Computer Vision Homework 1

B03902042 宋子維

Figure 1: lena.bmp

Description

There are two parts in this homework. For part 1, we need to implement the function to manipulate three images, which are upside-down lena.bmp, right-side-left lena.bmp and diagonally mirrored lena.bmp. In this part, we cannot use any library calls except for basic image I/O. For part 2, we are asked to use any software to rotate lena.bmp 45 degrees clockwise, shrink lena.bmp in half, and binarize lena.bmp at 128 to get a binary image.

Part 1: Programming

I use python to implement part 1. There are totally three python programs, upside-down.py, right-side-left.py and mirror.py, where I use python package **pillow** to process basic image I/O. In the three programs, there are some similar code fragments:

1. `original = Image.load(sys.argv[1])`: load the file `sys.argv[1]` (e.g., *lena.bmp* in hw1) as an **Image** object and return it to `original`.

2. `img = Image.new(original.mode, original.size)`: create a new **Image** object with the same mode and the size as `original` (e.g., gray scale and 512x512 in hw1), and return it to `img`.

3. `pix = img.load()`: return an image access object of **Image** object `img` to `pix`, which offers us to use `pix[x, y]` to get the pixel value at position (x, y).

4. `width, height = original.width, original.height`: assign the width and height of `original` to `width` and `height`. (e.g., 512 and 512 in hw1)

```
 1 import sys
 2 from PIL import Image
 3
 4 try:
 5     original = Image.open(sys.argv[1])
 6 except:
 7     print ("Fail to open", sys.argv[1])
 8     exit()
 9
10 upside_down = Image.new(original.mode, original.size)
11
12 pix_ori, pix_ud = original.load(), upside_down.load()
13
14 width, height = original.width, original.height
```

Figure 2-1: Similar code fragments in three programs

5. `img.save(sys.argv[2])`: save the **Image** object `img` to the file `sys.argv[2]` with JPEG format.

```
20 upside_down.save(sys.argv[2])
```

Figure 2-2: Similar code fragments in three programs

1. Upside-down lena.bmp

Run `$ python3.5 upside-down.py $IMG_IN $IMG_OUT`, and the program will generate an upside-down image of **IMG_IN** (e.g., *lena.bmp* in hw1), called **IMG_OUT** with JPEG format.

Figure 3-1: Upside-down lena.bmp

The algorithm I use is to reflect the whole image over line $y = \frac{\text{height} - 1}{2}$ and the reflection of the point (x, y) is $(x, \text{height} - y - 1)$. Hence, all I need to do is assign the value of pixel $(x, \text{height-y-1})$ in original image to pixel (x, y) in new image for all (x, y).

```
16 for x in range(0, width):
17     for y in range(0, height):
18         pix_ud [x, y] = pix_ori[x, height-y-1]
```

Figure 3-2: Principal code fragment in upside-down.py

where `pix_ud` (`pix_ori`) is the image access object of the upside-down (original) image.

2. Right-side-left lena.bmp

Run `$ python3.5 right-side-left.py $IMG_IN $IMG_OUT`, and the program will generate an right-side-left image of **IMG_IN** (e.g., *lena.bmp* in hw1), called **IMG_OUT** with JPEG format.

Figure 4-1: Right-side-left lena.bmp

The algorithm I use is to reflect the whole image over line $x = \frac{\text{width} - 1}{2}$ and the reflection of the point (x, y) is $(\text{width} - x - 1, y)$. Hence, all I need to do is assign the value of pixel $(\text{width} - x - 1, y)$ in original image to pixel (x, y) in new image for all (x, y).

```
16 for x in range(0, width):
17     for y in range(0, height):
18         pix_rl [x, y] = pix_ori[width-x-1, y]
```

Figure 4-2: Principal code fragment in right-side-left.py

where `pix_rl` (`pix_ori`) is the image access object of the right-side-left (original) image.

3. Diagonally mirrored lena.bmp

Run `$ python3.5 mirror.py $IMG_IN $IMG_OUT`, and the program will generate an mirror image of **IMG_IN** (e.g., *lena.bmp* in hw1), called **IMG_OUT** with JPEG format.

Figure 5-1: Diagonally mirrored lena.bmp

The algorithm I use is to reflect the whole image over line $y = x$ and the reflection of the point (x, y) is (y, x). Hence, all I need to do is assign the value of pixel (y, x) in original image to pixel (x, y) in new image for all (x, y).

```
16 for x in range(0, width):
17     for y in range(0, height):
18         pix_mir [x, y] = pix_ori[y, x]
```

Figure 5-2: Principal code fragment in mirror.py

where `pix_mir` (`pix_ori`) is the image access object of the diagonally mirrored (original) image.

Part 2: Software

I use **GIMP**(*GNU image manipulation program*) in part 2.

1. Rotate lena.bmp 45 degrees clockwise

Figure 6: Rotate lena.bmp 45 degrees clockwise

To do this, first load the *lena.bmp* to canvas and click the tool icon

in the toolbox. Then set the argument *Angle* to 45 degrees.

2. Shrink lena.bmp in half

Figure 7: Shrink lena.bmp in half

To do this, first load the *lena.bmp* to canvas and click the tool icon

in the toolbox. Then set the arguments *Width* and *Height* to 256 and 256.

3. Binarize lena.bmp at 128 to get a binary image

Figure 8-1: Binarize lena.bmp at 128 to get a binary image

To do this, first load the *lena.bmp* to canvas and access the threshold tool from the image menu through *Colors* → *Threshold*. Then set the left argument (127) in the picture to 127.

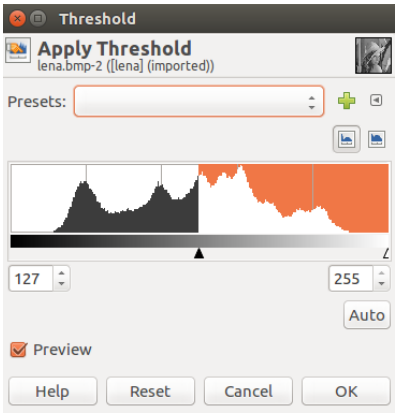

Figure 8-2: Threshold tool in GIMP