INT3404E 20 - Image Processing: Homeworks 2

22028129 Tăng Vĩnh Hà

1 Summary of the report

Digital image is presented by a matrix with N rows and M column. One element in the matrix is called "Pixel" and has its own coordinates (x,y), indicating that this pixel is on the x^{th} row and in the y^{th} column. Digital images can be broadly categorized into two types based on their color information: gray scale images and color images.

- **Gray scale images**: use shades of gray to represent intensity or brightness levels, ranging from black to white and they lack color information. The value of a pixel, called gray level, can range from 0 up to the maximum scale level **L** (it is usually set that $\mathbf{L} = 255$).
- **Color images**: contain color information in addition to gray level information. Each pixel is represented by a combination of values from different color channels. For example, when in RGB mode, each pixel is represented by three color channels: Red, Green, and Blue. With 8-bit representation for RGB image, each color has values ranging from 0 to 255. The combination of these three color channel values determines the final color of the pixel.

This report aims to introduce 3 basic images processing function: turning a RGB image into a gray scale image, flipping an image and rotating an image using Matplotlib's and OpenCV's built-in functions; along with necessary functions to display and save images.

2 Basic functions

2.1 Function: load image

Purpose: load an image in the format of OpenCV library

```
import numpy as np
import matplotlib as plt
import cv2
def load_image(image_path):
    image = cv2.imread(image_path)
    # need to convert the image in BGR channels into RGB color
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    return image
```

2.2 Function: display image

Purpose: display an image using Matplotlib's built-in functions.

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

2.3 Function: save image

Purpose: save and export (processed) images to given path.

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

3 Function: gray scale image

Purpose: transform a RGB image to a gray scale image by multiplying its original matrix: m x n x 3 with the vector: [0.2989, 0.587, 0.114] (m x n x 3 corresponding to the 3 channel: Red, Green, Blue). It means the the gray scale value of a pixel in the new picture will be:

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

with R,G,B are the corresponding values for each color channel of the same pixel in the original image.

```
def grayscale_image(image):
    height = image.shape[0]
    width = image.shape[1]

    img_gray = np.zeros((height, width))

    weights = np.array([0.2989, 0.5870, 0.1140])

    for i in range(height):
        for j in range(width):
            orig = image[i, j]
            gray = np.multiply(orig, weights)
            img_gray[i,j] = np.sum(gray)

    return img_gray
```

Output:

Hình 1: Input: Original image

Hình 2: Output: Gray scale version

4 Function: flip image

Purpose: flip an image horizontally with OpenCV's built-in function. Used functions with explained parameters below:

- **cv2.flip(src, flipCode[, dst])**:
  - **Parameters**:
    - src: Input array.
    - dst: Output array of the same size and type as src.

    - flip code: A flag to specify how to flip the array; 1 means flipping horizontally, around y-axis.
  - **Return**: an image.

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

Output:

Hình 3: Input: Original image

Hình 4: Output: Flipped version

5 Function: rotate image

Purpose: rotating an image with given angle. Rotating is a type of affine transformation which preserves lines and parallelism. Used functions with explained parameters below:

- **cv2.getRotationMatrix2D(center, angle, scale)**
  - **Parameters**
    - center: center of the rotation in the source image - often in tuple data type.
    - angle: angle for rotating, positive for anti-clockwise and negative for clockwise.
    - scale: scaling factor for the image.
  - **Return**: rotation matrix M (this matrix is used to multiply with original image to get the final rotated image) - further details of this matrix will not be discussed.
- **cv2.warpAffine(src, M, dsize, dst, flags, borderMode, borderValue)**
  - **Parameters**:
    - src: input image.
    - M: transformation matrix.
    - dsize: size of the output image.
    - Other parameters: not used in this report so there will not be any discussion.
  - **Return**: an image

```python
def rotate_image(image, angle):
    height = image.shape[0]
    width = image.shape[1]
    center = (width/2, height/2)
    size = (width, height)
    rot_mat = cv2.getRotationMatrix2D(center, angle, 1.0)
    res = cv2.warpAffine(image, rot_mat, size)
    return res
```

Output:

Hình 5: Input: Original image

Hình 6: Output: Rotated version