结果报告

Task 1:

函数 **image_patches( )** 的具体代码如下：

```
def image_patches(image, patch_size=(16, 16)):
    """
    Given an input image and patch_size,
    return the corresponding image patches made
    by dividing up the image into patch_size sections.

    Input- image: H x W
           patch_size: a scalar tuple M, N
    Output- results: a list of images of size M x N
    """

    # TODO: Use slicing to complete the function
    cut_width = patch_size[0]
    cut_length = patch_size[1]
    (width, length) = image.shape

    pic = np.zeros((cut_width, cut_length))

    num_width = int(width / cut_width)
    num_length = int(length / cut_length)
    output = []

    for i in range(0, num_width):
        for j in range(0, num_length):
            pic = image[i * cut_width: (i + 1) * cut_width, j * cut_length: (j +
1) * cut_length]
            output.append(pic)

    return output
```

运行结果图如下：

Task 2:

函数 convolve( ) 的代码如下：

```
def convolve(image, kernel):
    """
    Return the convolution result: image * kernel.
    Reminder to implement convolution and not cross-correlation!
    Caution: Please use zero-padding.

    Input- image: H x W
```

```
            kernel: h x w
    Output- convolve: H x W
    """
    (H, W) = image.shape
    kernel = np.array(kernel)
    (h, w) = kernel.shape
    kernel = kernel[::-1, ...][..., ::-1]
    h_pad = (h - 1) // 2
    w_pad = (w - 1) // 2

    image = np.pad(image, pad_width=[(h_pad, h_pad), (w_pad, w_pad)],
mode="constant", constant_values=0)
    output = np.zeros(shape=(H, W))
    for i in range(H):
        for j in range(W):
            output[i, j] = np.sum(np.multiply(image[i: i + h, j: j + w],
kernel))

    return output
```

运行结果如下：

函数 edge_detection( ) 的代码如下：

```
def edge_detection(image):
    """
    Return Ix, Iy and the gradient magnitude of the input image

    Input- image: H x W
    Output- Ix, Iy, grad_magnitude: H x W
    """
    # TODO: Fix kx, ky
    kx = [[1], [0], [-1]]  # 1 x 3
    ky = [[1],
          [0],
```

```
            [-1]]  # 3 x 1

    Ix = convolve(image, kx)
    Iy = convolve(image, ky)

    # TODO: Use Ix, Iy to calculate grad_magnitude
    grad_magnitude = np.sqrt(np.square(Ix)+np.square(Iy))

    return Ix, Iy, grad_magnitude
```

运行结果如下：

q3_edge.png

q3_edge_gaussian.png

Task 3:

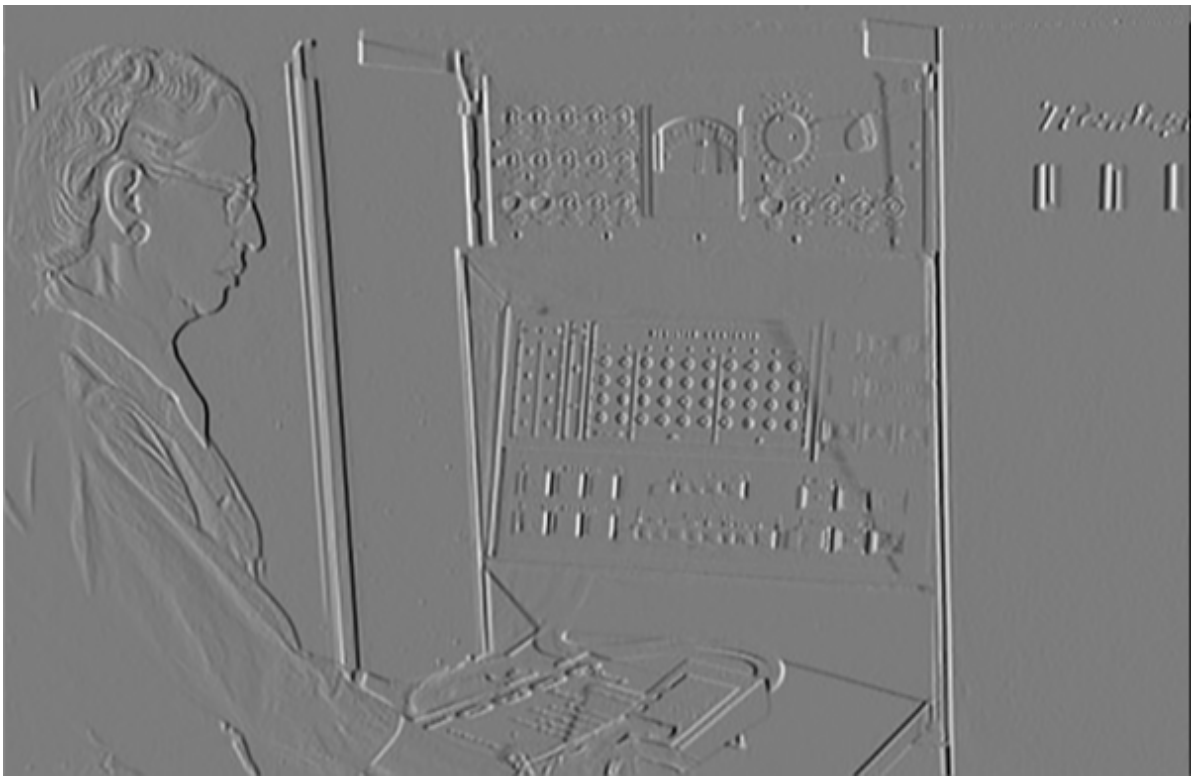
函数 sobel_operator( ) 的代码如下：

```
def sobel_operator(image):
    """
    Return Gx, Gy, and the gradient magnitude.

    Input- image: H x W
    Output- Gx, Gy, grad_magnitude: H x W
    """
    sx=[[1, 0, -1],
        [2, 0, -2],
        [1, 0, -1]]
    sy=[[1, 2, 1],
        [0, 0, 0],
        [-1, -2, -1]]

    # TODO: Use convolve() to complete the function
    Gx, Gy = convolve(image, sx), convolve(image, sy)
    grad_magnitude = np.sqrt(np.square(Gx)+np.square(Gy))

    return Gx, Gy, grad_magnitude
```

运行结果如下：

q2_edge_sobel.png

q2_Gx.png



q2_Gy.png