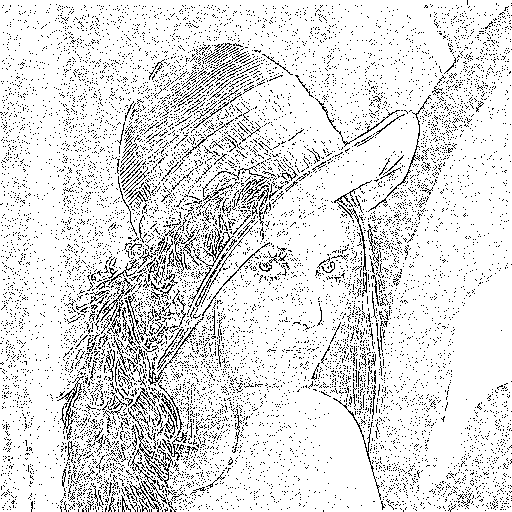
Homework 10

R13525009 羅筠笙

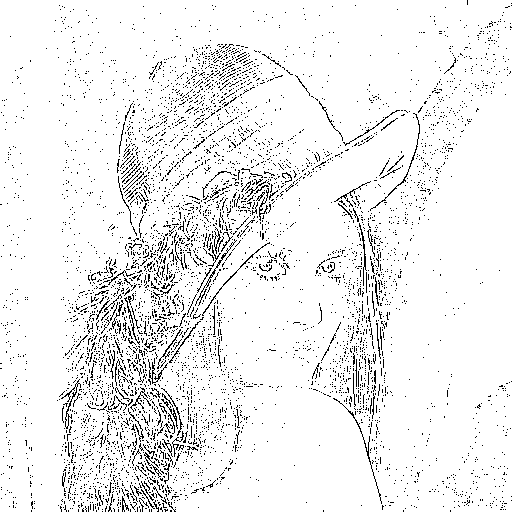
1. **Laplace Mask1 (0, 1, 0, 1, -4, 1, 0, 1, 0): 15**

Description: Applies a simple Laplace operator for edge detection and further filters the edge points based on neighborhood pixel characteristics, producing the final edge-detected image.



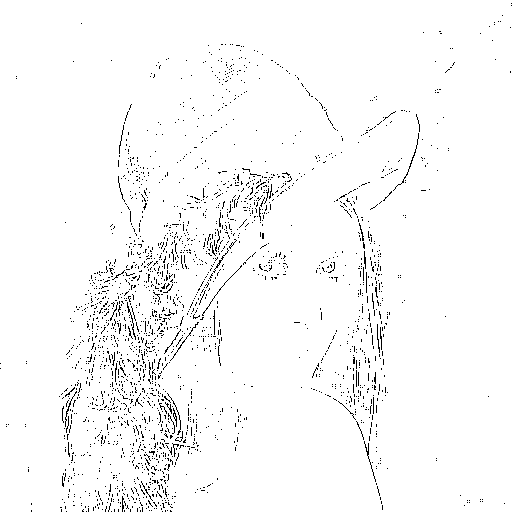
1. def laplace1(img\_arr, threshold):
2. mask = np.array([[0, 1, 0],
3. [1, -4, 1],
4. [0, 1, 0]])
5. # Expand the image
6. img\_arr = expand\_with\_replicate(img\_arr, 1)
7. # Using convolution
8. res1 = my\_conv(img\_arr, mask, threshold)
9. res1 = expand\_with\_replicate(res1, 1)
10. res2 = np.ones((img\_size0, img\_size1))
11. # Neighbor checking
12. for i in range(img\_size0):
13. for j in range(img\_size1):
14. if res1[i + 1, j + 1] == 1:
15. tmp = False
16. for k in range(3):
17. for l in range(3):
18. if res1[i + k, j + l] == -1:
19. tmp = True
20. break
21. if tmp:
22. break
23. if tmp:
24. res2[i, j] = 0
25. return res2 \* 255
26. Laplace Mask2 (1, 1, 1, 1, -8, 1, 1, 1, 1): 15

Description: Applies another Laplace operator for edge detection. Similar to question a, it filters edge points based on neighborhood characteristics, resulting in smoother edge outcomes.



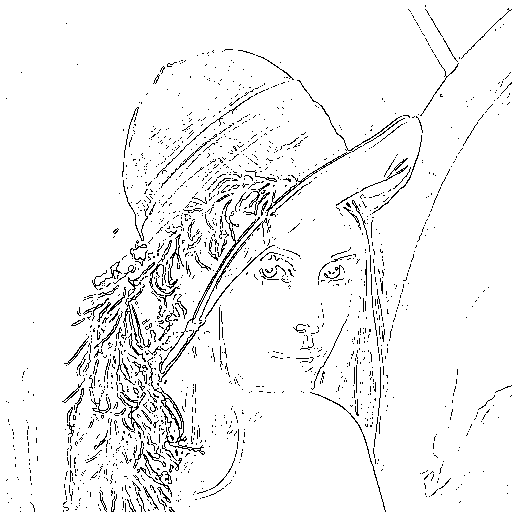
1. def laplace2(img\_arr, threshold):
2. mask = np.array([[1, 1, 1],
3. [1, -8, 1],
4. [1, 1, 1]], dtype=np.float64)
5. mask /= 3
6. # Expand input for padding using numpy
7. img\_arr = expand\_with\_replicate(img\_arr, 1)
8. # Perform convolution using numpy array
9. res1 = my\_conv(img\_arr, mask, threshold)
10. # Expand result for neighbor checking using numpy
11. res1 = expand\_with\_replicate(res1, 1)
12. res2 = np.ones((img\_size0, img\_size1)) # Initialize with ones
13. # Neighbor checking
14. for i in range(img\_size0):
15. for j in range(img\_size1):
16. if res1[i + 1, j + 1] == 1:
17. tmp = False
18. for k in range(3):
19. for l in range(3):
20. if res1[i + k, j + l] == -1:
21. tmp = True
22. break
23. if tmp:
24. break
25. if tmp:
26. res2[i, j] = 0
27. return res2 \* 255
28. Minimum variance Laplacian: 20

Description: Applies the minimum variance Laplace operator to detect edges, focusing on accurately representing edge features while minimizing noise, producing more stable detection results.



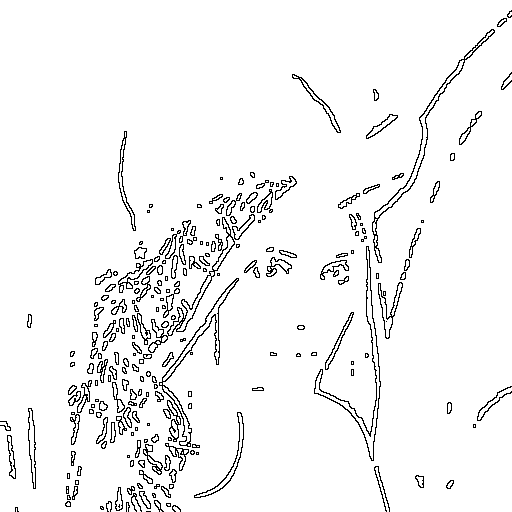
1. def minimum\_variance\_laplacian(img\_arr, threshold):
2. mask = np.array([[2, -1, 2],
3. [-1, -4, -1],
4. [2, -1, 2]], dtype=np.float64)
5. mask /= 3
6. # Expand input for padding using numpy
7. img\_arr = expand\_with\_replicate(img\_arr, 1)
8. # Perform convolution using numpy array
9. res1 = my\_conv(img\_arr, mask, threshold)
10. # Expand result for neighbor checking using numpy
11. res1 = expand\_with\_replicate(res1, 1)
12. res2 = np.ones((img\_size0, img\_size1)) # Initialize with ones
13. # Neighbor checking
14. for i in range(img\_size0):
15. for j in range(img\_size1):
16. if res1[i + 1, j + 1] == 1:
17. tmp = False
18. for k in range(3):
19. for l in range(3):
20. if res1[i + k, j + l] == -1:
21. tmp = True
22. break
23. if tmp:
24. break
25. if tmp:
26. res2[i, j] = 0
27. return res2 \* 255
28. Laplace of Gaussian: 3000

Description: Combines Gaussian smoothing with the Laplace operator. It removes high-frequency noise before edge detection, making it particularly suitable for noisy images.



1. def laplace\_gauss(img\_arr, threshold):
2. mask = np.array([[0, 0, 0, -1, -1, -2, -1, -1, 0, 0, 0],
3. [0, 0, -2, -4, -8, -9, -8, -4, -2, 0, 0],
4. [0, -2, -7, -15, -22, -23, -22, -15, -7, -2, 0],
5. [-1, -4, -15, -24, -14, -1, -14, -24, -15, -4, -1],
6. [-1, -8, -22, -14, 52, 103, 52, -14, -22, -8, -1],
7. [-2, -9, -23, -1, 103, 178, 103, -1, -23, -9, -2],
8. [-1, -8, -22, -14, 52, 103, 52, -14, -22, -8, -1],
9. [-1, -4, -15, -24, -14, -1, -14, -24, -15, -4, -1],
10. [0, -2, -7, -15, -22, -23, -22, -15, -7, -2, 0],
11. [0, 0, -2, -4, -8, -9, -8, -4, -2, 0, 0],
12. [0, 0, 0, -1, -1, -2, -1, -1, 0, 0, 0]])
13. # Expand input for padding using numpy
14. img\_arr = expand\_with\_replicate(img\_arr, 5)
15. # Perform convolution using numpy array
16. res1 = my\_conv(img\_arr, mask, threshold)
17. # Expand result for neighbor checking using numpy
18. res1 = expand\_with\_replicate(res1, 5)
19. res2 = np.ones((img\_size0, img\_size1)) # Initialize with ones
20. # Neighbor checking
21. for i in range(img\_size0):
22. for j in range(img\_size1):
23. if res1[i + 1, j + 1] == 1:
24. tmp = False
25. for k in range(3):
26. for l in range(3):
27. if res1[i + k, j + l] == -1:
28. tmp = True
29. break
30. if tmp:
31. break
32. if tmp:
33. res2[i, j] = 0
34. return res2 \* 255
35. Difference of Gaussian: 1

Description: Performs edge detection using the Difference of Gaussian method. It computes edges by applying two Gaussian filters with different standard deviations, emphasizing fine details and edge features in the image.



1. def difference\_gauss(img\_arr, threshold):
2. mask = np.array([[-1, -3, -4, -6, -7, -8, -7, -6, -4, -3, -1],
3. [-3, -5, -8, -11, -13, -13, -13, -11, -8, -5, -3],
4. [-4, -8, -12, -16, -17, -17, -17, -16, -12, -8, -4],
5. [-6, -11, -16, -16, 0, 15, 0, -16, -16, -11, -6],
6. [-7, -13, -17, 0, 85, 160, 85, 0, -17, -13, -7],
7. [-8, -13, -17, 15, 160, 283, 160, 15, -17, -13, -8],
8. [-7, -13, -17, 0, 85, 160, 85, 0, -17, -13, -7],
9. [-6, -11, -16, -16, 0, 15, 0, -16, -16, -11, -6],
10. [-4, -8, -12, -16, -17, -17, -17, -16, -12, -8, -4],
11. [-3, -5, -8, -11, -13, -13, -13, -11, -8, -5, -3],
12. [-1, -3, -4, -6, -7, -8, -7, -6, -4, -3, -1]])
13. # Expand input for padding using numpy
14. img\_arr = expand\_with\_replicate(img\_arr, 5)
15. # Perform convolution using numpy array
16. res1 = my\_conv(img\_arr, mask, threshold)
17. # Expand result for neighbor checking using numpy
18. res1 = expand\_with\_replicate(res1, 5)
19. res2 = np.ones((img\_size0, img\_size1)) # Initialize with ones
20. # Neighbor checking
21. for i in range(img\_size0):
22. for j in range(img\_size1):
23. if res1[i + 1, j + 1] == 1:
24. tmp = False
25. for k in range(3):
26. for l in range(3):
27. if res1[i + k, j + l] == -1:
28. tmp = True
29. break
30. if tmp:
31. break
32. if tmp:
33. res2[i, j] = 0
34. return res2 \* 255