Hong Trinh – 438443
Embedded Vision

Assignment 3

1. **Thresholding**

The simple way to make binary images is using the thresholding operation. We start with a gray scale image and a threshold value. If each pixel value of the gray scale image is smaller than the threshold value, then we assign to it the value 0 (black). Otherwise, assign to it the value 255 (white).

The algorithm for binary thresholding corresponds to the following: for each pixel of the image, if the value is less than a given threshold, it is set to zero. Otherwise, it is set to user defined value.

This is the simple way to make binary image. Using the gray scale image so pixel is between 0 and 255.

```
7  import cv2 as cv
8  import numpy as np
9
10
11 #read the image
12 img = cv.imread('gray.jpeg')
13 #Using function to convert image to black and white image
14 (thresh, im_bw) = cv.threshold(img, 127, 255, cv.THRESH_BINARY)
15 cv.imwrite('BinaryImage.jpg', im_bw)
16 #cv.imshow('Binary image', im_bw)
17 #cv.waitKey()
18 #cv.destroyAllWindows()
```

The first input of the function is the gray scale image which we want to convert. The threshold value is the second input, taking the value 127 which is the middle of the scale of the values in gray scale (from 0 to 255). The third value is the value that we want to convert when a pixel value is greater than the threshold value. We use the white value (255). Recall that we want to convert black and white. At the end we want a image having two value 0 and 255. The last input is the type of threshold we want to apply, so we pass THRESH_BINARY.

The output is a tuple with the first value can be ignored. The second value is the result image after applying the operation.

2. **Opening and closing**

Dilation operation adds pixels to the boundaries of the object in an image. Useful in joining broken parts of an object.

- A kernel is convolved with the image
- A pixel in original image will be 1 if at least one pixel under the kernel is 1
- Increase the white region in the image or the size of the foreground object increases

Erosion operation removes the pixels from the object boundaries. Erosion uses for removing small white noises and detach two connected objects:

- A kernel is convolved with the image
- A pixel in original image will be 1 only if all the pixels under the kernel are 1, otherwise, it is eroded (made to 0)
- All the pixels near the boundary will be discarded depending upon the size of the kernel
- So, the thickness or size of the foreground object decreases, or the white region decreases in the image

Each operation applies a structuring element to an input image and generate an output image.

Opening and Closing are combining dilation and erosion. **Opening** is the result of erosion then dilation, otherwise **closing** is the result of erosion after applying dilation operation.

Creating function erosion and dilation:

SE and SED are the kernel which image is convolved. It is the structuring element which is applied to original image.

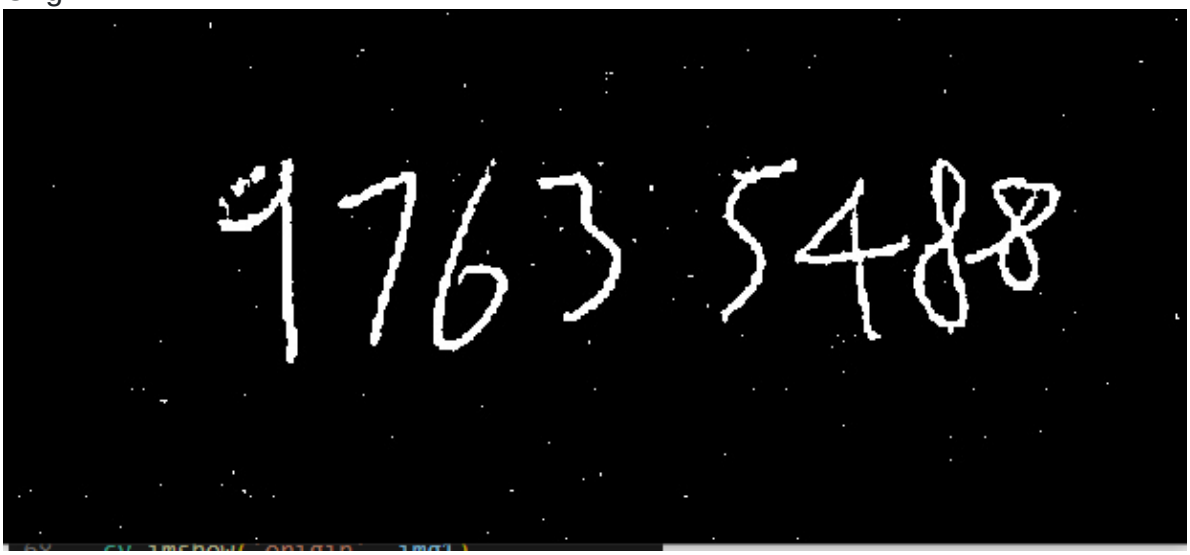
```
#Read the image for erosion
img1= cv.imread('mor.jpg',0)

def erosion(image):
    #Acquire size of the image
    m,n = image.shape
    # Define the structuring element
    SE= np.ones((5,5), dtype=np.uint8)
    constant= (5-1)//2

    #Define new image
    imgErode= np.zeros((m,n), dtype=np.uint8)
    #Erosion for morphology
    for i in range(constant, m-constant):
      for j in range(constant,n-constant):
        temp= image[i-constant:i+constant+1, j-constant:j+constant+1]
        product= temp*SE
        imgErode[i,j]= np.min(product)
    return imgErode
```

```
40  def dilation(image):
41      #Acquire size of the image
42      m,n = image.shape
43      #Define new image to store the pixels of dilated image
44      imgDilate= np.zeros((m,n), dtype=np.uint8)
45      #Define the structuring element
46      SED= np.array([[0,1,0], [1,1,1],[0,1,0]])
47      constant1=1
48      #Dilation operation without using inbuilt CV2 function
49      for i in range(constant1, m-constant1):
50        for j in range(constant1,n-constant1):
51          temp= image[i-constant1:i+constant1+1, j-constant1:j+constant1+1]
52          product= temp*SED
53          imgDilate[i,j]= np.max(product)
54      return imgDilate
```

Functions opening and closing call two operation erosion and dilation. Then call the functions with original image as parameter.

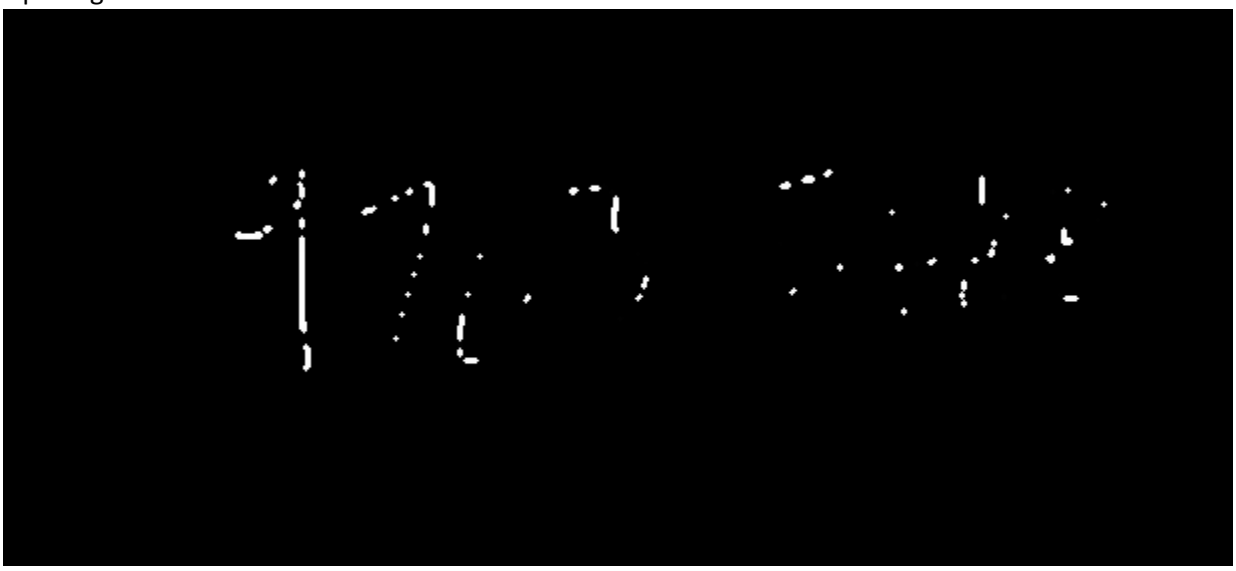
```
56  def opening(image):
57      o1 = erosion(image)
58      o2 = dilation(o1)
59      return o2
60
61  def closing(image):
62      c1 = dilation(image)
63      c2 = erosion(c1)
64      return c2
65
66  final_o = opening(img1)
67  final_c = closing(img1)
68  cv.imshow('origin', img1)
69  cv.imshow('opening', final_o)
70  cv.imshow('closing', final_c)
71  cv.waitKey()
72  cv.destroyAllWindows()
73
```

Origin:

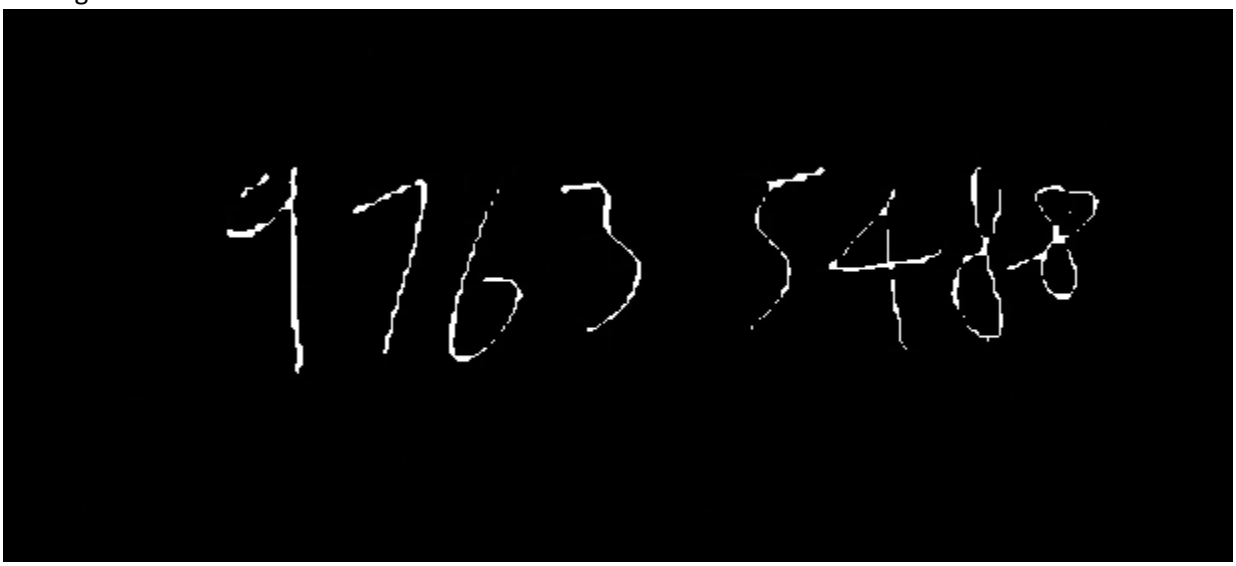


Hong Trinh – 438443
Embedded Vision

Opening:



Closing:



Reference:

Python OpenCV: Converting an image to black and white - techtutorialsx