Computer Vision Homework #1 Report

1. Read RGB image:

```
if not(os.path.isdir(os.getcwd() + '/result_img')):
    os.makedirs(os.getcwd() + '/result_img')
img1 = cv2.imread('test_img/taipei101.png')
img2 = cv2.imread('test_img/aeroplane.png')
```

Read RGB image with cv2.imread() function.

2. Covert image to grayscale image:

```
img1_gray = getGrayImg(img1)
img2_gray = getGrayImg(img2)
```

```
def getGrayImg(img):
    img_gray = np.zeros((img.shape[0], img.shape[1], 1), np.uint8)
    for i in range(0, img.shape[0]):
        for j in range(0, img.shape[1]):
            img_gray[i, j] = 0.299 * img[i, j, 2] + 0.587 * img[i, j, 1] + 0.114 * img[i, j, 0]
    return img_gray
```

In the getGrayImg() function, Create a new matrix 'img_gray' to store the result. For each pixel, multiply the RGB values by 0.299, 0.587, and 0.114 respectively, and then add them together to save the result in the corresponding pixel position.

3. Convolution operation:

```
kernel = np.array([[[-1], [-1], [-1]],[[-1], [8], [-1]],[[-1], [-1], [-1]]])
```

First, create a numpy array as kernel for convolution operation in the main function.

```
img1_conv = convolution(img1_gray, kernel, 1)
img2_conv = convolution(img2_gray, kernel, 1)
```

Then, pass the grayscale image, kernel, and stride as parameters into the convolution() function.

```
def convolution(img, kernel, stride):
    kernel = np.rot90(kernel, 2)
    padding_size = kernel.shape[0]//2
    kernel_size = kernel.shape[0]
```

In the convolution() function, the kernel should be flipped vertically and horizontally first, and then calculate the padding size(Which is calculated in order

to maintain the original size) and kernel size for creating new matrix in the next step.

```
img_pad = np.pad(img, ((padding_size, padding_size), (padding_size, padding_size), (0, 0)), mode='constant')
img_conv = np.zeros(((img.shape[0] + 2 * padding_size - kernel_size)//stride + 1, (img.shape[1] + 2 * padding_size - kernel_size)//stride + 1), dtype=int)
```

Before convolution operation, create a padding image of the grayscale image, and a NumPy array ‘img_conv’ to store the result.

```
for i in range(0, img_conv.shape[0]):
    for j in range(0, img_conv.shape[1]):
        if(i * stride + kernel_size < img_pad.shape[0] and j * stride + kernel_size < img_pad.shape[1]):
            conv_mat = img_pad[i * stride : i * stride + kernel_size, j * stride : j * stride + kernel_size] * kernel
            img_conv[i, j] = conv_mat.sum()
```

Next, multiply each matrix in the padded image that has the same size as the kernel by the kernel, add them together, and store the result in the corresponding position of 'img_conv'.

```
return np.clip(img_conv, 0, 255).astype(np.uint8)
```

Return the image-type array with values clipped to the range of 0 to 255.

4. Pooling operation:

```
img1_pool = pooling(img1_conv, 2, 2)
img2_pool = pooling(img2_conv, 2, 2)
```

pass the convolution image, kernel size, and stride as parameters into the pooling() function.

```
def pooling(img, kernel_size, stride):
    img_pool = np.zeros(((img.shape[0] - kernel_size) // stride + 1, (img.shape[1] - kernel_size) // stride + 1, 1), np.uint8)
    for i in range(0, img_pool.shape[0]):
        for j in range(0, img_pool.shape[1]):
            img_pool[i, j] = np.max(img[i * stride : i * stride + kernel_size, j * stride : j * stride + kernel_size])
    return img_pool
```

In the pooling() function, create a image-type NumPy array as the returned data, and then select the maximun value from each matrix with a kernel size.

5. Binarization operation:

```
img1_bin = binarization(img1_pool)
img2_bin = binarization(img2_pool)
```

```
def binarization(img):
    return np.where(img >= 128, 255, 0).astype(np.uint8)
```

In the binarization() function, return the image-type array with values divided into two values: 255 and 0.

6. Save images:

```
cv2.imwrite('result_img/taipei101 _Q1.png', img1_gray)
cv2.imwrite('result_img/aeroplane _Q1.png', img2_gray)
```

```
cv2.imwrite('result_img/taipei101 _Q2.png', img1_conv)
cv2.imwrite('result_img/aeroplane _Q2.png', img2_conv)
```

```
cv2.imwrite('result_img/taipei101 _Q3.png', img1_pool)
cv2.imwrite('result_img/aeroplane _Q3.png', img2_pool)
```

```
cv2.imwrite('result_img/taipei101 _Q4.png', img1_bin)
cv2.imwrite('result_img/aeroplane _Q4.png', img2_bin)
```

Save images with cv2.imwrite() function.

7. Result

grayscale	convolution	pooling	binarization