電腦視覺（一）Homework1

R11521701 程懷恩

1. Upside-Down

```
#upside-down
def UpsideDown():
    for i in range(0, img_height):
        for j in range(0, img_width):
            img_transfered[i][j]=img[img_height-i-1][j]
    cv2.imwrite('upside-down_lena.bmp',img_transfered)
```

程式碼：新建一個與原始相片相同的 numpy 空陣列，並由列倒著將原陣列像素寫入新陣列中。

2. Right-Side-Left

```
#right-side-left
def RightSideLeft():
    for i in range(0, img_height):
        for j in range(0, img_width):
            img_transfered[i][j]=img[i][img_width-j-1]
    cv2.imwrite('right-side-down_lena.bmp',img_transfered)
```

程式碼：新建一個與原始相片相同的 numpy 空陣列，並由行倒著將原陣列

像素寫入新陣列中。

3. Diagonally Flip

```
#diagonally flip
def DiagonallyFlip():
    for i in range(0, img_height):
        for j in range(0, img_width):
            img_transfered[i][j]=img[img_height-i-1][img_width-j-1]
    cv2.imwrite('diagonally-flip_lena.bmp',img_transfered)
```

程式碼：新建一個與原始相片相同的 numpy 空陣列，將原陣列每顆像素由

右下到左上倒著寫入新陣列中。

4. Rotate 45 degrees clockwise

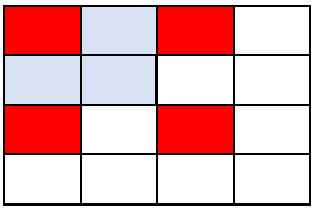
```python
#rotate
def rotate(img, angle , center=None, scale=1.0):
    img_height = img.shape[0]
    img_width = img.shape[1]

    if center is None:
        center = (img_height/2, img_width/2)

    #rotate
    M = cv2.getRotationMatrix2D(center,angle,scale)
    rotated = cv2.warpAffine(img, M, (img_height, img_width))

    return rotated

img_rotated = rotate(img, 360-45)
cv2.imwrite('rotated.bmp',img_rotated)
```

程式碼：此處先設定旋轉軸心（預設 default 為圖片中心點）以及縮放大小（預設也為 1），接著透過 getRotationMatrix2D( )設定中心與旋轉角度(以順時針為正)，再利用 warpAffine( )設定旋轉後圖片大小，此處設定與原圖相同。

5. Shrink in half

```python
# shrink(resize)
scale = 2
img_transfered= np.zeros((img.shape[0]//scale, img.shape[1]//scale, 3))

def Resizer(img, shape, scale):
    height = shape[0]
    width = shape[1]
    re_height = shape[0]//scale
    re_width = shape[1]//scale
    #img_transfered= np.zeros((img.shape[0]//scale, img.shape[1]//scale))

    for i in range(0,re_height):
        for j in range(0,re_width):
            for k in range (scale):
                img_transfered[i][j] += img[2*i+k][2*j]
                img_transfered[i][j] += img[2*i][2*j+k]

    #print(img_transfered[0][0])

Resizer(img, img.shape, 2)
img_transfered=img_transfered//4
#print(img_transfered[0][0]) check if resize success
cv2.imwrite('resized.bmp',img_transfered)
```

程式碼：先藉由 scale 確立縮放大小，透過前兩層迴圈找出所有像素，而第三個迴圈做以 2*2 範圍內的像素格相加，以下為示意圖：



找出原圖的紅色格子後將其與右、右下、下三格相加後，除以 4 取 RGB 的平均值，達到縮放的效果。

6. Binarize at 128 to get a binary image

```
#reload lena.bmp by grey scale

img = cv2.imread('lena.bmp',0)
def binarize():
    for i in range(0, img.shape[0]):
        for j in range(0, img.shape[1]):
            #print(img[i][j]) checker
            if(img[i][j]>128):
                img[i][j]=255
            else:
                img[i][j]=0

    cv2.imwrite('binarized.bmp',img)

binarize()
```

程式碼：將圖片重新以灰階讀取，每一個像素內階只有一個灰階 grayscale，

若其大於 128 則改為 255，反之則設為 0，達成二值處理。