INT3404E 20 - Image Processing: Homework 1

Vu Nguyet Hang 22028079

March 2024

1 Problem

A picture or image can be represented as a **NumPy** array of "pixels", with dimensions H × W× C, where H is the height, W is the width, and C is the number of color channels. To illustrate the coordinate system, the origin is at the top left corner and the first dimension indicates the Y (row) direction, while the second dimension indicates the X (column) dimension. Typically we will use an image with channels that give the Red, Green, and Blue "level" of each pixel, which is referred to in the short form RGB. The value for each channel ranges from 0 (darkest) to 255 (lightest). However, when loading an image through **Matplotlib**, this range will be scaled from 0 (darkest) to 1 (brightest) instead, and will be a real number, rather than an integer. This report shows results of performing several manipulations to the original image.

Figure 1: Original image

2 Settings

2.1 Libraries

The libraries utilized are **OpenCV**, **Matplotlib**, and **Numpy**, implemented in Python, using Visual Studio Code virtual environment for development.

2.2 Basic Functions

Reading an image

To read an image, *imread*() method of **OpenCV** library is used. This method loads an image from a specified file. If the image cannot be read, method returns an empty matrix.

```
def load_image(image_path):
    return cv2.imread(image_path)
```

Saving an image

Use **OpenCV**'s function *imwrite*() to save an image to a specific file, the first argument is the filename, the second argument is the image to be saved.

```
def save_image(image, output_path):
    cv2.imwrite(output_path, image)
```

Displaying an image

OpenCV loads images in the BGR (Blue - Green - Red) channel order. **Matplotlib**, on the other hand, expects images in the RGB (Red - Green - Blue) channel order. Therefore, it is crucial to convert the format correctly to avoid color distortions.

```
def display_image(image, title="Image"):
    RGB_img = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    plt.imshow(RGB_img)
    plt.title(title)
    plt.axis('off')
    plt.show()
```

3 Image Processing Functions

Converting an image to gray scale

In a grayscale image, the pixel value of the 3 channels will be the same for a particular X, Y coordinate. The equation for the pixel value [1] is given by:

$$p = 0.299R + 0.587G + 0.114B$$

Where the R, G, B are the values for each of the corresponding channels. We will do this by creating an array called img_gray with the same shape as img

```
def grayscale_image(image):
    height, width = image.shape[:2]
    img_gray = np.zeros((height, width), dtype=np.uint8)

    for i in range(height):
        for j in range(width):
            R, G, B = image[i, j]
            p = 0.299 * R + 0.587 * G + 0.114 * B
            img_gray[i, j] = p

        return img_gray
```

Figure 2: Gray image

Figure 3: Gray flipped image

Flipping an image

In **OpenCV**, an image can be flipped using the function *flip*(). We can flip the image across X-axis, Y-axis and across both axes. It accepts a flag **flipCode** as an argument: 0 (flip horizontally), any positive integer (flip vertically). Below is the code for flipping the image horizontally.

```
def flip_image(image):
    return cv2.flip(image, 1)
```

Rotating an image

To rotate an image by an arbitrary angle with **OpenCV**, we use *getRotationMatrix2D()*. This function takes the following input parameters:

1. *center*: the central point of rotation in the given image
2. *angle*: the degrees of rotation

3. *scale*: isotropic scaling factor

This function returns a transformation matrix which can be used on any image using the *warpAffine()* function.

```
def rotate_image(image, angle):
    height, width = image.shape[:2]
    center =  (width / 2, height / 2)

    rotation_matrix = cv2.getRotationMatrix2D(center, angle, 1)
    rotate_image = cv2.warpAffine(image, rotation_matrix, (width, height))

    return rotate_image
```

Figure 4: Gray rotated image

4 Source Code

Source code and images are placed in GitHub: link.