**Computer Vision HW1: Basic Image Manipulation**

**R10741015 鄭傑鴻**

**Sept. 7, 2022**

**Original Picture:**

****

(With shape (512, 512))

**Part 1:**

1. Upside down

* Description:

For every element (pixel) in the new picture, denoted as , assign pixel of the original picture to it. (The height of the picture is 512, i & j from 0 to 511)

* Code:

def upsideDown(self, save=True):  
 pic\_new = np.zeros\_like(self.pic)  
 rownum = self.pic.shape[0]  
 colnum = self.pic.shape[1]  
 for i in range(rownum-1):  
 for j in range(colnum-1):  
 pic\_new[i][j] = self.pic[rownum-1-i][j]  
 showImg(pic\_new)  
 if save: cv2.imwrite('lena\_upsideDown.bmp', pic\_new)

* Resulting Image:



1. Right-side Left

* Description:

For every element (pixel) in the new picture, denoted as , assign pixel on the original picture to it. (The width of the picture is 512, i & j from 0 to 511)

* Code:

def rightsideLeft(self, save=True):  
 pic\_new = np.zeros\_like(self.pic)  
 rownum = self.pic.shape[0]  
 colnum = self.pic.shape[1]  
 for i in range(rownum-1):  
 for j in range(colnum-1):  
 pic\_new[i][j] = self.pic[i][colnum-1-j]  
 showImg(pic\_new)  
 if save: cv2.imwrite('lena\_rightsideLeft.bmp', pic\_new)

* Resulting Image:



1. Diagonally Flip

* Description:

For every element (pixel) in the new picture, denoted as , assign pixel on the original picture to it. (The width and height of the picture are both 512, i & j from 0 to 511)

* Code:

def diagonalFilp(self, save=True):  
 pic\_new = np.zeros\_like(self.pic)  
 rownum = self.pic.shape[0]  
 colnum = self.pic.shape[1]  
 for i in range(rownum-1):  
 for j in range(colnum-1):  
 pic\_new[i][j] = self.pic[rownum-1-i][colnum-1-j]  
 showImg(pic\_new)  
 if save: cv2.imwrite('lena\_diagonalFlip.bmp', pic\_new)

* Resulting Image:



**Part 2:**

1. Rotate 45 degrees clockwise

* Description:

Since part 2 has no restrictions regarding using libraries, I use the *imutils.rotate()* method, with a negative 45 degrees counter-clockwise.

* Code:

def rotate45(self, save=True):  
 pic\_new = imutils.rotate(self.pic, -45)  
 showImg(pic\_new)  
 if save: cv2.imwrite('lena\_rotate45.bmp', pic\_new)

* Resulting Image:



1. Shrink height and width in half

* Description:

Since part 2 has no restrictions regarding using libraries, I use the *cv2.resize()* method, with a new shape (256, 256).

* Code:

def shrinkHalf(self, save=True):  
 h\_new = int(self.pic.shape[0]/2)  
 w\_new = int(self.pic.shape[1]/2)  
 pic\_new = cv2.resize(self.pic, (h\_new, w\_new), cv2.INTER\_AREA)  
 showImg(pic\_new)  
 if save: cv2.imwrite('lena\_shrinkHalf.bmp', pic\_new)  
 print('Shape for the new picture: ', pic\_new.shape)

* Resulting Image:





1. Binarize at 128 to get a binary image

* Description:
* For every element (pixel) in the new picture, denoted as , if the correspondent pixel in the original picture is greater than 128, assign value 255 to , else assign 0 to . (Value 255 resulting in white pixel, and value 0 result in black pixel. i & j from 0 to 511)
* Code:

def binarize(self, save=True):  
 pic\_new = np.zeros\_like(self.pic)  
 rownum = self.pic.shape[0]  
 colnum = self.pic.shape[1]  
 for i in range(rownum-1):  
 pic\_new[i] = (self.pic[i] > 128)\*255  
 showImg(pic\_new)  
 if save: cv2.imwrite('lena\_binarize.bmp', pic\_new)

* Resulting Image:

