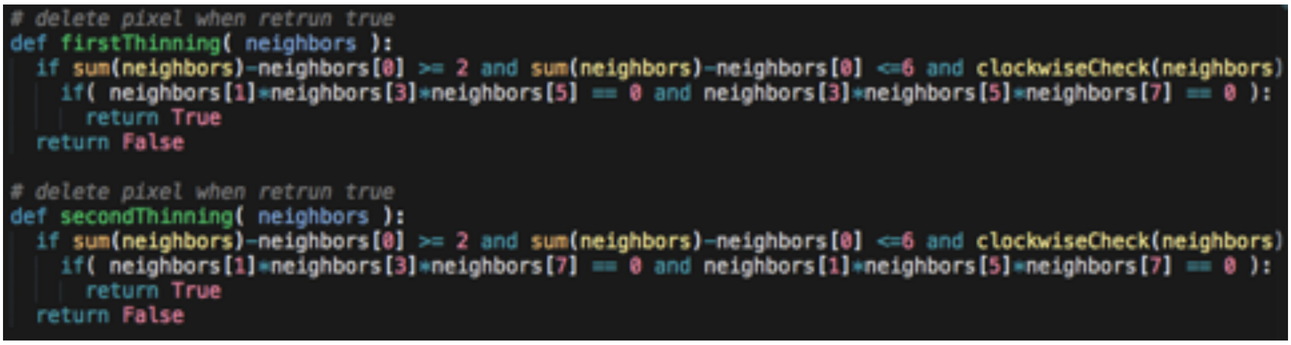
R02944010 李柏緯

Computer Vision Homework 7

How to run it

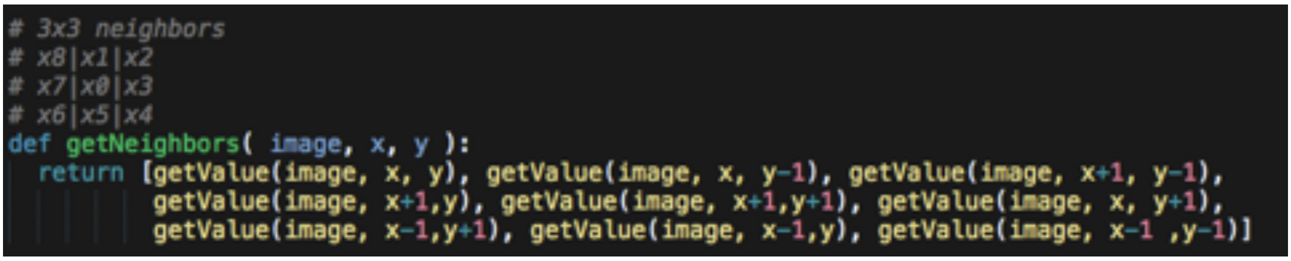
```
python image.py lena.bmp
```

Principal code fragment



```
# delete pixel when retrun true
def firstThinning( neighbors ):
  if sum(neighbors)-neighbors[0] >= 2 and sum(neighbors)-neighbors[0] <=6 and clockwiseCheck(neighbors)
    if( neighbors[1]*neighbors[3]*neighbors[5] == 0 and neighbors[3]*neighbors[5]*neighbors[7] == 0 ):
      return True
  return False

# delete pixel when retrun true
def secondThinning( neighbors ):
  if sum(neighbors)-neighbors[0] >= 2 and sum(neighbors)-neighbors[0] <=6 and clockwiseCheck(neighbors)
    if( neighbors[1]*neighbors[3]*neighbors[7] == 0 and neighbors[1]*neighbors[5]*neighbors[7] == 0 ):
      return True
  return False
```



```
# 3x3 neighbors
# x8|x1|x2
# x7|x0|x3
# x6|x5|x4
def getNeighbors( image, x, y ):
  return [getValue(image, x, y), getValue(image, x, y-1), getValue(image, x+1, y-1),
          getValue(image, x+1,y), getValue(image, x+1,y+1), getValue(image, x, y+1),
          getValue(image, x-1,y+1), getValue(image, x-1,y), getValue(image, x-1 ,y-1)]
```

Description

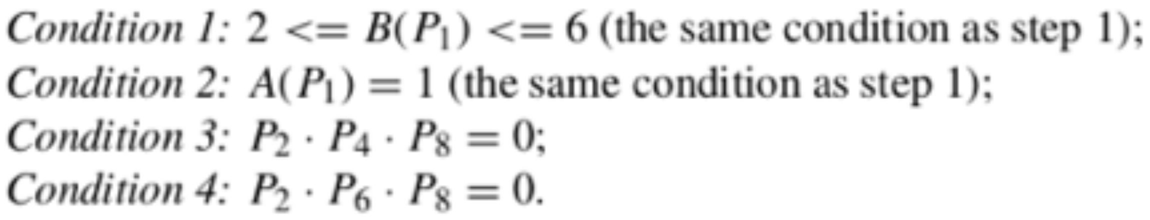
Check every pixel’s neighbors in image.

- $A(P_1)$ = number of 0, 1 patterns (transitions from 0 to 1) in the ordered sequence of $P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9, P_2$.
- $B(P_1) = P_2 + P_3 + P_4 + P_5 + P_6 + P_7 + P_8 + P_9$ (number of black or 1 pixel, neighbors of P_1).

Step 1 conditions for selecting black points to remove:

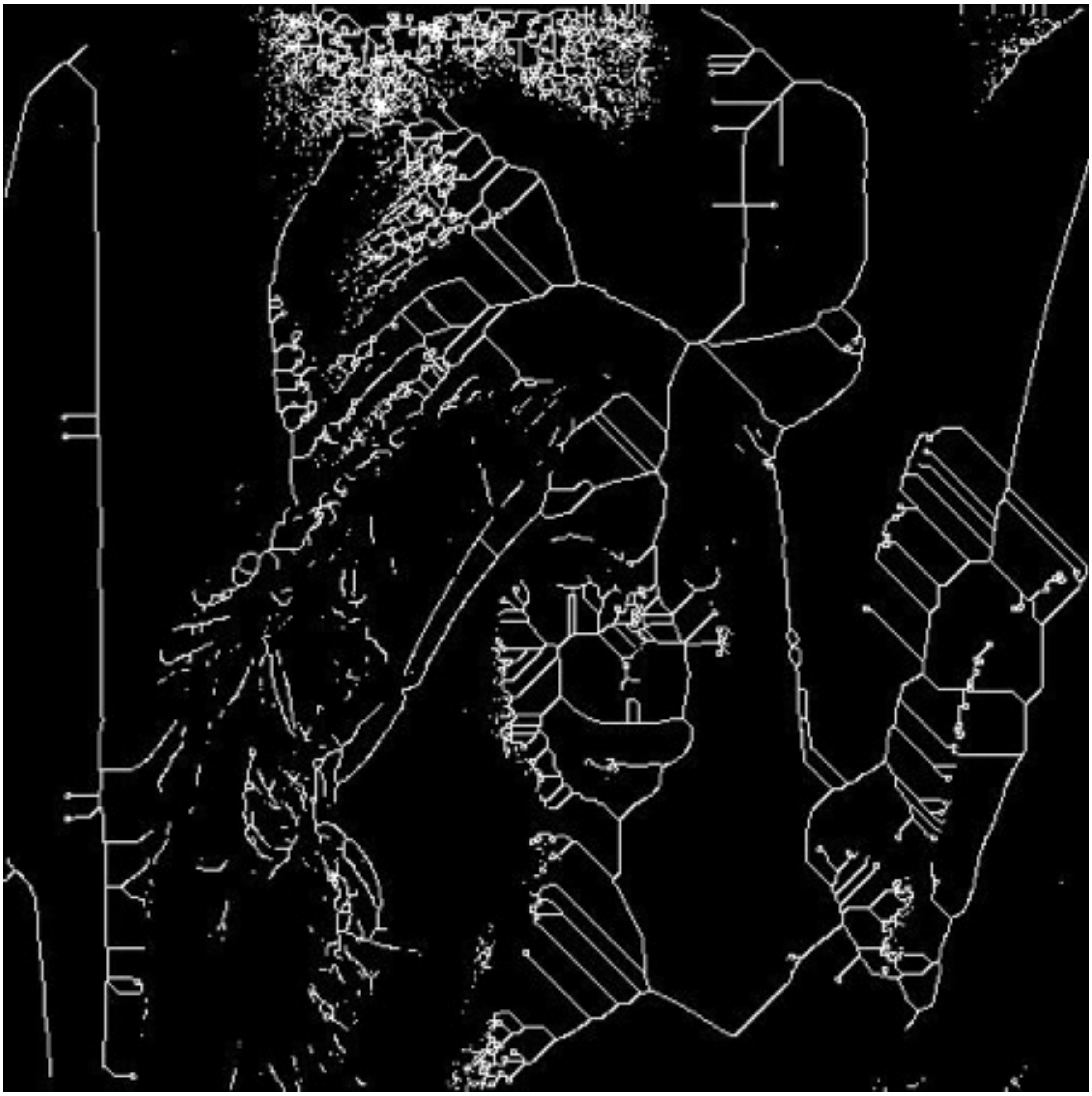
$$\textit{Condition 1: } 2 <= B(P_1) <= 6;$$
$$\textit{Condition 2: } A(P_1) = 1;$$
$$\textit{Condition 3: } P_2 \cdot P_4 \cdot P_6 = 0;$$
$$\textit{Condition 4: } P_4 \cdot P_6 \cdot P_8 = 0.$$

Step 2 conditions for selecting black points to remove:



Condition 1: $2 <= B(P_1) <= 6$ (the same condition as step 1);
Condition 2: $A(P_1) = 1$ (the same condition as step 1);
Condition 3: $P_2 \cdot P_4 \cdot P_8 = 0$;
Condition 4: $P_2 \cdot P_6 \cdot P_8 = 0$.

Result



R02944010 李柏緯

Reference

http://nayefreza.wordpress.com/2013/05/11/zhang-suen-thinning-algorithm-java-implementation/