Computer Vision I

Homework 1 - Basic Image Manipulation B04902090 資工四 施長元

Usage of the full code:

`python main.py [Image_path] [Task]`

`Tasks: upside-down, right-side-left, diagonally-mirrored, 45-clockwise, shrink, binarize`

After the code exit, the output file will be in the same directory with the main.py

Environment: python3.6 on Windows Linux Subsystem (Ubuntu 16.04)

Contents:

1. Use B_PIX to write a program to generate

(a) upside-down lena.im

Sample usage: `python main.py lena.bmp upside-down`

```
if sys.argv[2] == "upside-down":
    for i in range(img.shape[0]):
        result[i] = img[img.shape[0] - i - 1]
```

Algorithm: Simply swap the lines vertically.

(b) right-side-left lena.im

Sample usage: `python main.py lena.bmp right-side-left`

```
elif sys.argv[2] == "right-side-left":
    for i in range(img.shape[1]):
        result[:, i] = img[:, img.shape[1] - i - 1]
```

Algorithm: Same as (a), but do it horizontally.

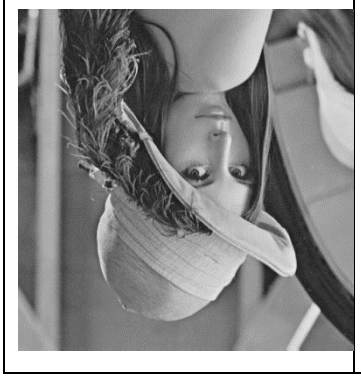
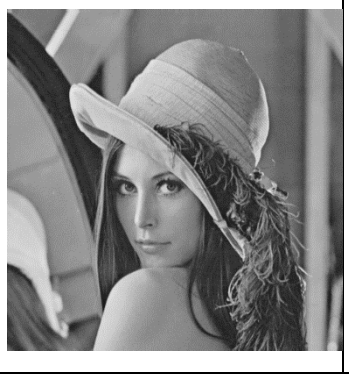
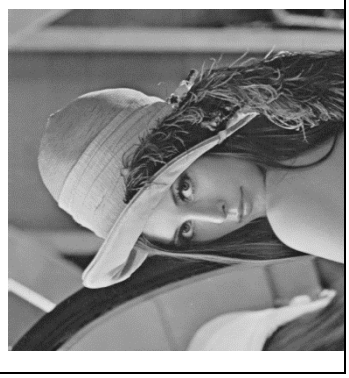
(c) diagonally mirrored lena.im

Sample usage: `python main.py lena.bmp diagonally-mirrored`

```
elif sys.argv[2] == "diagonally-mirrored":
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            result[i, j] = img[j, i]
```

Algorithm: Simply swap the x, y axis.

Results:

upside-down	right-side-left	diagonally-mirrored
		

2. Use Photoshop to

Note: I don't have Photoshop license, so I did these tasks on python.

(a) rotate lena.im 45 degrees clockwise

Sample usage: `python main.py lena.bmp 45-clockwise`

```
elif sys.argv[2] == "45-clockwise":
    result_pre = np.zeros((img.shape[0] * 2, img.shape[1] * 2))
    result = np.zeros((img.shape[0] * 2, img.shape[1] * 2))
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            result_pre[i + j, i - j] = img[j, i]
    result[:, 0 : int(result.shape[1] / 2)] = result_pre[:, int(result.shape[1] / 2) : result.shape[1]]
    result[:, int(result.shape[1] / 2) : result.shape[1]] = result_pre[:, 0 : int(result.shape[1] / 2)]

    result_f = np.zeros(img.shape)
    for i in range(result.shape[0]):
        for j in range(result.shape[1]):
            result_f[int(i/2), int(j/2)] += result[i, j]
    result = result_f / 2
```

Algorithm: Multiply the image with 45° transfer matrix (onto a 2x bigger plain image), then shrink the result image to the size identical to original image.
In this case, a square image will contain some null area. I set them to 0, which is black.

(b) shrink lena.im in half

Sample usage: `python main.py lena.bmp shrink`

```
elif sys.argv[2] == "shrink":
    result = np.zeros((int(img.shape[0] / 2), int(img.shape[1] / 2)))
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            result[int(i/2), int(j/2)] += img[i, j]
    result /= 4
```

Algorithm: Sum the pixels in a 4x4 square up, then divide them with 4 to represent a pixel.

(c) binarize lena.im at 128 to get a binary image

Sample usage: `python main.py lena.bmp binarize`

```
elif sys.argv[2] == "binarize":
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            result[i, j] = (255 if img[i, j] > 128 else 0)
```

Algorithm: Pixel by pixel, set it to 255 if the pixel is bigger than 128, otherwise to 0.

Results:

45-clockwise	shrink	binarize