HW 4: Binary Morphology

Source Code

All questions are written in Python code, please refer to the file “main.py”.
All images will be stored in the folder “res” (automatically create a new folder).
In accordance with the **FAQ** of course website:

- All parts of the question are written from scratch, except for plotting images

Answer

1. Dilation
   Algorithm:
   1) Iterate binary image
   2) For every pixel of a binary image, compare it with the kernel. If kernel pixel is 1, make binary image pixel to 255

```
def dilationFunc(kernel, original):
    height, width = original.shape
    ycenter = int(kernel.shape[0] / 2)
    xcenter = int(kernel.shape[1] / 2)
    dilation = original.copy()
    for i in range(height):
        for j in range(width):
            if original[i][j] == 255:
                for x in range(kernel.shape[0]):
                    for y in range(kernel.shape[1]):
                        xdest = i + x - ycenter
                        ydest = j + y - xcenter
                        if (0 <= xdest < height) and (0 <= ydest <
width) and (kernel[x][y] == 1):
                            dilation[xdest][ydest] = 255
    return dilation
```

2. Erosion

   Algorithm:

   1) Iterate binary image
   2) Check if kernel matches exactly the same as surroundings of binary image
   3) If kernel matches, make pixel 255, otherwise 0

```
def erosionFunc(kernel, original, ycenter, xcenter):
    height, width = original.shape
    erosion = original.copy()
    for i in range(height):
        for j in range(width):
            matchFlag = True
            for x in range(kernel.shape[0]):
                for y in range(kernel.shape[1]):
                    if kernel[x][y] == 1:
                        xdest = i + x - ycenter
                        ydest = j + y - xcenter
                        if (0 <= xdest < height) and (0 <=
ydest < width):
                            if original[xdest][ydest] == 0:
                                matchFlag = False
                                break
                        else:
                            matchFlag = False
                            break
            if matchFlag:
                erosion[i][j] = 255
            else:
                erosion[i][j] = 0
```

```
    return erosion
```

3. Opening

   Algorithm:

   1) Do dilation algorithm from erosion image

```
opening = dilationFunc(kernel, erosion)
cv2.imwrite("res/opening.bmp", opening)
```

4. Closing

   Algorithm:

   1) Do erosion algorithm from dilation image

```
closing = erosionFunc(kernel, dilation, ycenter, xcenter)
cv2.imwrite("res/closing.bmp", closing)
```

5. Hit-and-miss transform

   Algorithm:

   1) Make new L-shaped kernel
   2) Do erosion with binary image with kernel origin (1, 0)
   3) Do erosion with complement of binary image with kernel origin (0, 1)
   4) Find intersection of 2 erosion images

```
def complementFunc(original):
    height, width = original.shape
    comp = original.copy()
    for i in range(height):
        for j in range(width):
            if comp[i][j] == 0:
                comp[i][j] = 255
            else:
                comp[i][j] = 0
    return comp

def intersectFunc(img1, img2):
    height, width = img1.shape
```

```
    intersect = img1.copy()
    for i in range(height):
        for j in range(width):
            if img1[i][j] != img2[i][j]:
                intersect[i][j] = 0
    return intersect

kernel_j = np.array([
    [0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0],
    [1, 1, 0, 0, 0],
    [0, 1, 0, 0, 0],
    [0, 0, 0, 0, 0],
])
kernel_k = np.array([
    [0, 0, 0, 0, 0],
    [0, 1, 1, 0, 0],
    [0, 0, 1, 0, 0],
    [0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0],
])
erosion1 = erosionFunc(kernel_j, binary, 2, 1)
comp = complementFunc(binary)
erosion2 = erosionFunc(kernel_k, comp, 2, 1)
hitmiss = intersectFunc(erosion1, erosion2)
cv2.imwrite("res/hitmiss.bmp", hitmiss)
```