

```
1 # 24-678 Computer Vision for Engineers
2 # Ryan Wu (ID:weihuanw)
3 # PS05 Binary image processing - detecting blobs,
  contours, and central axes
4 # Due 11/3/2023 (Fri) 5 pm
5
6 # import necessary packages
7 import cv2
8 import numpy as np
9
10 # function for loading given images
11 def load_images(image_path1, image_path2):
12     wall1_image = cv2.imread(image_path1, cv2.
  IMREAD_GRAYSCALE)
13     wall2_image = cv2.imread(image_path2, cv2.
  IMREAD_GRAYSCALE)
14     return wall1_image, wall2_image
15
16 # function for image dilation and erosion
17 def dilation_erosion(image, iterations=3):
18     kernel = cv2.getStructuringElement(cv2.
  MORPH_CROSS, (3, 3))
19     erode_image = cv2.erode(image, kernel,
  iterations=iterations)
20     dilate_image= cv2.dilate(erode_image, kernel,
  iterations=iterations)
21     return dilate_image
22
23 # function for drawing contours
24 def blob_contours(image):
25     # inverting the image
26     inverted_image = cv2.bitwise_not(image)
27     # funding contours using the inverted image
28     contours, _ = cv2.findContours(inverted_image,
  cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
29     # converting the image back to BGR
30     contour_image = cv2.cvtColor(inverted_image,
  cv2.COLOR_GRAY2BGR)
31     # drawing contours on the image
32     for contour in contours:
33         color = tuple(np.random.randint(0, 255, 3)).
```

```

33 tolist())
34         cv2.drawContours(contour_image, [contour
35 ], -1, color, 2)
36         # converting the image back
37         contour_image_color = cv2.bitwise_not(
38         contour_image)
39         # returning the image with contours
40         return contour_image_color
41
42 # function for detecting cracks
43 def detect_crack_and_thin(image,
44     contour_length_threshold = 2000):
45     # inverting the image
46     inverted_image = cv2.bitwise_not(image)
47     crack_image = image.copy()
48     # Finding contours
49     contours, _ = cv2.findContours(inverted_image,
50     cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
51     contour_crack = []
52     # Filtering contours
53     for contour in contours:
54         # calculate contour arc length
55         arc_length = cv2.arcLength(contour, True)
56
57         # filter contours based on area
58         if arc_length <= contour_length_threshold:
59             contour_crack.append(contour)
60
61     cv2.drawContours(crack_image, contour_crack, -1
62     , (255, 255, 255), 10)
63
64     # calling the thinning function
65     thinned_crack_image = thinning(crack_image)
66
67     return thinned_crack_image
68
69 # function for thinning
70 def thinning(image):
71     # reverse image (black background)
72     black_image = cv2.bitwise_not(image)
73     # Kernel: 4 neighbor

```

```

69     k_e = cv2.getStructuringElement(cv2.
    MORPH_CROSS, (3, 3))
70     # Target image
71     thin = np.zeros(black_image.shape, dtype=np.
    uint8)
72     # repeat until no white area
73     while cv2.countNonZero(black_image) != 0:
74         er = cv2.erode(black_image, k_e)
75         # OPEN: erosion then dilation (remove
    noise)
76         op = cv2.morphologyEx(er, cv2.MORPH_OPEN,
    k_e)
77         subset = cv2.subtract(er, op)
78         thin = cv2.bitwise_or(subset, thin)
79         black_image = er.copy()
80
81     # invert the thinned image back to white
    background
82     thinned_image = cv2.bitwise_not(thin)
83     return thinned_image
84
85 # main function
86 def main():
87     # calling load_images function
88     wall1_image, wall2_image = load_images('wall1.
    png', 'wall2.png')
89     if wall1_image is not None and wall2_image is
    not None:
90         # calling dilation_erosion function
91         wall1_blobed = dilation_erosion(
    wall1_image)
92         wall2_blobed = dilation_erosion(
    wall2_image)
93
94         # calling blob_contours function
95         wall1_contoured = blob_contours(
    wall1_blobed)
96         wall2_contoured = blob_contours(
    wall2_blobed)
97
98     # calling detect_crack function (setting

```

```
98 contour length threshold to 2000 for wall1 and 500  
   for wall2)  
99         wall1_thinned_crack_image =  
detect_crack_and_thin(wall1_blobed,  
contour_length_threshold=2000)  
100        wall2_thinned_crack_image =  
detect_crack_and_thin(wall2_blobed,  
contour_length_threshold=500)  
101  
102        # display and save images  
103        # cv2.imshow('Wall 1 Image Blobs',  
wall1_blobed)  
104        # cv2.imshow('Wall 2 Image Blobs',  
wall2_blobed)  
105        cv2.imwrite("wall1-blobs.png",  
wall1_blobed)  
106        cv2.imwrite("wall2-blobs.png",  
wall1_blobed)  
107  
108        # cv2.imshow('Wall 1 Image Contours',  
wall1_contoured)  
109        # cv2.imshow('Wall 2 Image Contours',  
wall2_contoured)  
110        cv2.imwrite("wall1-contours.png",  
wall1_contoured)  
111        cv2.imwrite("wall2-contours.png",  
wall2_contoured)  
112  
113        # cv2.imshow('Wall 1 Crack Image',  
wall1_thinned_crack_image)  
114        # cv2.imshow('Wall 2 Crack Image',  
wall2_thinned_crack_image)  
115        cv2.imwrite("wall1-cracks.png",  
wall1_thinned_crack_image)  
116        cv2.imwrite("wall2-cracks.png",  
wall2_thinned_crack_image)  
117  
118        cv2.waitKey(0)  
119        cv2.destroyAllWindows()  
120    else:  
121        print("Error in loading wall images.")
```

```
122
123 # if __name__ == "__main__":
124 main()
125
```