Computer Vision HW01 旁聽生 陳又蕾

Brief description:
Used python, numpy, scipy.ndimage, and OpenCV to generate the following images.

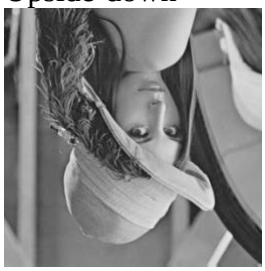
Principal Code Fragment:

```
def main():
    img = cv2.imread('lena.bmp', 0)
    cv2.imwrite('1a.bmp', upside_down(img))
    cv2.imwrite('1b.bmp', right_side_left(img))
    cv2.imwrite('1c.bmp', diagonally _ flip(img))
    cv2.imwrite('2d.bmp', rotate(img))
    cv2.imwrite('2e.bmp', shrink(img))
    cv2.imwrite('2f.bmp', binarize(img))
```

lena.bmp

1.

a. Upside-down



⇨ Algorithm: reverse the row of the pixels / x-coordinates of the input image

b. Right-side-left

⇨ reverse the columns of the pixels / y-coordinates of the input image

c. Diagonally flip

⇨ created a black image of same size and set each pixel so that its row and column is opporsite to the respective pixel of the original image.

2.

a. Rotate 45 degrees clockwise

⇨ used rotate(input, angle, reshape) from scipy.ndimage. The angle parameter is set to 45 degrees, allowing the image to rotate to that value.
* sets reshape to false, which maintains the image's original size, but the image would not be contained completely in the output

b. Shrink in half

⇨ used cv2 .resize(input, (width, height),)interpolation from opencv to resize the image. The width and height is set to ½ of the original image width and height. The shrunk image is then pasted on a black image (with all of its pixels set to 0) of the size of original image is created in order to display the differences in size more clearly.

c. Binarize at 128 to get a binary picture

⇨ looped through all the pixels of the original image and set those that have a value above 128 to 255 and those that have a value below 128 to 0 on the respective pixel of the image copy.