Homework Report for Computer Vision

Yu Xiang, Luo

October 10, 2023

You can check this github for more information

(a) Dilation

```
if kernel[x, y]:
    pixel_x, pixel_y = a + x - centerKernel[0], b + y - centerKernel[1]
    if ((0 <= pixel_x < image.size[0]) and (0 <= pixel_y < image.size[1])):
        returnImage.putpixel((pixel_x, pixel_y), 1)
```

In this code, `a, b` means the current coordinate, and we put the pixel to 1 if there's any point is 1 in its kernel space.

(b) Erosion

```
if kernel[x, y]:
    pixel_x, pixel_y = a + x - centerKernel[0], b + y - centerKernel[1]
    if ((0 <= pixel_x < image.size[0]) and (0 <= pixel_y < image.size[1])):
        if not image.getpixel((pixel_x, pixel_y)):
            flag = False
            break
```

In this code, flag is true only if all the pixel in kernel space are 1. If `flag` is `True`, the pixel of `(a, b)` would set to 1.

(c) Opening

(d) Closing

```
def opening(image, kernel):
    return dilation(erosion(image, kernel, centerKernel), kernel)

def closing(image, kernel):
    return erosion(dilation(image, kernel), kernel, centerKernel)
```


Opening and closing are simply the combinated usage of dilation and erosion.

(e) Hit-and-miss

```
def hitmiss(image, kernelJ, centerkernelJ, kernelK, centerkernelK):
    return intersection(erosion(complement(image), kernelK, centerkernelK), erosion(image, kernelJ, centerkernelJ))
```

Details of intersection and complement can be found at the source code.