INT3404E 20 - Image Processing: Homework 1

Doan Duc Kien - 21020207

This report presents the implementation of three functions in homework 1, namely flip_image, rotate_image and grayscale_image. Each section includes the corresponding function's code, a brief explanation and an example result.

1 flip_image

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

The function *flip_image* flips an image horizontally using OpenCV's cv2.flip function with a flipCode of 1, which means in the horizontal. The result is illustrated in figure 1.

(a) Original image (b) Flipped image

Figure 1: Result of flip_image

2 rotate_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)
    M = cv2.getRotationMatrix2D(center, angle, 1.0)
    rotated = cv2.warpAffine(image, M, (w, h))
    return rotated
```

To rotate an image by a specific angle, 2 steps are needed:

- 1. Calculate a rotation matrix using cv2.getRotationMatrix2D. The matrix is given by:

$$\begin{bmatrix} \alpha & \beta & (1-\alpha)\cdot \texttt{center.x} - \beta \cdot \texttt{center.y} \\ -\beta & \alpha & \beta \cdot \texttt{center.x} + (1-\alpha)\cdot \texttt{center.y} \end{bmatrix}$$

  where

$$\alpha = \texttt{scale} \cdot \cos \texttt{angle},$$
$$\beta = \texttt{scale} \cdot \sin \texttt{angle},$$
$$center = (x, y) \text{ is the center point of the image}$$

- 2. Apply the transformation given by the rotation matrix using cv2.warpAffine.

Figure 2 illustrates the original image after a 45-degree left rotation.

(a) Original image (b) Rotated Image (45°)

Figure 2: Result of rotate_image

3 grayscale_image

```
def grayscale_image(image):
    img_gray = 0.299 * image[:, :, 0:1] + 0.587 * image[:, :, 1:2] + 0.114 * image[:, :, 2:]
    return np.broadcast_to(img_gray.astype(image.dtype), image.shape)
```

To convert an image to grayscale format, we construct an array whose elements are given by a weighted average of RGB values in the original image:

$$p = 0.229R + 0.587G + 0.114B$$

Making use of NumPy's addition operator on arrays, we can directly average the RGB channels. Since the resulting array only has one channel, we broadcast it to make a 3 channel image that has identical values across channels. An example of input and output of this function is shown in figure 3.

(a) Original image (b) Grayscale image

Figure 3: Result of grayscale_image