CIE 6004(Fall 2019) Assignment 2

AUTHOR

Name: Jian Zhang
Student ID: 219012058
MSc Program: Data Science

October 25, 2019

Contents

1 Introduction

This report is about the Assignment 2 of "Image Processing and Computer Vision". We mainly discuss straight line detection, boundary following and morphological image processing. For most parts of the exercise, the function is implemented by myself instead of applying the function of library. And my code is based on Python.

The report includes the description of the problems, the algorithm to solve them and the results of the resolution.

2 Exercise

2.1 Straight Line Detection

(a) Apply Canny edge detector to the image of House.png using available functions. Adjust the parameters to obtain a good result.

The following steps to generate a good result.

Step1: Apply $7 * 7$ mean filter to the house image.
Step2: Apply gaussianblur to the image by $17 * 17$ kernel.
Step3: Apply canny method and tune parameters to get a good result.

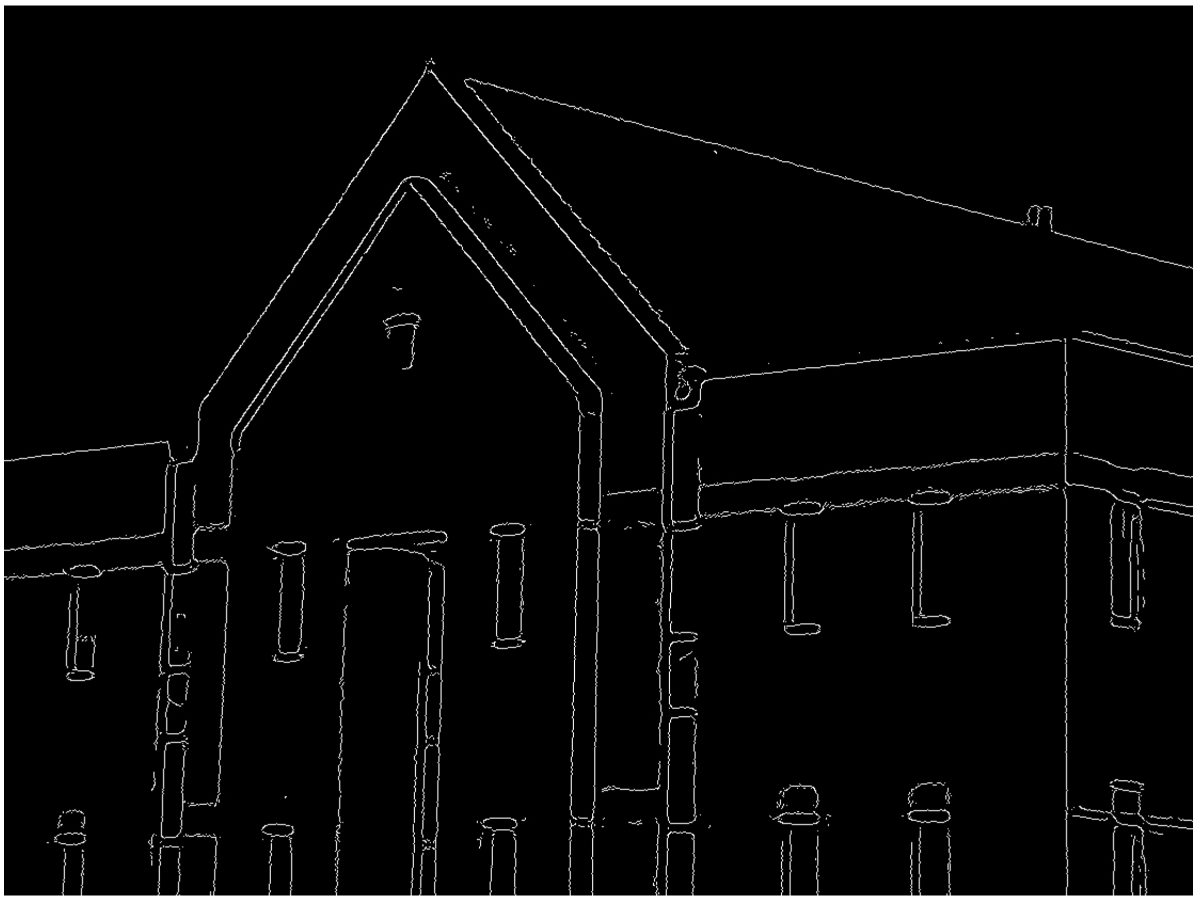


Figure 1: Canny edge of 'House.jpg'

(b) On the basics of (a), implement the Hough transforamtion to detect the straight line on 'House.jpg'

Follow the Hough transformation algorithm, we calculate the Hesse normal form $r = xcos(\theta) + ysin(\theta)$ for every pixel.

Then, set the threshold for the accumulator. If the number of pair (r, θ) is bigger than threshold(choose 90 for House.jpg), add this line into the ListLines.

After that, we have many lines in the list. We paint the lines on the 'House.jpg'

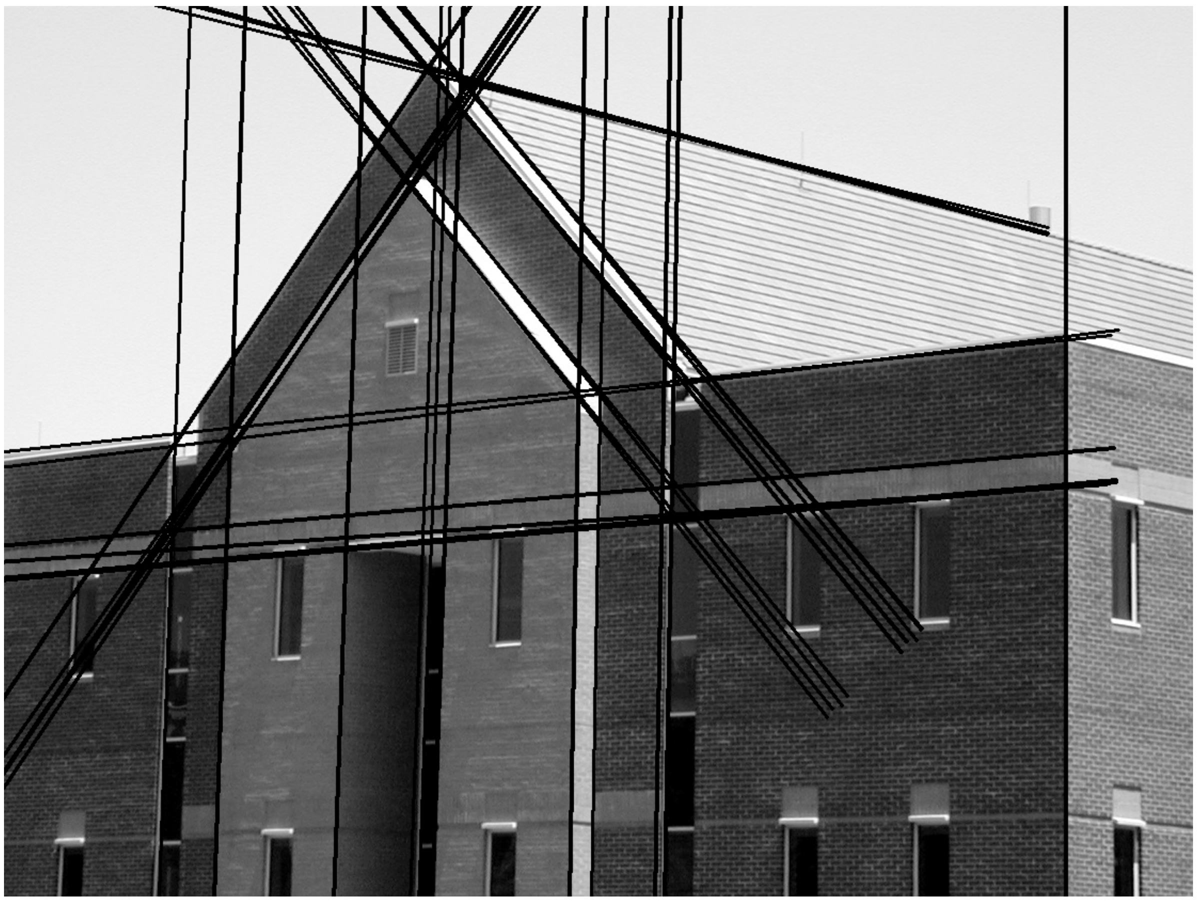
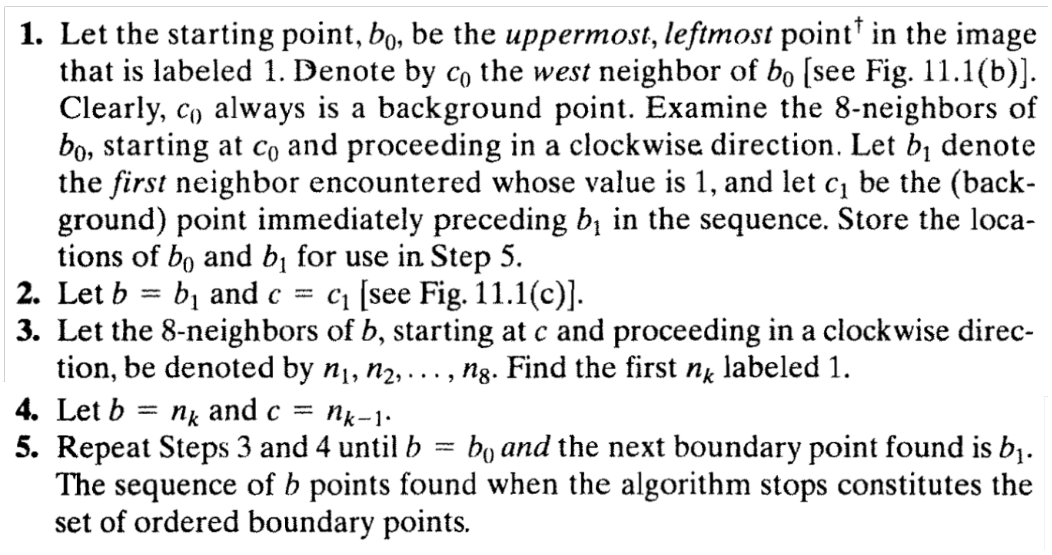


Figure 2: Lines detection on 'House.jpg'

2.2 Boundary Following

(a) Implement the boundary following algorithm

Boundary following algorithm steps follows:



1. Let the starting point, b_0, be the *uppermost, leftmost* point[†] in the image that is labeled 1. Denote by c_0 the *west* neighbor of b_0 [see Fig. 11.1(b)]. Clearly, c_0 always is a background point. Examine the 8-neighbors of b_0, starting at c_0 and proceeding in a clockwise direction. Let b_1 denote the *first* neighbor encountered whose value is 1, and let c_1 be the (background) point immediately preceding b_1 in the sequence. Store the locations of b_0 and b_1 for use in Step 5.
2. Let $b = b_1$ and $c = c_1$ [see Fig. 11.1(c)].
3. Let the 8-neighbors of b, starting at c and proceeding in a clockwise direction, be denoted by $n_1, n_2, \ldots, n_8$. Find the first n_k labeled 1.
4. Let $b = n_k$ and $c = n_{k-1}$.
5. Repeat Steps 3 and 4 until $b = b_0$ *and* the next boundary point found is b_1. The sequence of b points found when the algorithm stops constitutes the set of ordered boundary points.

Figure 3: Boundary following algorithm

For implementation, I initialize a dictionary for clockwise operations. Given a tuple of position, it returns the next clockwise position. Then, I find the uppermost and leftmost point b0. Start from b0, I check each pixel around current point and record bi, ci if checked point is positive. Finally, if the loop comes to terminal condition(Jacob's condition), then we find all the points of shape.

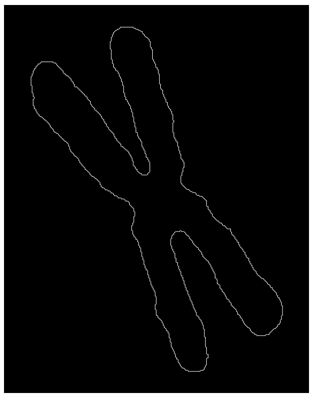
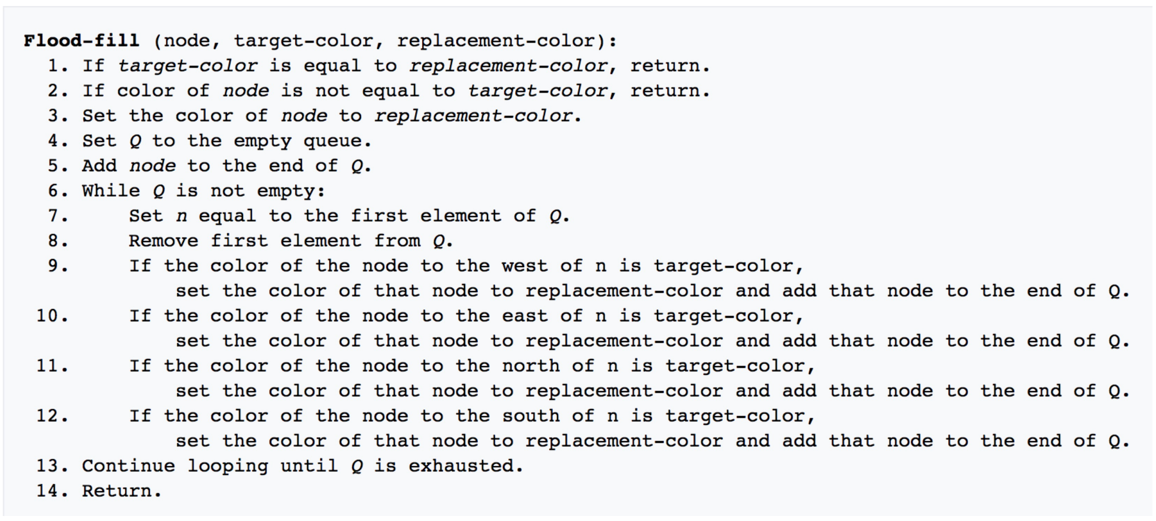


Figure 4: Boundary Following Points

(b) In addition, paint the X shape with white color.

I use Flood-fill Algorithm to reach the goal.



```
Flood-fill (node, target-color, replacement-color):
  1. If target-color is equal to replacement-color, return.
  2. If color of node is not equal to target-color, return.
  3. Set the color of node to replacement-color.
  4. Set Q to the empty queue.
  5. Add node to the end of Q.
  6. While Q is not empty:
  7.     Set n equal to the first element of Q.
  8.     Remove first element from Q.
  9.     If the color of the node to the west of n is target-color,
             set the color of that node to replacement-color and add that node to the end of Q.
 10.     If the color of the node to the east of n is target-color,
             set the color of that node to replacement-color and add that node to the end of Q.
 11.     If the color of the node to the north of n is target-color,
             set the color of that node to replacement-color and add that node to the end of Q.
 12.     If the color of the node to the south of n is target-color,
             set the color of that node to replacement-color and add that node to the end of Q.
 13. Continue looping until Q is exhausted.
 14. Return.
```

Figure 5: Flood-fill Algorithm

As the image contains many pixels, I use queue instead of recursive to finish the algorithm. Also, I choose four-way instead of eight-way, which is enough for this problem.

To apply this algorithm, I need to choose a pixel inside the X shape as parameter, which I simply choose the center of the image here.

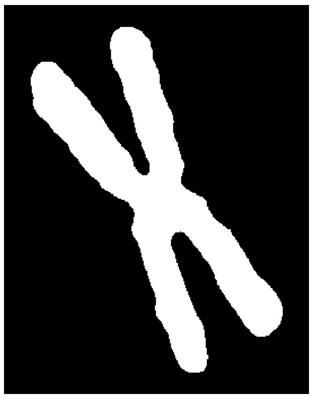


Figure 6: Painted X shape

2.3 Morphological Image Processing

(a) Implement Opening operation to eliminate the most thinnest lines between different objects on the image of 'Wirebond.png', but other lines should be kept.

(b) Implement Closing operation to make the text on the image of 'Text.png' look better.

To solve these two problems, we should implement erosion and dilation function. Then, choose a proper kernel for each problem.

For problem(a), we apply Opening operation, which applies the erosion function first and then the dilation function. The kernel is $3 * 3$ matrix and all components are equal to one.

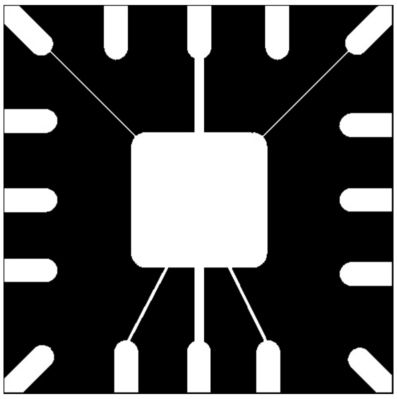


Figure 7: Erosion of Wirebond.jpg

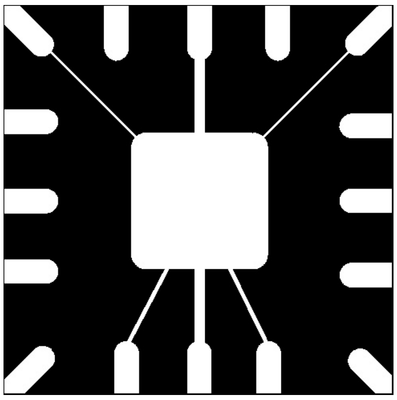
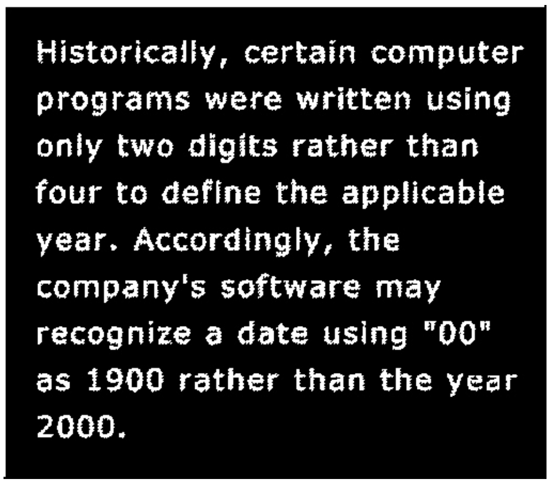


Figure 8: Opening operation of Wirebond.jpg

For problem(b), we apply Closing operation, which applies the dilation function first and then the dilation function.

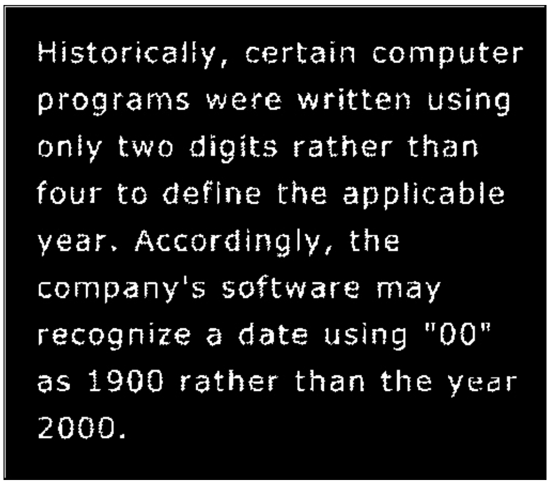
The kernel is $3 * 3$ matrix:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$



Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.

Figure 9: Erosion of Wirebond.jpg



Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.

Figure 10: Opening operation of Wirebond.jpg