Image Processing Homework No.1

> Student ID：41047902S　Name：鄭淮薰

Problem Statement

1. Input a color image *C*(*R*,*G*,*B*)
2. Output the color image *C*
3. Transform the color image *C* into a grayscale image *I* by *I* = (*R*+*G*+*B*)/3
4. Show the grayscale image (Read image files: including JPG files, BMP files, and PNG files.)

Input / Output

Using `python` to run this program, and here are some parameters user can use to specify the input and output

```
usage: hw1.py [-h] [-i IMAGE] [-o OUTPUT] [-t TYPE]
options:
  -h,         --help           | show this help message and exit
  -i IMAGE,   --image IMAGE    | path to the image
  -o OUTPUT, --output OUTPUT   | name of the output image
  -t TYPE,    --type           | TYPE  type of the output image
```

Test Results

Test case 1 uses JPG image files as input for testing. `Fig1.2` is a color image file named `jpg_color.jpg`, and `Fig1.3` is the result of `jpg_grayscale.jpg` after being converted to grayscale.

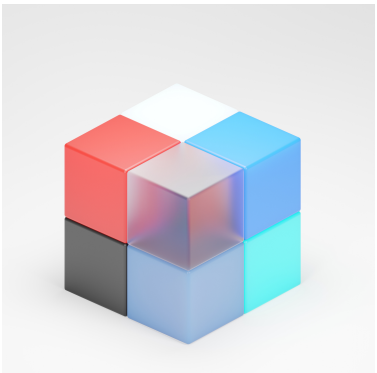

Fig1.1：Color image *C* (Input)

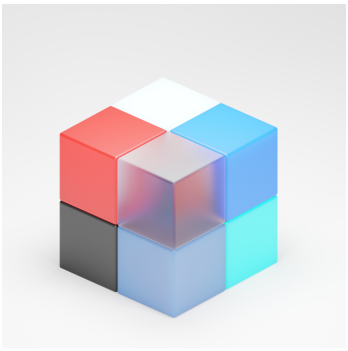

Fig1.2：Color image *C* (Output)

Fig1.3：GaryScale image *I*

Test case 2 uses BMP image files as input for testing. `Fig2.2` is a color image file named `bmp_color.jpg`, while `Fig2.3` is the result of `bmp_grayscale.jpg` after being converted to grayscale.

Fig2.1：Color image *C* (Input)

Fig2.2：Color image *C* (Output)

Fig2.3：GaryScale image *I*

Test case 3 uses PNG image files as input for the test results. `Fig3.2` is a color image file named `png_color.jpg`, while `Fig3.3` is the result of `png_grayscale.jpg` after being converted to grayscale.

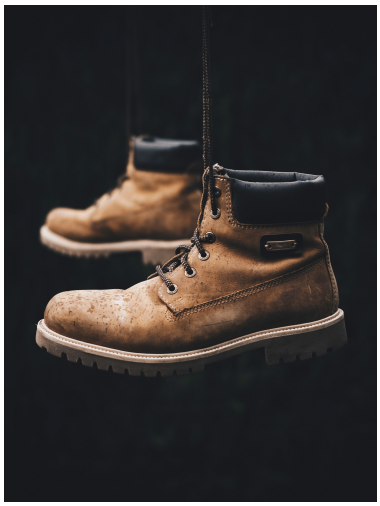

Fig3.1：Color image *C* (Input)

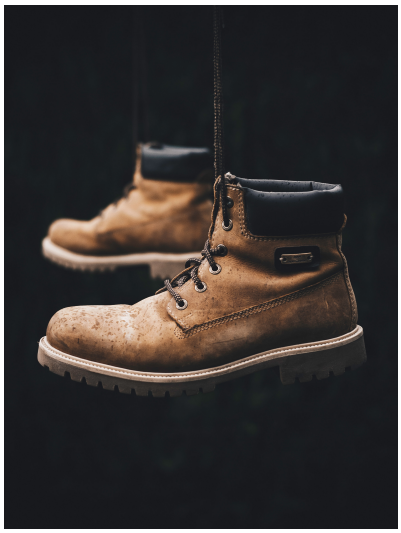

Fig3.2：Color image *C* (Output)

Fig3.3：GaryScale image *I*

Comments

In this assignment, I used the OpenCV package to process images and successfully read various common image formats, including JPG, BMP, and PNG.

While converting images into grayscale, I experimented with two different methods. First, I utilized the formula provided in the assignment, which is $I = (R + G + B)/3$, to determine the average value of the three color channels for each pixel, obtaining the corresponding grayscale value. Although this method yielded satisfactory results, its processing speed was

relatively slow, taking several seconds to process a simple image. Therefore, I tried the `cvtColor` method provided by the `cv2` package, which employs a more efficient and accurate formula, $Y = 0.299R + 0.587G + 0.114B$, to perform grayscale conversion. This method not only improved processing speed but also produced superior image quality.

To improve the flexibility of the program, I also used the `argparse` package to process command line parameters. Users can specify input image paths, output image names, and formats through the command line, making it more convenient to complete image processing tasks.

Source Code

```
import cv2
import argparse

# parse the command line arguments
parser = argparse.ArgumentParser()
parser.add_argument('-i', '--image', type=str, default='jpg_image.jpg', help='path to the image')
parser.add_argument('-o', '--output', type=str, default='output', help='name of the output image')
parser.add_argument('-t', '--type', type=str, default='jpg', help='type of the output image')
args = parser.parse_args()

# Read the image
img = cv2.imread(args.image)
cv2.imwrite(args.output + '_color.' + args.type, img)

# Convert the image to grayscale
## Method 1: using numpy
# Convert to grayscale using the formula I = (R+G+B)/3
# gray = np.zeros((img.shape[0], img.shape[1]), dtype=np.uint8)
# for i in range(img.shape[0]):
#     for j in range(img.shape[1]):
#         gray[i,j] = np.sum(img[i,j]) // 3

## Method 2: using cv2.cvtColor
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Output the image
cv2.imwrite(args.output + '_grayscale.' + args.type, img)
```