CV HW4

R11922196 林佑鑫

(a) Dilation

Brief discription, algorithm: iterate through 每個 pixel，並設定 h, w 的 range 避免 out of bound 問題。若任一 kernel 爲 1 的位置有 region 的值爲 255 則 output_img 的該 pixel 設定爲 255，以 np.sum(region * kernel)來判斷。

Parameters: None

Principal code fragment:

```python
def dilation(img, kernel):
    height, width = img.shape
    half_ks = kernel.shape[0] // 2

    output_img = np.zeros_like(img)
    for h in range(half_ks, height-half_ks):
        for w in range(half_ks, width-half_ks):
            region = img[h-half_ks:h+half_ks+1, w-half_ks:w+half_ks+1]

            if np.sum(region * kernel) > 0:
                output_img[h-half_ks, w-half_ks] = 255

    return output_img
```

Resulting image:

(b) Erosion

Brief discription, algorithm: iterate through 每個 pixel，並設定 h, w 的 range 避免 out of bound 問題。需所有 kernel 爲 1 的位置有 region 的值爲 255 則 output_img 的該 pixel 設定爲 255，以 np.all(region[kernel == 1] == 255)來判斷。

Parameters: None

Principal code fragment:

```python
def erosion(img, kernel):
    height, width = img.shape
    half_ks = kernel.shape[0] // 2

    output_img = np.zeros_like(img)
    for h in range(half_ks, height-half_ks):
        for w in range(half_ks, width-half_ks):
            region = img[h-half_ks:h+half_ks+1, w-half_ks:w+half_ks+1]

            if np.all(region[kernel == 1] == 255):
                output_img[h-half_ks, w-half_ks] = 255

    return output_img
```

Resulting image:

(c) Opening

Brief discription, algorithm: 對 binary_img 用上面 function 先做 erosion 再做 dilation。

Parameters: None

Principal code fragment:

```python
    # (c) Opening
    opening_img = dilation(erosion_img, octogonal_kernel)
    cv2.imwrite('c.png', opening_img)
```

Resulting image:

(d) Closing

Brief discription, algorithm: 對 binary_img 用上面 function 先做 dilation 再做 erosion。

Parameters: None

Principal code fragment:

```
84      # (d) Closing
85      closing_img = erosion(dilation_img, octogonal_kernel)
86      cv2.imwrite('d.png', closing_img)
```

Resulting image:

(e) Hit-and-miss transform

Brief discription, algorithm: iterate through 每個 pixel，並設定 h, w 的 range 避免 out of bound 問題。以 J_kernel 對 region、K_kernel 對 region complement 做 erosion、方法與上面一樣，兩者結果做 and，爲 True 的情況設定 output_img 爲該 pixel 爲 255。

Parameters: None

Principal code fragment:

```python
def hit_and_miss(img, J_kernel, K_kernel):
    height, width = img.shape
    half_ks = max(J_kernel.shape[0] // 2, K_kernel.shape[0] // 2)

    output_img = np.zeros_like(img)

    for h in range(half_ks, height-half_ks):
        for w in range(half_ks, width-half_ks):
            region = img[h-half_ks:h+half_ks+1, w-half_ks:w+half_ks+1]

            if np.all(region[J_kernel == 1] == 255) and np.all(~region[K_kernel == 1] == 255):
                output_img[h, w] = 255

    return output_img
```

Resulting image: