**電腦視覺**

**Homework 7**

**Contents:**

1. Description
2. Methodology
3. Code fragments
4. Images

Description

This is the report for the seventh homework for Computer Vision 2019. It is completed in full.

The environment used is Windows, with the code being written in Python 3 with the cv2 module.

Methodology

The method described in the slides (as provided on the homework webpage) is followed. It takes **7** iterations for the resulting images to become identical.

The resulting Python program, **hw7.py**, should produce two images: **meow.bmp**, which is the 64x64 thinned image, and **meow\_large.bmp**, which is an enlarged version of **meow.bmp**.

In general, you do not have to consider the enlarged picture if the smaller one is clear enough; the large image is just there for clarity.

Code Fragments

Yokoi calculation (nearly identical to HW6):

1. **def** f(self, a, b, c, d):
2. **return** 5 **if** a == b == c == d == self.R **else** (boo(a == self.Q) + boo(b == self.Q) + boo(c == self.Q) + boo(d == self.Q))
4. **def** H(self, x, a, b, c):
5. **if** x != a: **return** self.S
6. **if** x == a == b == c: **return** self.R
7. **if** x == a **and** (x != b **or** x != c): **return** self.Q
8. **assert** 0
10. **def** yokoi\_value(self, image, i, j):
11. rmax, cmax = image.shape[0], image.shape[1]
12. **def** c(r, c):
13. **return** image[r, c] **if** 0 <= r < rmax **and** 0 <= c < cmax **else** -1
14. a1 = self.H(c(i,j), c(i,j+1), c(i-1,j+1), c(i-1,j))
15. a2 = self.H(c(i,j), c(i-1,j), c(i-1,j-1), c(i,j-1))
16. a3 = self.H(c(i,j), c(i,j-1), c(i+1,j-1), c(i+1,j))
17. a4 = self.H(c(i,j), c(i+1,j), c(i+1,j+1), c(i,j+1))
18. f\_ = self.f(a1, a2, a3, a4)
19. **return** f\_
21. **def** yokoi(self, sixfour):
22. temp = sixfour.copy()
23. h, w = sixfour.shape[0], sixfour.shape[1]
24. **for** i **in** range(h):
25. **for** j **in** range(w):
26. **if** sixfour[i,j] == WHITE:
27. sixfour[i, j] = self.yokoi\_value(temp, i, j)
28. **else**:
29. sixfour[i, j] = 0
30. **return** sixfour

Marked and Pair Relationship:

1. **def** pair\_relationship(yokoi\_img): # yokoi\_img is an array NUMBERS, not an image
2. marked = set()
3. r, c = yokoi\_img.shape[0], yokoi\_img.shape[1]
4. **def** h(i, j):
5. **return** 1 **if** 0 <= i < r **and** 0 <= j < c **and** yokoi\_img[i, j] == 1 **else** 0
6. **for** i **in** range(r):
7. **for** j **in** range(c):
8. **if** yokoi\_img[i, j] > 0:
9. **if** (h(i+1, j) + h(i-1, j) + h(i, j+1) + h(i, j-1) >= 1) **and** yokoi\_img[i, j] == 1:
10. marked.add((i,j))
11. **return** marked
13. **def** mark(image) -> set:
14. yokoi\_img = Yokoi.yokoi(image) # get yokoi connectivity matrix
15. # print(yokoi\_img)
16. marked = pair\_relationship(yokoi\_img)
17. **return** marked

Actual thinning operation:

1. **def** thin(image):
2. old\_img = image.copy()
3. **while** True:
4. new\_img = thin\_once(old\_img.copy())
5. **if** is\_identical(old\_img, new\_img): **break**
6. old\_img = new\_img.copy()
7. **return** old\_img

Images

Original image:



Resized back to 512 x 512:

