INT3404E 20 - Image Processing: Homework 1

Lê Vũ Minh - 21020649

1 Basic Python

Completed on UET Course!

2 Google Colaboratory

Submitted on UET Course and at this link on GitHub!

3 OpenCV, Numpy, Matplotlib, and LaTeX

- The code for the required image transformation functions was committed at the following GitHub link
- At the lecturer's request, the code is now included in the following report.

First, we will observe the original image, before applying any transformation:

Figure 1: Original image

3.1 Accessory functions

- Loading image with `cv2`:

```python
# Load an image from file as function
def load_image(image_path):
    """
    Load an image from file, using OpenCV
    """
    image = cv2.imread(image_path)
    return image
```

- Displaying image in `cv2` with `matplotlib.pyplot`

```python
# Display an image as function
def display_image(image, title="Image"):
    """
    Display an image using matplotlib. Rembember to use plt.show() to display the image
    """
    plt.imshow(image)
```

```
    plt.title(title)
    plt.show()
```

- Saving an image

```
# Save an image as function
def save_image(image, output_path):
    """
    Save an image to file using OpenCV
    """
    cv2.imwrite(output_path, image)
```

3.2 Gray-scaled image

To apply the gray-scale effect on an image, there are 2 approaches. First, is to use `cv2`'s built-in `cvtColor` method to change immediately change the color channel. By default, the image is loaded under the `BGR` format (blue-green-red). We are changing it to the `GRAY` channel.

```
# Grayscale an image as a function
def grayscale_image(image):
    return cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

Additionally, the more manual method suggested by the problem involves a linear combination of the 3 values on the color channel of the image for each pixel, to convert the pixel from the RGB channel to the grayscale channel.

```
# Grayscale an image as function
def grayscale_image(image):
    height, width, channel_no = image.shape
    img_gray = np.zeros((height, width), dtype=np.uint8)

    for y in range(height):
        for x in range(width):
            b, g, r = image[y, x]
            img_gray[y, x] = int(0.299 * r + 0.587 * g + 0.114 * b)

    return img_gray
```

The function yields the following image:

Figure 2: Gray-scaled image

3.3 Flipping image horizontally

In OpenCV, a loaded image is treated like a `numpy` 3D array. Therefore, we can flip the image horizontally by performing a Numpy array flip operation on the horizontal axis:

```python
# flip an image as function
def flip_image(image):
    """
    Flip an image horizontally using OpenCV
    """
    return image[:, ::-1]
```

This function, in combination with the gray-scale function earlier, yields the following image:

Figure 3: Gray-scaled and flipped image

3.4 Rotating image

We are using `cv2`'s `getRotationMatrix2D` and `warpAffine` methods to rotate the image:

```python
# rotate an image as function
def rotate_image(image, angle):
    """
    Rotate an image using OpenCV. The angle is in degrees
    """
    image_center = tuple(np.array(image.shape[1::-1]) / 2)
    rot_mat = cv2.getRotationMatrix2D(image_center, angle, 1.0)
    result = cv2.warpAffine(image, rot_mat, image.shape[1::-1], flags=cv2.INTER_LINEAR)
    return result
```

The result of the 3 functions combined yields the following image:

Figure 4: Rotated image