Homework 7

R13525009 羅筠笙

1. Yokoi Connectivity Number

Description: This program processes a binarized Lena image to perform iterative thinning. It begins by downsampling the image to a 64x64 grid using the top-left pixel from each 8x8 block. Then, it calculates the Yokoi connectivity number for each pixel to determine its structural importance and marks candidate pixels for removal. The thinning process removes marked pixels iteratively while preserving the overall shape. Once the thinning process stabilizes, the final result is upsampled back to the original 512x512 resolution for clearer visualization.



1. from PIL import Image as im
2. import numpy as np
3. # Load the image
4. img = im.open('lena.bmp')
5. img\_arr = np.asarray(img)
6. img\_arr\_binary = (img\_arr >= 128).astype(np.uint8) \* 255
7. # Downsample the image
8. def downsample(arr):
9. m, n = arr.shape
10. result = np.zeros((m // 8, n // 8), dtype=arr.dtype)
11. for i in range(m // 8):
12. for j in range(n // 8):
13. result[i, j] = arr[i \* 8, j \* 8]
14. return result
15. # Upsample the image back to 512x512
16. def upsample(arr):
17. m, n = arr.shape
18. result = np.zeros((m \* 8, n \* 8), dtype=arr.dtype)
19. for i in range(m \* 8):
20. for j in range(n \* 8):
21. result[i, j] = arr[i // 8, j // 8]
22. return result
23. # Expand the image array with a border of zeros for boundary conditions
24. def expand\_with\_zeros(arr):
25. m, n = arr.shape
26. result = np.zeros((m + 2, n + 2), dtype=arr.dtype)
27. for i in range(m):
28. for j in range(n):
29. result[i + 1, j + 1] = arr[i, j] # Put the original pixel to the center
30. return result
31. # Function for computing Yokoi connectivity number
32. def h(b, c, d, e):
33. if b != c:
34. return 's'
35. else:
36. if d == b and e == b:
37. return 'r'
38. else:
39. return 'q'
40. def f(a1, a2, a3, a4):
41. if a1 == 'r' and a2 == 'r' and a3 == 'r' and a4 == 'r':
42. return 5
43. count = sum(1 for a in [a1, a2, a3, a4] if a == 'q')
44. return count
45. def yokoi(arr):
46. m, n = arr.shape
47. result = [[0] \* n for \_ in range(m)]
48. arr = expand\_with\_zeros(arr)
49. for i in range(m):
50. for j in range(n):
51. a1 = h(arr[i+1][j+1], arr[i+1][j+2], arr[i][j+2], arr[i][j+1])
52. a2 = h(arr[i+1][j+1], arr[i][j+1], arr[i][j], arr[i+1][j])
53. a3 = h(arr[i+1][j+1], arr[i+1][j], arr[i+2][j], arr[i+2][j+1])
54. a4 = h(arr[i+1][j+1], arr[i+2][j+1], arr[i+2][j+2], arr[i+1][j+2])
55. tmp = f(a1, a2, a3, a4)
56. if tmp and arr[i+1][j+1]:
57. result[i][j] = tmp
58. return np.array(result)
59. # Mark pairs in the Yokoi array for thinning
60. def mark\_pairs(yokoi\_arr):
61. m, n = len(yokoi\_arr), len(yokoi\_arr[0])
62. result = [[0] \* n for \_ in range(m)]
63. yokoi\_arr = expand\_with\_zeros(yokoi\_arr)
64. for i in range(m):
65. for j in range(n):
66. if yokoi\_arr[i+1][j+1] == 1:
67. count = sum(1 for x, y in [(i, j+1), (i+1, j), (i+2, j+1), (i+1, j+2)] if yokoi\_arr[x][y] == 1)
68. if count >= 1:
69. result[i][j] = 1
70. return np.array(result)
71. # Helper function for thinning
72. def h2(b, c, d, e):
73. return 1 if b == c and (b != d or b != e) else 0
74. def f2(a1, a2, a3, a4, x):
75. return 0 if a1 + a2 + a3 + a4 == 1 else x
76. # Perform the thinning process on the image
77. def thinning\_process(original, marked\_arr):
78. m, n = original.shape
79. original = expand\_with\_zeros(original)
80. modified = 0 # Track if changes occur
81. for i in range(m):
82. for j in range(n):
83. if marked\_arr[i][j]:
84. a1 = h2(original[i+1][j+1], original[i+1][j+2], original[i][j+2], original[i][j+1])
85. a2 = h2(original[i+1][j+1], original[i][j+1], original[i][j], original[i+1][j])
86. a3 = h2(original[i+1][j+1], original[i+1][j], original[i+2][j], original[i+2][j+1])
87. a4 = h2(original[i+1][j+1], original[i+2][j+1], original[i+2][j+2], original[i+1][j+2])
88. temp = f2(a1, a2, a3, a4, original[i+1][j+1])
89. if temp != original[i+1][j+1]:
90. modified = 1
91. original[i+1][j+1] = temp
92. # Remove the zero-padded border
93. thinned\_result = [[original[i+1][j+1] for j in range(n)] for i in range(m)]
94. return np.array(thinned\_result), modified
95. # Downsample, apply thinning, and upsample for visualization
96. downsampled = downsample(img\_arr\_binary)
97. change = 1
98. # Update the image until the last output never changed.
99. while change:
100. yokoi\_arr = yokoi(downsampled)
101. marked\_arr = mark\_pairs(yokoi\_arr)
102. downsampled, change = thinning\_process(downsampled, marked\_arr)
103. # # Upsample for clearer visualization
104. final\_result = upsample(downsampled)