INT3404E 20 - Image Processing: Homeworks 1

Cao Trung Hieu

1 Grayscaling Image

```python
def grayscale_image(image):
    """
    Convert an image to grayscale. Convert the original image to a grayscale image.
    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
    """

    # Copy image
    img = image[:,:,:]
    w, h, _ = image.shape

    R = np.array(img[:, :, 0])
    G = np.array(img[:, :, 1])
    B = np.array(img[:, :, 2])

    # Grayscale image
    img[:, :, 0] = 0.299 * R + 0.587 * G + 0.114 * B
    img[:, :, 1] = img[:, :, 0]
    img[:, :, 2] = img[:, :, 0]

    return img
```

The function **grayscale_image** is used to convert an image to gray_scale. This function receives a RGB image and converts that image to gray_scale. The function will convert image using the formula (1).

$$p = 0.299R + 0.587G + 0.114B \tag{1}$$

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.
On the function, there are two main step including creating a copy of the original image and gray_scale copied image. For example, The figure 2 is an gray_scale image of the original image (figure 1).

Figure 1: Original Image

2 Flipping Image

Figure 2: Grayscale Image

```python
def flip_image(image):
    """
    Flip an image horizontally using OpenCV
    """
    return cv2.flip(image, 1)
```

The above function flip the image horizontally using **cv2.flip** function. This function has two main 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.

After flipping, the image is shown in figure 2, the result is presented in figure 3

Figure 3: Flipped Image

3 Rotating Image

```python
    def rotate_image(image, angle):
    """
    Rotate an image using OpenCV. The angle is in degrees
    """
    (h, w) = image.shape[:2]

    center = (w // 2, h // 2)

    # perform the rotation
    M = cv2.getRotationMatrix2D(center, angle, scale=1.0)
    img = cv2.warpAffine(image, M, (w, h))

    return img
```

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 coordinate of center.
2. Then, a **2D Rotation Matrix** is created by using the **cv2.getRotationMatrix2D** function:
   (a) *center* is the center of rotation, which in this case is the center of the image.
   (b) *angle* is the input angle of rotation.
   (c) *scale* is the scaling factor which scales the image.
3. The rotation is applied to the image using **cv2.warpAffine** function.
   (a) *src* is the source image.
   (b) M is the calculated rotation matrix.
   (c) *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 4 below.

Figure 4: Rotated Image

4 Loading Image

```python
def load_image(image_path):
    """
    Load an image from file, using OpenCV
    """
    return cv2.cvtColor(cv2.imread(image_path), cv2.COLOR_BGR2RGB)
```

The above function uses cv2 to load image and convert image from BGR to RGB.

5 Displaying Image

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

The above function uses matplotlib to show image.

6 Saving Image

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

The above function uses cv2 to save image.