

```

7  def dilation():
8      img = cv2.imread("Lena.bmp")
9      height=img.shape[0]
10     width=img.shape[1]
11     gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
12     cv2.imwrite("gray.bmp", gray)
13     #二值化
14     binary=cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
15     for i in range(height):
16         for j in range(width):
17             if binary[i,j]<128:
18                 binary[i,j]=0
19             elif binary[i,j]>=128:
20                 binary[i,j]=255
21     cv2.imwrite("binary.bmp", binary)
22     #dilation
23     binary.flags.writeable = True
24     kernel=np.array([[0,1,1,1,0],
25                     [1,1,1,1,1],
26                     [1,1,1,1,1],
27                     [1,1,1,1,1],
28                     [0,1,1,1,0]])
29     dilation=np.zeros((height,width),dtype=int)
30     for i in range(height):
31         for j in range(width):
32             if binary[i,j]<128:
33                 binary[i,j]=0
34             elif binary[i,j]>=128:
35                 binary[i,j]=1
36     for i in range(height):
37         for j in range(width):
38             if(binary[i][j]==1):
39                 for x in range(-2,3,1):
40                     for y in range(-2,3,1):
41                         if (0<=x+i and x+i<height and 0<=y+j and y+j<width):
42                             if kernel[x+2][y+2]==1:
43                                 dilation[i+x][y+j]=255
44     dilation_img=Image.fromarray(dilation)
45     dilation_img=dilation_img.convert("L")
46     #dilation_img.show()
47     dilation_img.save("dilation.bmp")
48
49
50

```



dilation

```

53 #erosion
54 img = cv2.imread("lena.bmp")
55 height=img.shape[0]
56 width=img.shape[1]
57 gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
58 cv2.imwrite("gray.bmp", gray)
59 #二值化
60 binary=cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
61 for i in range(height):
62     for j in range(width):
63         if binary[i,j]<128:
64             binary[i,j]=0
65         elif binary[i,j]>=128:
66             binary[i,j]=255
67 cv2.imwrite("binary.bmp", binary)
68 binary.flags.writeable = True
69
70 for i in range(height):
71     for j in range(width):
72         if binary[i,j]<128:
73             binary[i,j]=0
74         elif binary[i,j]>=128:
75             binary[i,j]=1
76 #print(binary[1][1])
77 kernel=np.array([[0,1,1,1,0],
78                 [1,1,1,1,1],
79                 [1,1,1,1,1],
80                 [1,1,1,1,1],
81                 [0,1,1,1,0]])
82 kernel_sum=0
83 for i in range(kernel.shape[0]):
84     for j in range(kernel.shape[1]):
85         kernel_sum=kernel_sum+kernel[i][j]
86 erosion=np.zeros((height,width),dtype=int)
87 for i in range(height):
88     for j in range(width):
89         count=0
90         for x in range(-2,3,1):
91             for y in range(-2,3,1):
92                 if (0<=x+i and x+i<height and 0<=y+j and y+j<width):
93                     count=count+(binary[i+x][j+y]*kernel[x+2][y+2])
94             if count==kernel_sum:
95                 erosion[i][j]=255
96 erosion_img=Image.fromarray(erosion)
97 erosion_img=erosion_img.convert("L")
98 erosion_img.save("erosion.bmp")
99

```



erosion



Opening

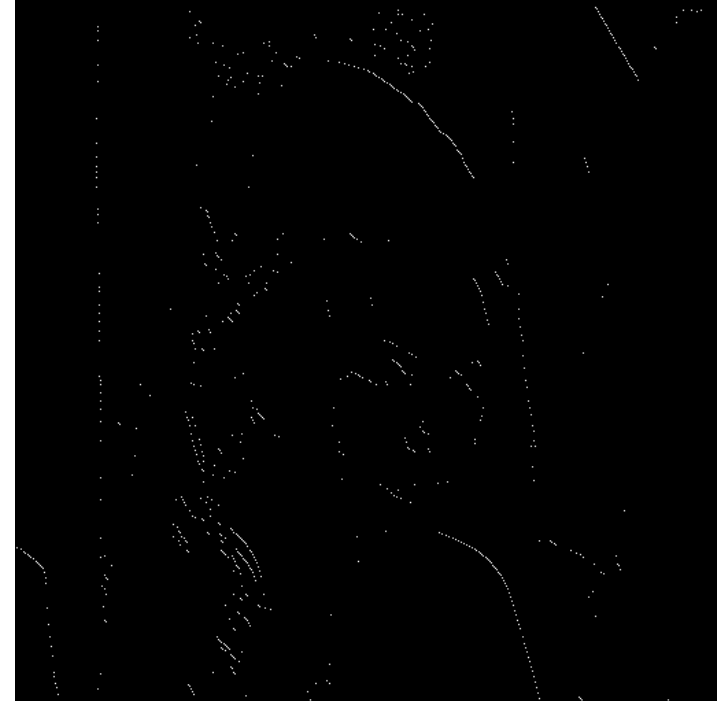


closing

```

264 #hit and miss
265 img = cv2.imread("lena.bmp")
266 height=img.shape[0]
267 width=img.shape[1]
268 gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
269 cv2.imwrite("gray.bmp", gray)
270 a=0
271 binary=cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
272 for i in range(height):
273     for j in range(width):
274         if binary[i,j]<128:
275             binary[i,j]=0
276         elif binary[i,j]>=128:
277             binary[i,j]=255
278 cv2.imwrite("binary.bmp", binary)
279 binary.flags.writeable = True
280 for i in range(height):
281     for j in range(width):
282         if binary[i,j]<128:
283             binary[i,j]=0
284         elif binary[i,j]>=128:
285             binary[i,j]=1
286 #B(binary) inverse
287 binary_c=cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
288 for i in range(height):
289     for j in range(width):
290         if binary_c[i,j]<128:
291             binary_c[i,j]=0
292         elif binary_c[i,j]>=128:
293             binary_c[i,j]=255
294 binary_c.flags.writeable = True
295 for i in range(height):
296     for j in range(width):
297         if binary_c[i,j]<128:
298             binary_c[i,j]=0
299         elif binary_c[i,j]>=128:
300             binary_c[i,j]=1
301 for i in range(height):
302     for j in range(width):
303         if binary_c[i,j]==1:
304             binary_c[i,j]=0
305         elif binary_c[i,j]==0:
306             binary_c[i,j]=1
307 cv2.imwrite("binary_c.bmp", binary_c)
308
309 J_kernel=np.array([[0,0,0],
310                    [1,1,0],
311                    [0,1,0]])
312 J_kernel_sum=0
313 for i in range(J_kernel.shape[0]):
314     for j in range(J_kernel.shape[1]):
315         J_kernel_sum=J_kernel_sum+J_kernel[i][j]
316 K_kernel=np.array([[0,1,1],
317                    [0,0,1],
318                    [0,0,0]])
319 K_kernel_sum=0
320 for i in range(K_kernel.shape[0]):
321     for j in range(K_kernel.shape[1]):
322         K_kernel_sum=K_kernel_sum+K_kernel[i][j]
323
324 A_erosion_J=np.zeros((height,width),dtype=int)
325 Ac_erosion_K=np.zeros((height,width),dtype=int)
326 hit_and_miss=np.zeros((height,width),dtype=int)
327
328 for i in range(height):
329     for j in range(width):
330         count=0
331         for x in range(-1,2,1):
332             for y in range(-1,2,1):
333                 if (0<=x+1 and x+1<=height and 0<=y+1 and y+1<=width):
334                     count=count+(binary[i+x][j+y]*J_kernel[x+1][y+1])
335                 if count==J_kernel_sum:
336                     A_erosion_J[i][j]=255
337
338 for i in range(height):
339     for j in range(width):
340         count=0
341         for x in range(-1,2,1):
342             for y in range(-1,2,1):
343                 if (0<=x+1 and x+1<=height and 0<=y+1 and y+1<=width):
344                     count=count+(binary_c[i+x][j+y]*K_kernel[x+1][y+1])
345                 if count==K_kernel_sum:
346                     Ac_erosion_K[i][j]=255
347
348 for i in range(height):
349     for j in range(width):
350         if A_erosion_J[i][j]==Ac_erosion_K[i][j]==255:
351             hit_and_miss[i][j]=255
352         else:
353             hit_and_miss[i][j]=0
354 hit_and_miss_img=Image.fromarray(hit_and_miss)
355 hit_and_miss_img=hit_and_miss_img.convert('L')
356 hit_and_miss_img.save("hit_and_miss.bmp")
357

```



Hit and miss