INT3404E 20 - Image Processing: Homework 1

Nguyen Minh Kien

1 Load Image:

```python
def load_image(image_path):
    image = cv2.cvtColor(cv2.imread(image_path), cv2.COLOR_BGR2RGB)
    return image
```

The above function firstly reads an image from the specified path and then converts it from **BGR** color space to **RGB** color space using OpenCV. This ensures that the image is in the correct format for further processing or display of other functions.

2 Display Image:

```python
def display_image(image, title="Image"):
    plt.title(title)
    imagePlot = plt.imshow(image)
    plt.show()
```

The display_image function utilizes **Matplotlib** library to display an image:
1. Firstly, **plt.title(title)** sets the title of the image display window to the string provided in the **title** paraameter. If no title is given when the function is called, it defaults to "Image".
2. Then, a NumPy array, which represents the input image and contained in the **image** parameter, is passed to the **imshow()** function.
3. Finally, by calling the **show()** function of **Matplotlib** libary, the image's graphical representation will be generated and a window will be opened to display the input image with the specified title.

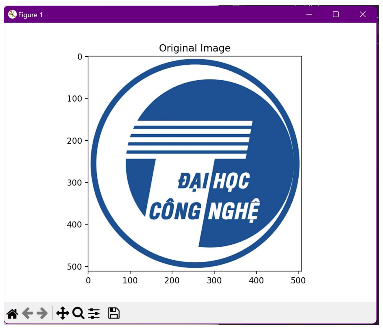

Figure 1: Pop-up window to display the input image

3 Gray-scale Image:

```python
def grayscale_image(image):
    grayscaleImage = image.copy()
    R = np.array(image[:, :, 0])
    G = np.array(image[:, :, 1])
    B = np.array(image[:, :, 2])
    avg = R * .299 + G * .587 + B * .114
    for i in range (0, 3):
        grayscaleImage[:, :, i] = avg
    return grayscaleImage
```

The above function, converts a color image into gray-scale. Here is the pipeline of how it works:
1. Firstly, it creates a copy of the input image.
2. Then, it separates the **Red**, **Green**, and **Blue** channels of the image into 3 separate arrays **(R, G, B)**.
3. It calculates the average intensity of the color using the formula below:

$$avg = R * .299 + G * .587 + B * .114$$

The above formula is a standard method for converting an image to gray-scale, where the weights are chosen based on the human eye's sensitivity to these colors.
4. Then, it replaces each color channel in the copied image with this average intensity, result in turning the input image into gray-scale, and returns the converted image.
The final result can be seen in Figure 1 below.

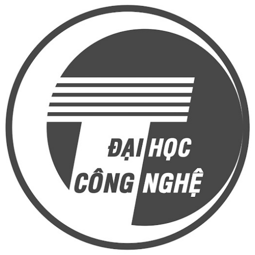

Figure 2: Converted to Gray-scale Image

4 Save Image:

```
def save_image(image, output_path):
    originalPath = os.getcwd()
    newPath = output_path.split("/")
    try:
        os.mkdir(newPath[0], False)
    except FileExistsError:
        pass
    os.chdir(os.path.join(os.getcwd(), '') + newPath[0])
    cv2.imwrite(newPath[1], image)
    os.chdir(originalPath)
```

In summary, this is how the save_image function works:
1. Firstly, it stores the original working directory in a variable called **originalPath** for future usages. The specified output is then split by the "/" character for processing and stored in an array called **newPath**.
2. The following **try:** and **except FileExistError:** block attempts to to create any necessary directory with the name specified in the **newPath** array. If the directory already exists, it simply continues without raising an error in the terminal.
3. The function then saves the input image to the specified location, and restores the original working directory.

5 Flip Image:

```
def flip_image(image):
    res = cv2.flip(src = image, flipCode = 1)
    return res
```

The above function flip the image horizontally using **cv2.flip** function. **cv2.flip(src, flipCode)** requires 2 parameters:

- *src* is the source image.
- *flipCode* is a flag which is used to identify the axis of rotation, i.e., 0 is x-axis, 1 is y-axis, -1 is both axes.
The final result is shown in Figure 2 below.

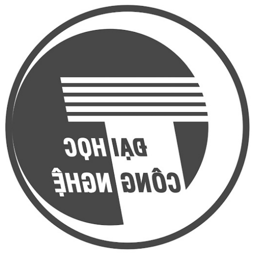

Figure 3: Flipped Image

6 Rotate Image:

```
def rotate_image(image, angle):
    height, width = image.shape[:2]
    rotateMatrix = cv2.getRotationMatrix2D(
        center = (width / 2, height / 2),
        angle = angle,
        scale = 1
    )
    res = cv2.warpAffine(
        src = image,
        M = rotateMatrix,
        dsize = (width, height)
    )
    return res
```

The above function rotates an image by a specified angle (in degrees) using a 2D rotation matrix. Here is the pipeline of how it works:
1. It calculates the height and width of the input image.
2. Then, a **2D Rotation Matrix** is created by using the **cv2.getRotationMatrix2D** function:
- *center* is the center of rotation, which in this case is the center of the image.
- *angle* is the input angle of rotation.
- *scale* is the scaling factor which scales the image.
3. The rotation is applied to the image using **cv2.warpAffine** function.
- *src* is the source image.
- *M* is the calculated rotation matrix.
- *dsize* is the size of the output image, which is the same as the input image in this case.
The final result is presented in Figure 3 below.

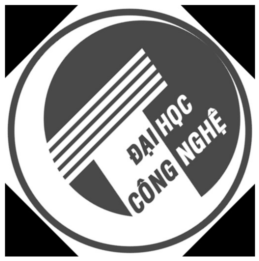

Figure 4: Rotated Image