', num2str(cols), ' x ',
num2str(channels)]);
disp(['Center pixel value: ', num2str(img_green(center_row, center_col, :))]);

% Convert the image to grayscale
img_gray = rgb2gray(img);
figure(5);
imshow(img_gray);
title('Image in Gray');

% Display the image size, color model, and center pixel value for the gray
image
disp('Gray Image:');
disp(['Image size: ', num2str(size(img_gray, 1)), ' x ', num2str(size(img_gray,
2)), ' x 1']);
disp(['Center pixel value: ', num2str(img_gray(center_row, center_col))]);
```

Output

```
Image size: 2500 x 2500 x 3
Color Model: RGB
Center pixel value: 237  112    2
Red Image:
Image size: 2500 x 2500 x 3
Center pixel value: 237    0    0
Blue Image:
Image size: 2500 x 2500 x 3
Center pixel value: 0  0  2
Green Image:
Image size: 2500 x 2500 x 3
Center pixel value: 0  112    0
Gray Image:
Image size: 2500 x 2500 x 1
Center pixel value: 137
```

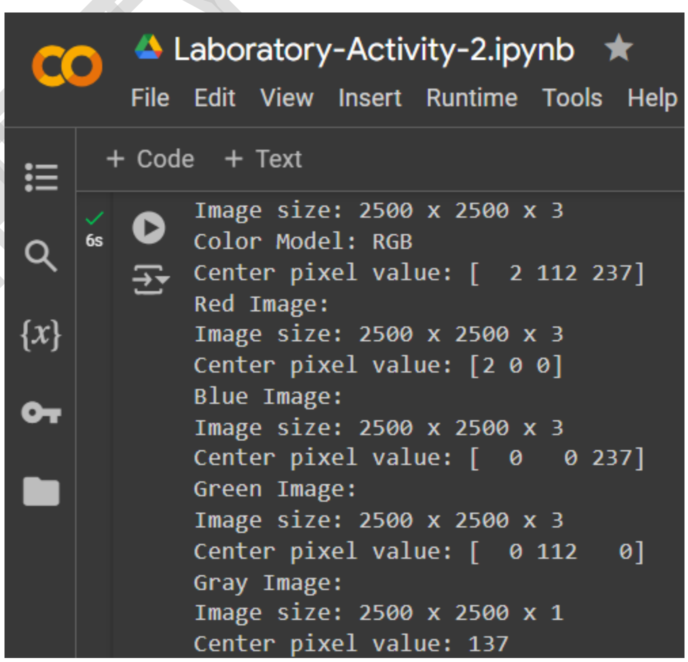

Python Source Code Equivalent

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Read an image
img = cv2.imread('/content/drive/MyDrive/Colab/flower.jpg')

# Increase image size 5 times
img = cv2.resize(img, (img.shape[1] * 5, img.shape[0] * 5))

# Display the original image
plt.figure(1)
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.title('Original Image')

# Get image dimensions (rows, columns, color channels)
rows, cols, channels = img.shape
print(f'Image size: {rows} x {cols} x {channels}')

# Check color model (grayscale or RGB)
if channels == 1:
    print('Color Model: Grayscale')
```

```
else:
    print('Color Model: RGB')

# Access individual pixels (example: center pixel)
center_row = rows // 2
center_col = cols // 2
center_pixel = img[center_row, center_col]
print(f'Center pixel value: {center_pixel}')

# Convert the image to red
img_red = img.copy()
img_red[:, :, 1] = 0  # Remove green channel
img_red[:, :, 2] = 0  # Remove blue channel
plt.figure(2)
plt.imshow(cv2.cvtColor(img_red, cv2.COLOR_BGR2RGB))
plt.title('Image in Red')

# Display the image size, color model, and center pixel value for the red
image
print('Red Image:')
print(f'Image size: {rows} x {cols} x {channels}')
print(f'Center pixel value: {img_red[center_row, center_col]}')

# Convert the image to blue
img_blue = img.copy()
img_blue[:, :, 0] = 0  # Remove red channel
img_blue[:, :, 1] = 0  # Remove green channel
plt.figure(3)
plt.imshow(cv2.cvtColor(img_blue, cv2.COLOR_BGR2RGB))
plt.title('Image in Blue')

# Display the image size, color model, and center pixel value for the blue
image
print('Blue Image:')
print(f'Image size: {rows} x {cols} x {channels}')
print(f'Center pixel value: {img_blue[center_row, center_col]}')

# Convert the image to green
img_green = img.copy()
img_green[:, :, 0] = 0  # Remove red channel
img_green[:, :, 2] = 0  # Remove blue channel
plt.figure(4)
plt.imshow(cv2.cvtColor(img_green, cv2.COLOR_BGR2RGB))
```

```
plt.title('Image in Green')

# Display the image size, color model, and center pixel value for the green
image
print('Green Image:')
print(f'Image size: {rows} x {cols} x {channels}')
print(f'Center pixel value: {img_green[center_row, center_col]}')

# Convert the image to grayscale
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plt.figure(5)
plt.imshow(img_gray, cmap='gray')
plt.title('Image in Gray')

# Display the image size, color model, and center pixel value for the gray
image
print('Gray Image:')
print(f'Image size: {img_gray.shape[0]} x {img_gray.shape[1]} x 1')
print(f'Center pixel value: {img_gray[center_row, center_col]}')

plt.show()
```

Outputs

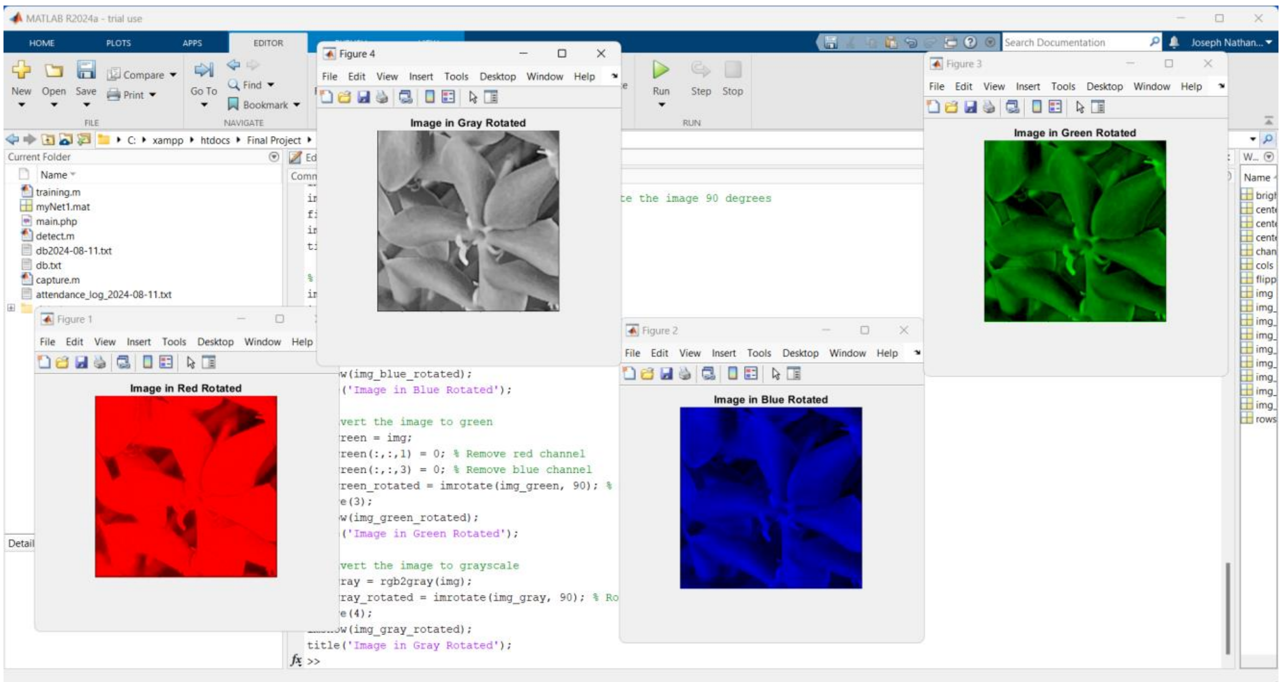


- Explore other geometric operations like image rotation (imrotate).

Source Code

```
% Read an image
img = imread("C:/Users/Elitebook 840 G7/Documents/3rd Year - 3rd Sem/Elective
(Laboratory)/flower.jpg");

% Convert the image to red
img_red = img;
img_red(:,:,2) = 0; % Remove green channel
img_red(:,:,3) = 0; % Remove blue channel
img_red_rotated = imrotate(img_red, 90); % Rotate the image 90 degrees
figure(1);
imshow(img_red_rotated);
title('Image in Red Rotated');

% Convert the image to blue
img_blue = img;
img_blue(:,:,1) = 0; % Remove red channel
img_blue(:,:,2) = 0; % Remove green channel
img_blue_rotated = imrotate(img_blue, 90); % Rotate the image 90 degrees
figure(2);
imshow(img_blue_rotated);
title('Image in Blue Rotated');

% Convert the image to green
img_green = img;
img_green(:,:,1) = 0; % Remove red channel
img_green(:,:,3) = 0; % Remove blue channel
img_green_rotated = imrotate(img_green, 90); % Rotate the image 90 degrees
figure(3);
imshow(img_green_rotated);
title('Image in Green Rotated');

% Convert the image to grayscale
img_gray = rgb2gray(img);
img_gray_rotated = imrotate(img_gray, 90); % Rotate the image 90 degrees
figure(4);
imshow(img_gray_rotated);
title('Image in Gray Rotated');
```

Output

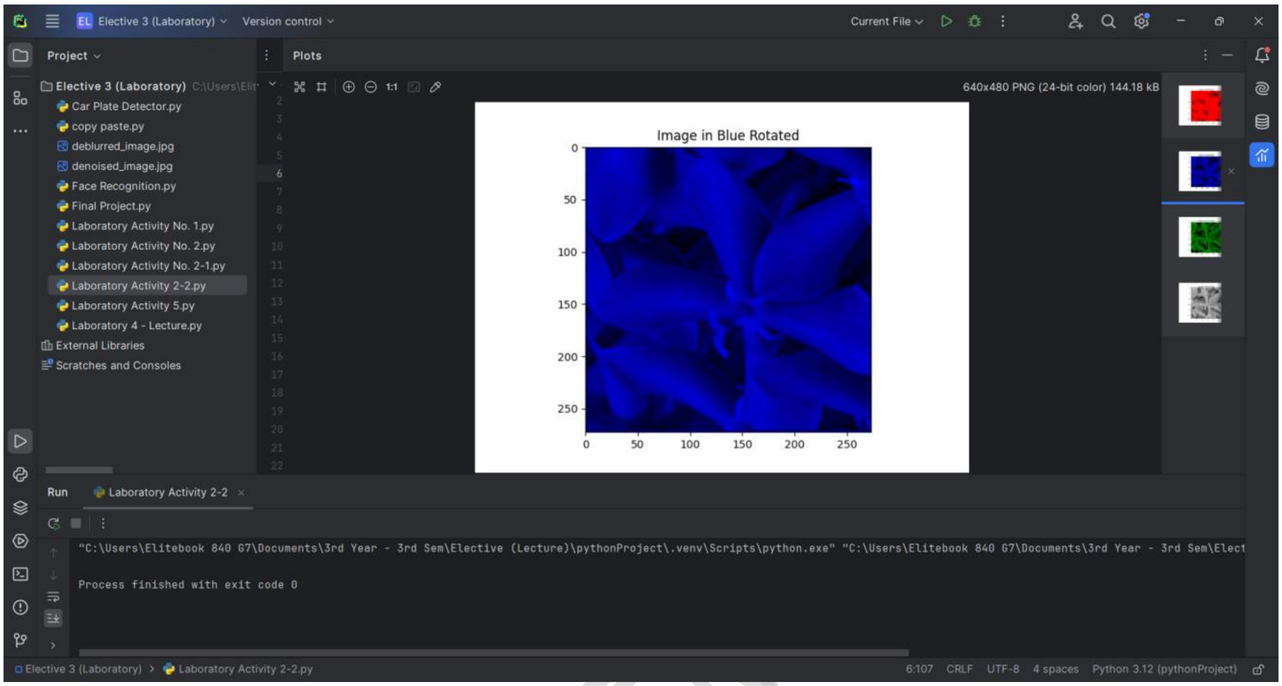


Python Source Code Equivalent

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Read an image
img = cv2.imread('/content/drive/MyDrive/Colab/flower.jpg')

# Convert the image to RGB for display with matplotlib
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

# Convert the image to red
img_red = img_rgb.copy()
img_red[:, :, 1] = 0  # Remove green channel
img_red[:, :, 2] = 0  # Remove blue channel
img_red_rotated = np.rot90(img_red)  # Rotate the image 90 degrees
plt.figure(1)
plt.imshow(img_red_rotated)
plt.title('Image in Red Rotated')
```

```
# Convert the image to blue
img_blue = img_rgb.copy()
img_blue[:, :, 0] = 0  # Remove red channel
img_blue[:, :, 1] = 0  # Remove green channel
img_blue_rotated = np.rot90(img_blue)  # Rotate the image 90 degrees
plt.figure(2)
plt.imshow(img_blue_rotated)
plt.title('Image in Blue Rotated')

# Convert the image to green
img_green = img_rgb.copy()
img_green[:, :, 0] = 0  # Remove red channel
img_green[:, :, 2] = 0  # Remove blue channel
img_green_rotated = np.rot90(img_green)  # Rotate the image 90 degrees
plt.figure(3)
plt.imshow(img_green_rotated)
plt.title('Image in Green Rotated')

# Convert the image to grayscale
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img_gray_rotated = np.rot90(img_gray)  # Rotate the image 90 degrees
plt.figure(4)
plt.imshow(img_gray_rotated, cmap='gray')
plt.title('Image in Gray Rotated')

plt.show()
```

Outputs


Elective 3 (Laboratory)
Version control
Current File
Project
Plots
Elective 3 (Laboratory)
Car Plate Detector.py
copy paste.py
deblurred_image.jpg
denoised_image.jpg
Face Recognition.py
Final Project.py
Laboratory Activity No. 1.py
Laboratory Activity No. 2.py
Laboratory Activity No. 2-1.py
Laboratory Activity 2-2.py
Laboratory Activity 5.py
Laboratory 4 - Lecture.py
External Libraries
Scratches and Consoles
640x480 PNG (24-bit color) 144.18 kB
Image in Blue Rotated
Run
Laboratory Activity 2-2
"C:\Users\Elitebook 840 G7\Documents\3rd Year - 3rd Sem\Elective (Lecture)\pythonProject\.venv\Scripts\python.exe" "C:\Users\Elitebook 840 G7\Documents\3rd Year - 3rd Sem\Elect
Process finished with exit code 0
Elective 3 (Laboratory) > Laboratory Activity 2-2.py
6:107 CRLF UTF-8 4 spaces Python 3.12 (pythonProject)

2. Visualize the results, analyze and interpret:

The applied algorithms in this laboratory activity had several notable effects on the image and demonstrated varying degrees of effectiveness in achieving the desired outcomes.

a. **Image Brightening**:
   - **Effect**: The image was brightened by adding a constant value to all pixel intensities. This made the colors of the image appear more vivid and enhanced visibility in darker areas.
   - **Effectiveness**: This method was effective in making the image appear more vibrant and visible. However, it also increased the risk of pixel saturation, where some pixel values might exceed the maximum allowable intensity, causing loss of detail in the brightest areas.

b. **Image Flipping**:
   - **Effect**: The image was flipped horizontally. This changed the orientation of the image, effectively creating a mirror image.
   - **Effectiveness**: This operation was effective in changing the perspective without altering the image content. It is particularly useful in applications where different orientations of the same image are needed, such as in data augmentation for machine learning.

c. **Color Channel Manipulation**:
   - **Effect**: The image was converted to display only the red, green, or blue channel by zeroing out the other channels. This isolated the contribution of each color channel to the overall image.
   - **Effectiveness**: This was effective in visualizing the individual color channels and understanding their impact on the composite image. It is useful for color analysis and for applications that require channel-specific processing.

d. **Image Rotation**:
   - **Effect**: The image was rotated by 90 degrees. This altered the orientation, making vertical elements appear horizontal and vice versa.
   - **Effectiveness**: This operation was effective in changing the image orientation without altering the pixel values. It is useful in scenarios that require different perspectives of the same image.

e. **Image Resizing**:
   - **Effect**: The image size was increased by a factor of 5. This enlarged the image and made finer details more noticeable.
   - **Effectiveness**: This was effective in enhancing the visibility of small details. However, it also increased the computational load and the risk of pixelation, where the image may appear blocky due to the interpolation method used.

Overall, the applied algorithms were effective in achieving their respective goals, such as enhancing visibility, changing orientation, and isolating color channels. The results demonstrate the potential of these techniques in various image processing applications, although care must be taken to manage potential drawbacks like pixel saturation and increased computational demands.

IV. Conclusion

The group successfully demonstrated the implementation of various principles and techniques in image processing using MATLAB/Octave and OpenCV with Python during this laboratory activity. They achieved the objective of acquiring images from specified file paths and displaying them using appropriate functions in both MATLAB and Python. This step is crucial for any image processing task as it serves as the input data. The group was able to read and display the acquired images, confirming that the image acquisition process was correctly executed, thereby verifying that the image file was correctly loaded into the working environment.

The group also analyzed the color models of the images to determine whether they were grayscale or RGB. This analysis was essential in understanding the color composition of the images. By determining the color model, the group could decide on the appropriate processing techniques to apply, depending on whether the image was in grayscale or RGB format.

Additionally, the group isolated and visualized individual color channels (red, green, and blue). This enhanced their understanding of the image's color composition and allowed them to see the contribution of each color channel to the overall image. This step was important for tasks that require color-specific processing, such as enhancing certain features within an image.

The group performed various image modifications, including brightening, flipping, and rotating the images. These operations demonstrated basic arithmetic and geometric transformations, which are fundamental in image processing. Brightening the image made the colors more vivid, flipping the image changed its orientation, and rotating the image provided a different perspective, all of which are useful in different image processing scenarios.

Furthermore, the group successfully flipped the images horizontally, showcasing their ability to alter the orientation of the images. This technique is useful for data augmentation and other applications that require different perspectives of the same image. Overall, this laboratory activity provided the group with hands-on experience with essential image processing techniques, reinforcing theoretical concepts and demonstrating their practical applications.

References

[1] D.J.D. Sayo. "University of the City of Manila Computer Engineering Department Honor Code," PLM-CpE Departmental Policies, 2020.

[2] Admin. (2019, May 17). Image Processing 101 Chapter 1.2: Color models. Dynamsoft Blog. https://www.dynamsoft.com/blog/insights/image-processing/image-processing-101-color-models/

[3] Keim, R. (2018, August 17). Understanding Color Models Used in Digital Image Processing. All About Circuits. https://www.allaboutcircuits.com/technical-articles/understanding-color-models-used-in-digital-image-processing/

[4] Pranjal. (2022, August 22). Color space Image Processing (Explained with RGB, CMY, HSI, Color Models). CronJ. https://www.cronj.com/blog/color-space-image-processing/