Count Parking Lots

Tianzheng Huang
Li Xu

Dept. of Computer Science, University of Missouri,
Columbia, MO 65211, USA

Digital Image Processing Final Project
Wed Dec 16th , 2015, 5:00 pm

Outline

- Motivation and Background
- Image Processing Tasks and Goals
- Algorithm and Implementation Details
- Results: Demo and Evaluation
- Summary and Future Work

Motivation and Background

More and more people have private car, and find an available parking lots is harder than before.

The traditional way to check whether the parking area is full or not is arranging a sensor at the entrance of the parking location.

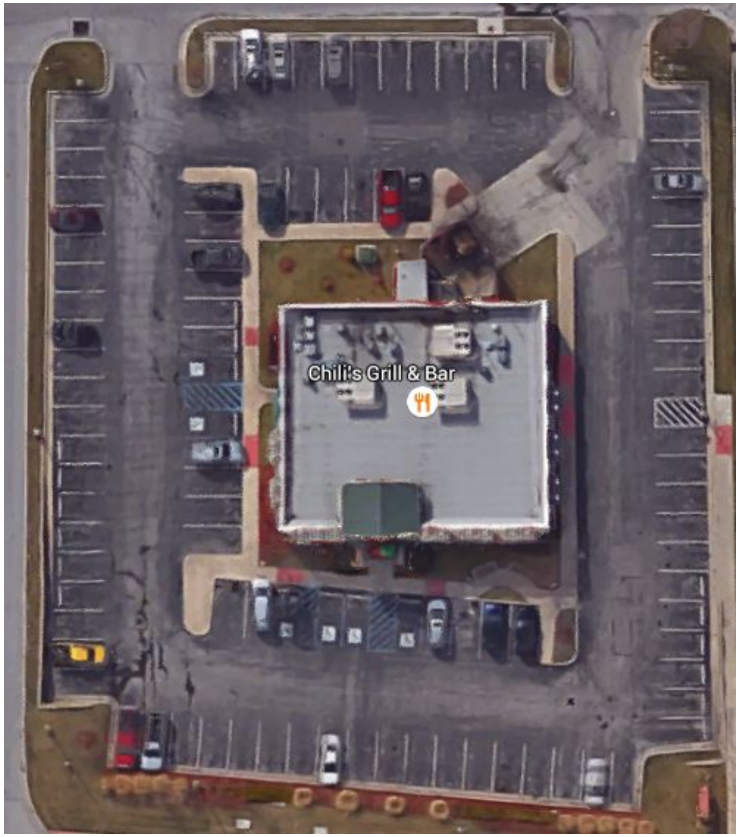
Disadvantage : costly and not efficient.

Image Processing Tasks & Goals

Here are sample images, come from Google Map

Sample 1

Sample 2



Sample 3

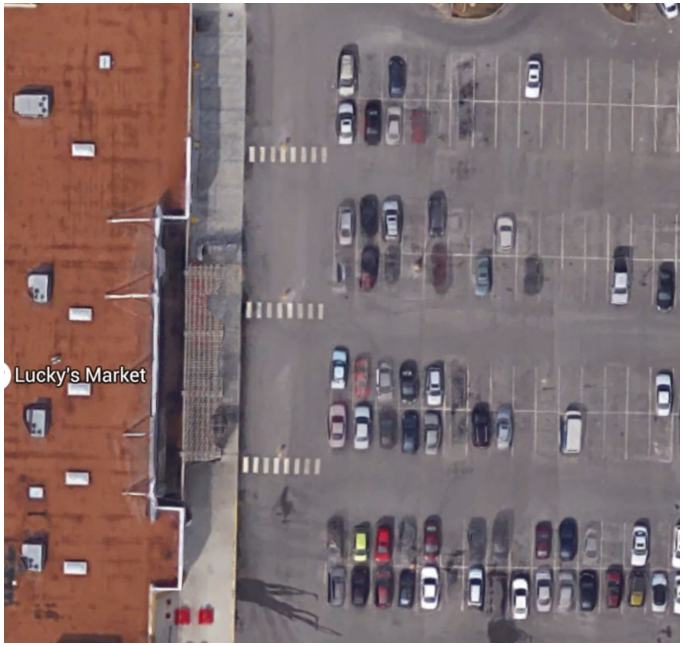


Image Processing Tasks & Goals

We totally have 7 steps to achieve our parking lot detection.

Step1: using sobel filter to get Gx, Gy edge detection image.

Step2: using connected component label to get all connected area.

Step3: first time remove noise. (the threshold is [20, 200]).

Step4: record each line' s information. (leftmost_x1, rightmost_x2, highest_y1, lowest_y2)

Step5: second time remove noise (remove not line-shape spot).

Step6: keep the parking line and remove other line-shape but not the parking line noise.

Step7: count number of Gx and Gy image parking lines.

Algorithm and Implementation Details

Step1: Sobel filter edge detection: Because the parking lots in the picture have different direction : horizonnl(Gx) and vertical(Gy)

Step2: Connected component label: Because we need use the parrallel line to be standard line. And use it to get special information and contrast other noise line.

Step 3: Get each lable size and set a threshold [20. 200], first time remove noise.

Algorithm and Implementation Details

Whetherlabelx (to check if the label has got its beginning point)

label	0	1	......	n
ischecked(0/1)	0	0	....	0

Recordlabelx

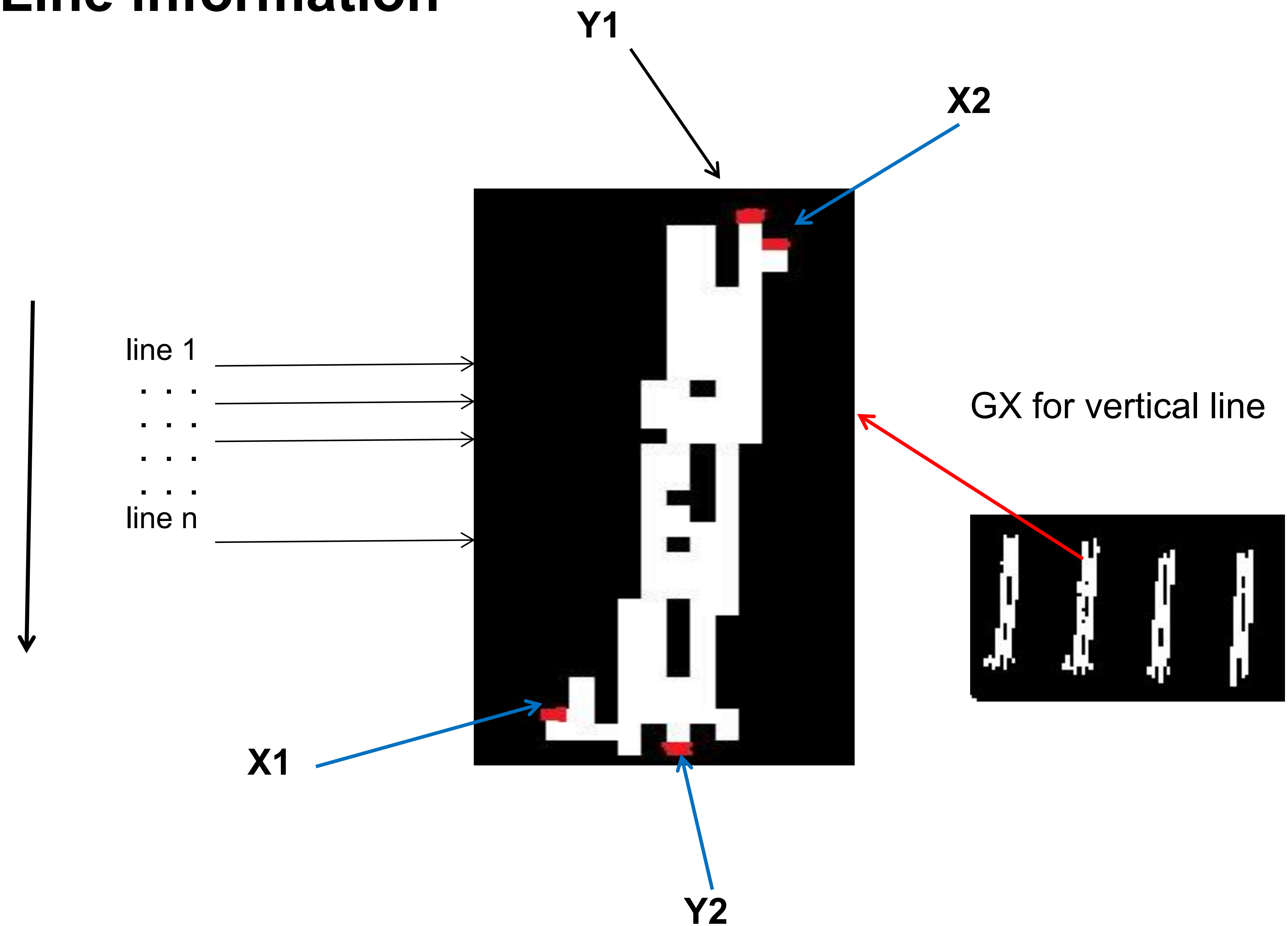
	label	1	....	n
1	Mostleft(X1)			
2	Mostup(Y1)			
3	Mostright(X2)			
4	Mostdown(Y2)			
5	Width(X2-X1)			
6	Hight(Y2-Y1)			
7	CenterX(X1+X2/2)			
8	CenterY(Y1+Y2/2)			
9	Ischecked(0/1)			
10	Isline(0/1)			

Label information

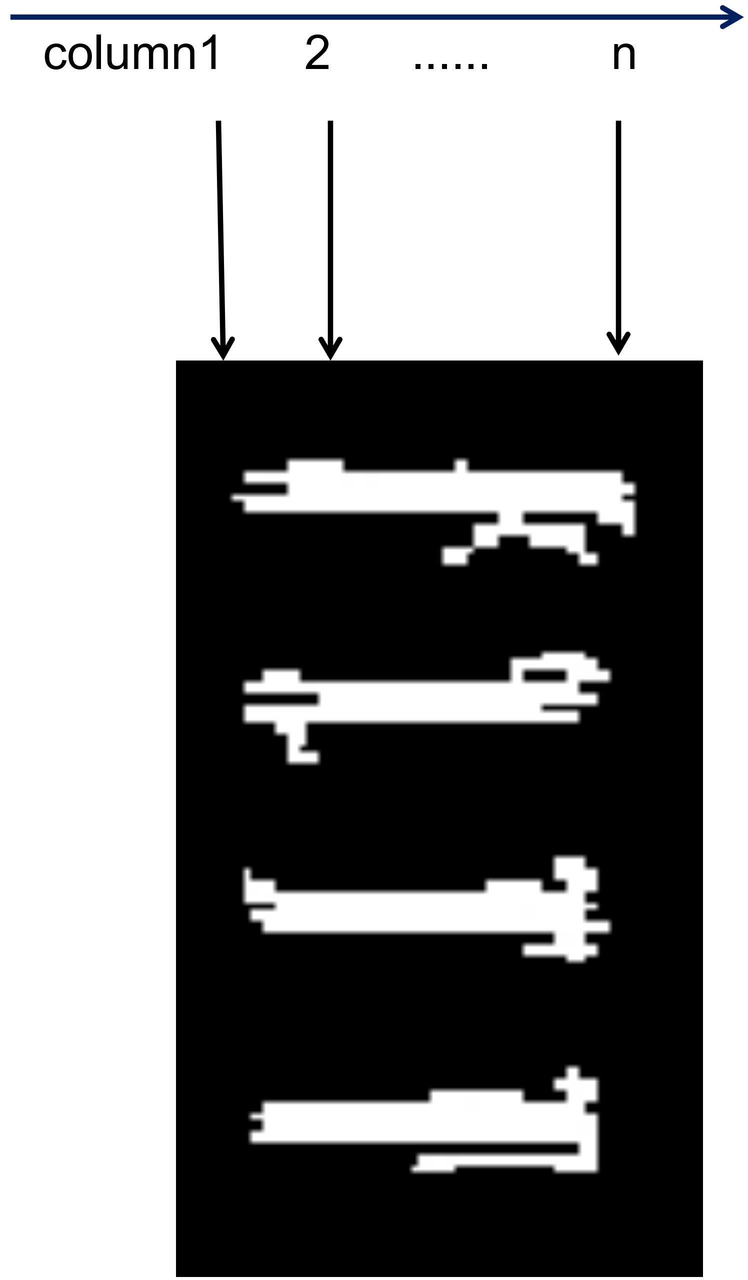
Recordlabely

	label	1	....	n
1	Mostup(Y1)			
2	Mostleft(X1)			
3	Mostdown(Y2)			
4	Mostright(X2)			
5	Hight(Y2-Y1)			
6	Width(X2-X1)			
7	CenterY(Y1+Y2/2)			
8	CenterX(X1+X2/2)			
9	Ischecked(0/1)			
10	Isline(0/1)			

Line information



Line information



GY or horizontal line

Step4: pseudocode

Algorithm#1: record X1,X2,Y1,Y2

```
for i = 1 to height of image
    for j = 1 to width of image
         if Label(i,j) is labeled
            index = Label( i, j )
     if whetherlabelx(index)==0//no check before
        X1= j;
        Y1= i;
        whetherlabely(index) = 1 ;

            if whetherlabely(index) == 1
        if j<X1
          X1 = j
       else if j>X2
          X2  = j
           Y2 = i
```

Algorithm#2: calculate the mode width and length

```
e.g for GX
max_x = max width of components in GX
modex = zeros(1,max_xwidth);
for i = 1 to labelnumx
    if width(i)>0
        modex(width(i) ++;
normalwidthx = 0
for i = 1 to max_x
    if mode(i)>normalwidthx
        normalwidthx =  i;
end
```

similiar way to count normalwidthy

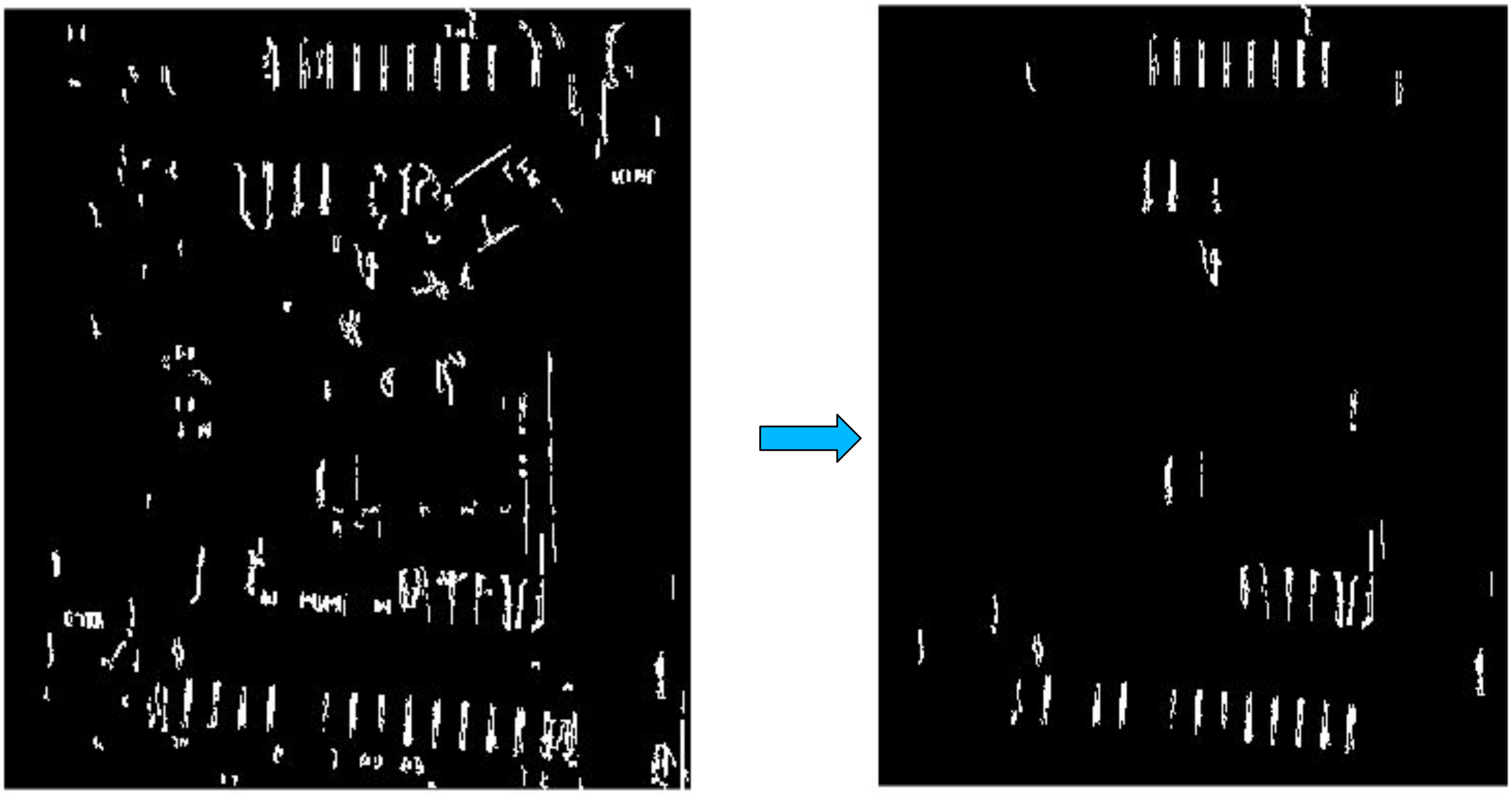
modex example

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	6	15	20	22	16	24	26	15	8	8	3	3	

normalwidthx = 7 in this image

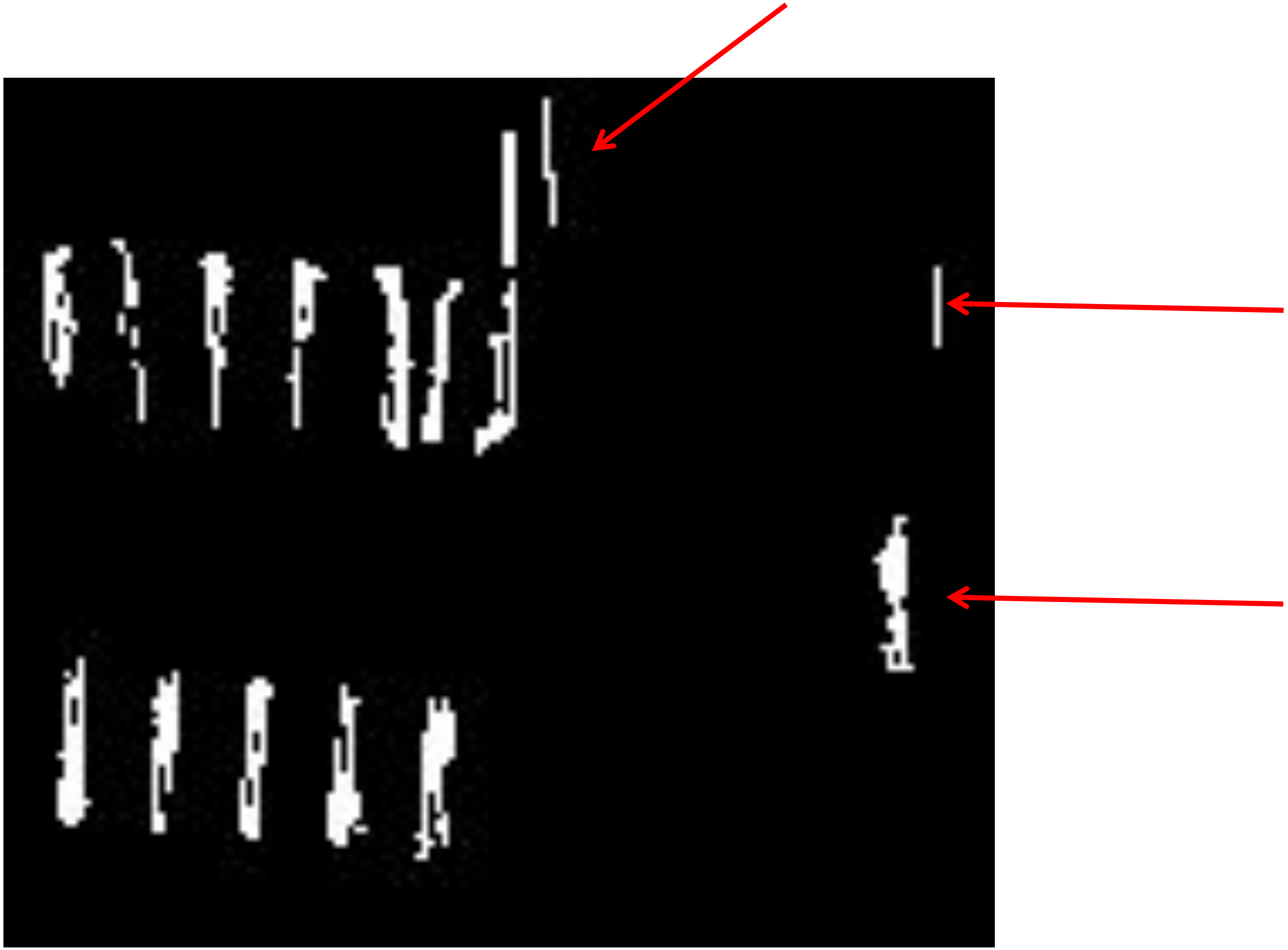
Step 5: Remove width and height is not like a line

Because line-components appear most in the picture(after removing noise in stage 3, we have remove many noises). We statistic that the width and length that appears most by checking the list of recordlabel
and that is the round width and length of a parking line.



width>1.5normalwith or hight<3/4normal height or >3/2 normal height
remove the component whetherlabel(index) = 0

Step 6 :Remove some line-like components



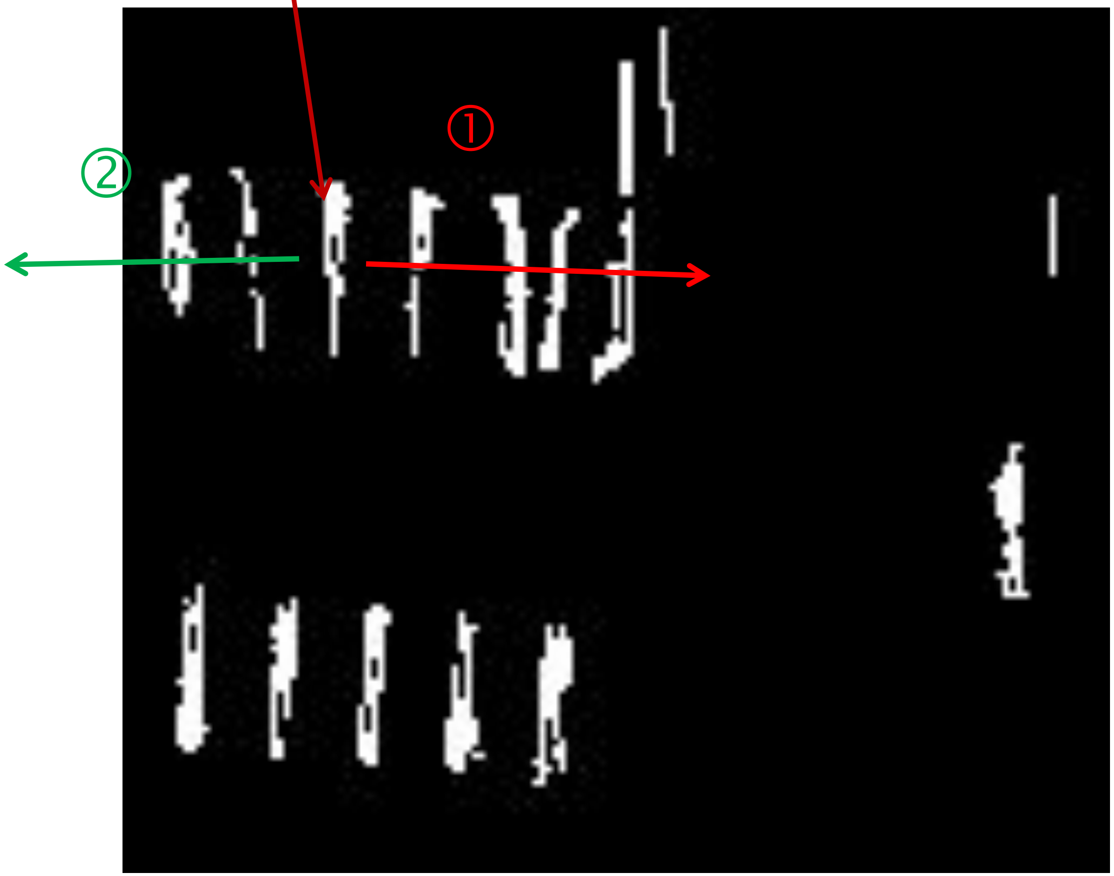
There are some lines seperated can not be considered to a parking line.

Algorithm#3 how to check line-like components not a line

```
for i = 0 to labelnum
  if whetherlabelx  == 1 //is a component left in present image?
     if Ischecked== 0  // no checked label
         Ischecked=1 // has checked the label
         cx = centerx(i), cy = centery(i)
         countback = countwhite = 0
         while countback >5normalwidth && not over the boundary
            index = label(cy,cx)
      if index~= 0// is not black
          if countback >2normalwidth && label(cy,cx)!= i
              countx++;
              Ischecked(index) = 0;
              Isline(i) = Isline(index) = 0;
          countblack = 0 ;
          countwhite ++;
      if index == 0 //is black
          countwhite = 0;
          countblack++;
      cx++; //move one right pixel
end
```

it is the same way to check one's left line by change cx++ to cx--

if we find label refer
to this component



use 2 variable countwhite
and countblack.

If meet black countblack add
1 and countwhite turn to 0

if meet white judge if countblack is over 2 times normal line width
if so thought here is a parking lot and the next line and former line is
a parking lot
then turn countblack to 0 and turn countwhite add 1

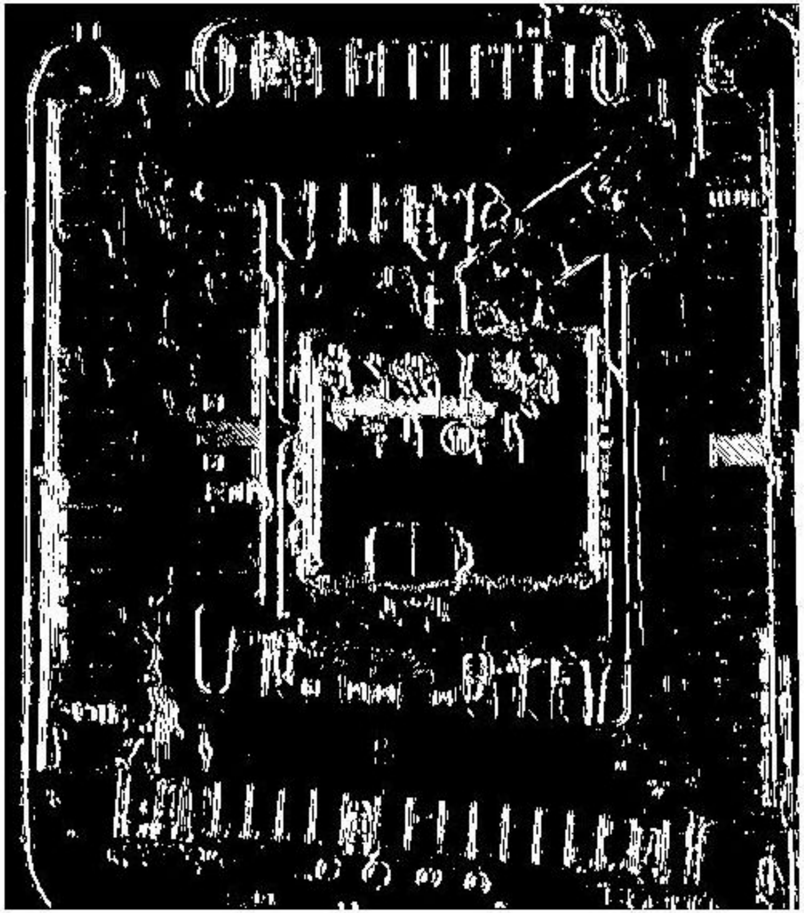
Results: Demo & Evaluation

This is our input image comes from the Google Map

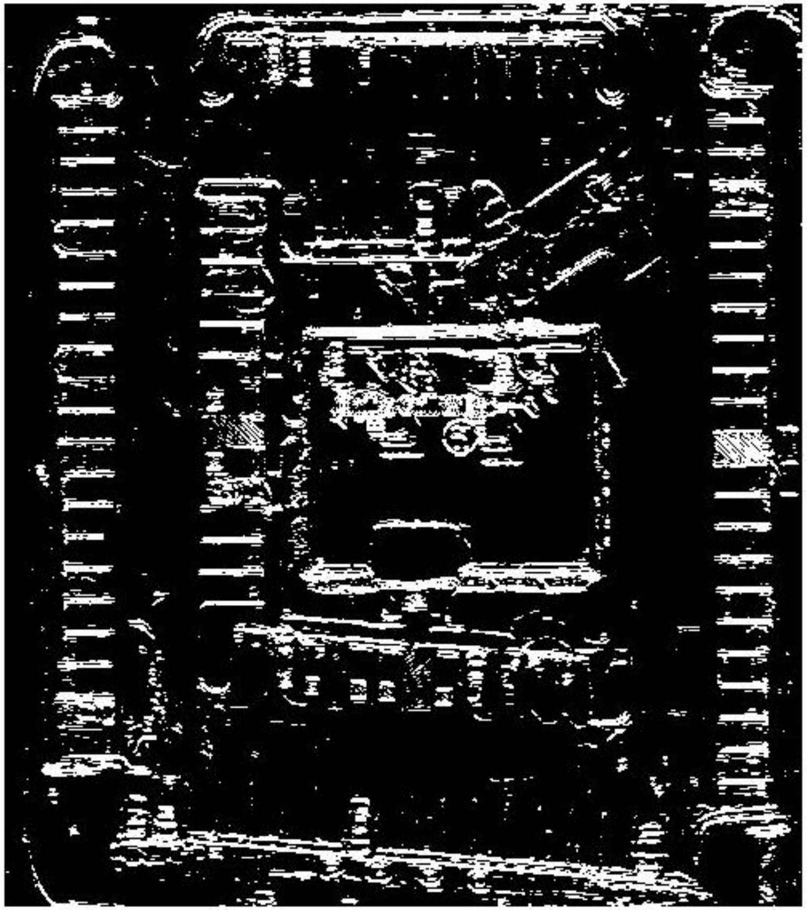
Results: Demo & Evaluation

After step1, we get x and y direction image.

Gx



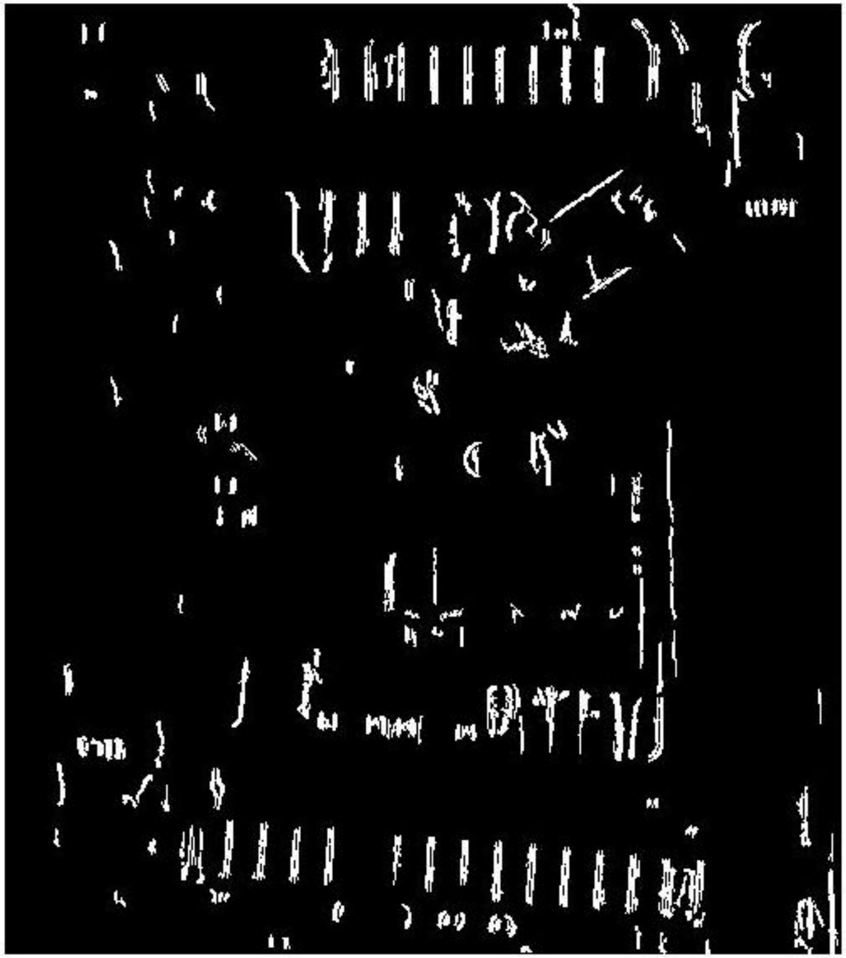
Gy



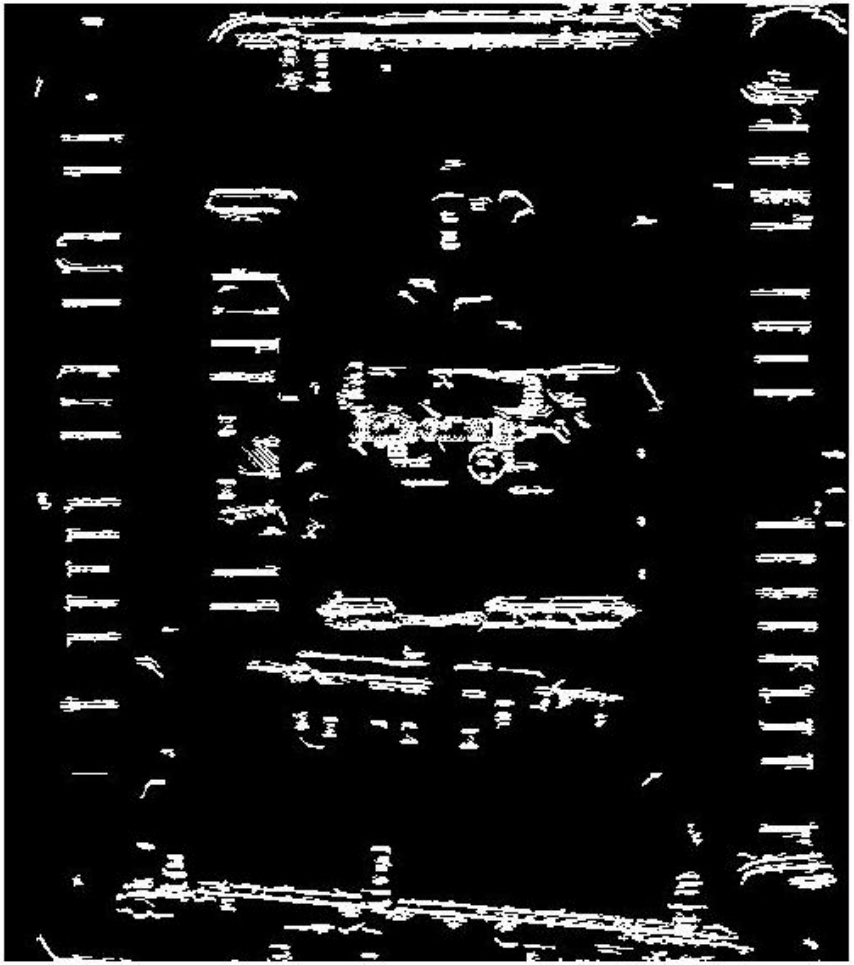
Results: Demo & Evaluation

After step 3, we get first time remove noise image

Gx



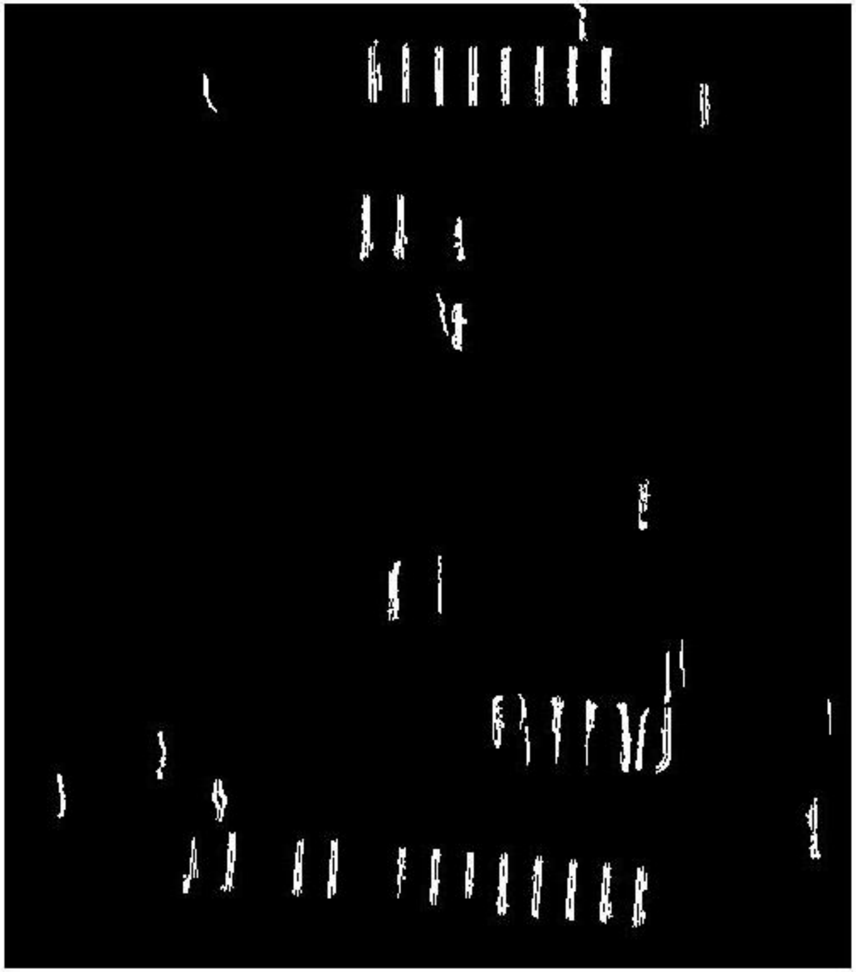
Gy



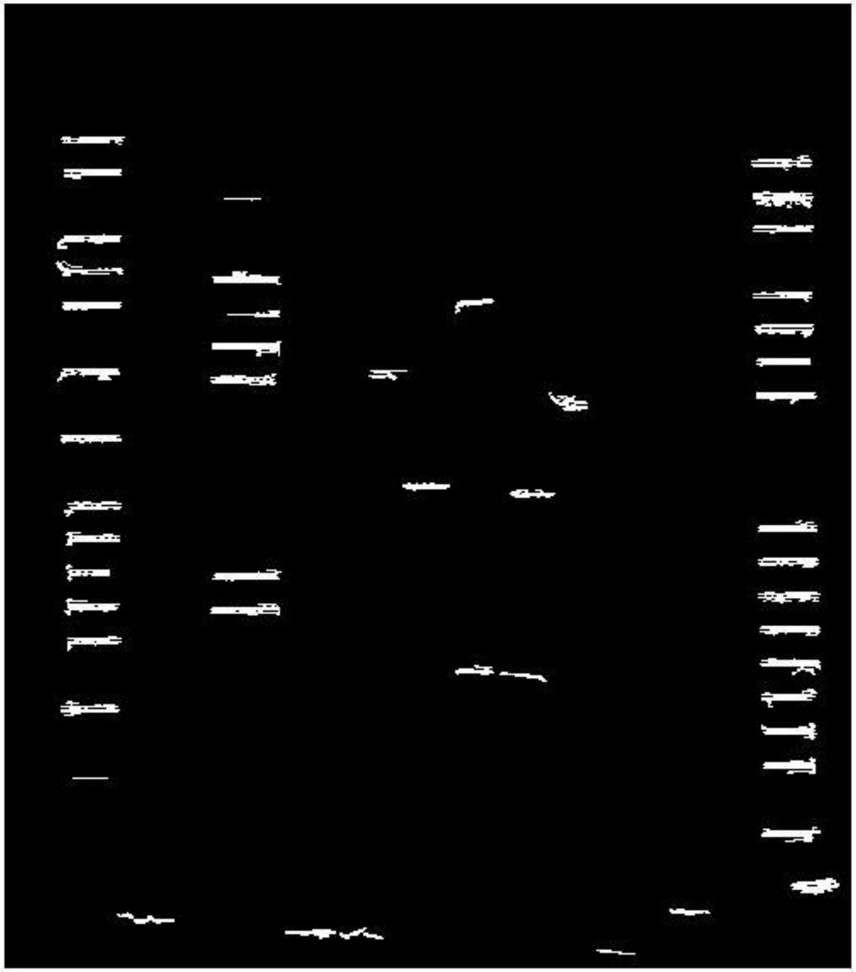
Results: Demo & Evaluation

After step 5, we keep all line-shape image.

Gx



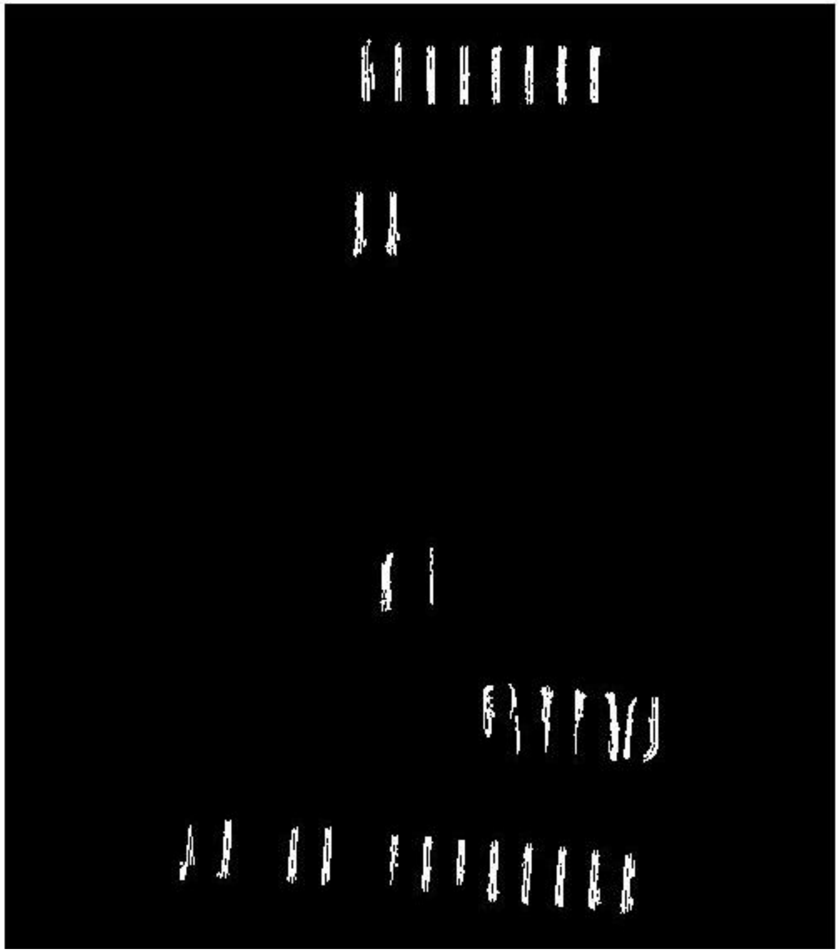
Gy



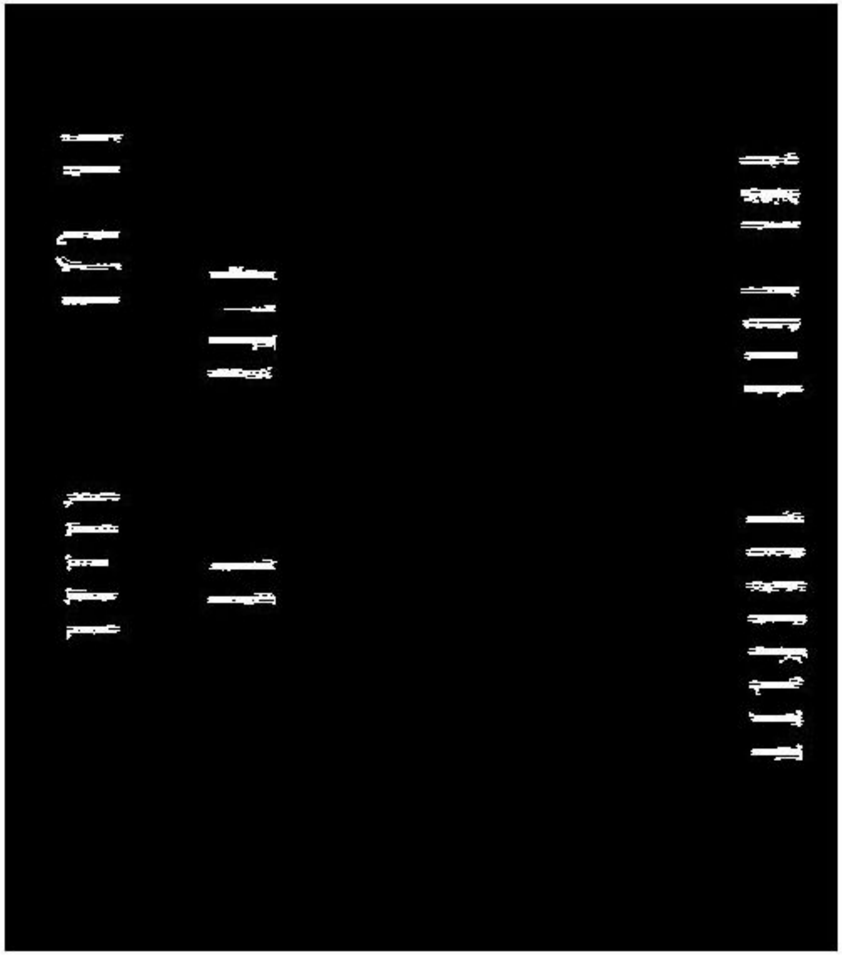
Results: Demo & Evaluation

After step 6, we keep all parking lines.

Gx

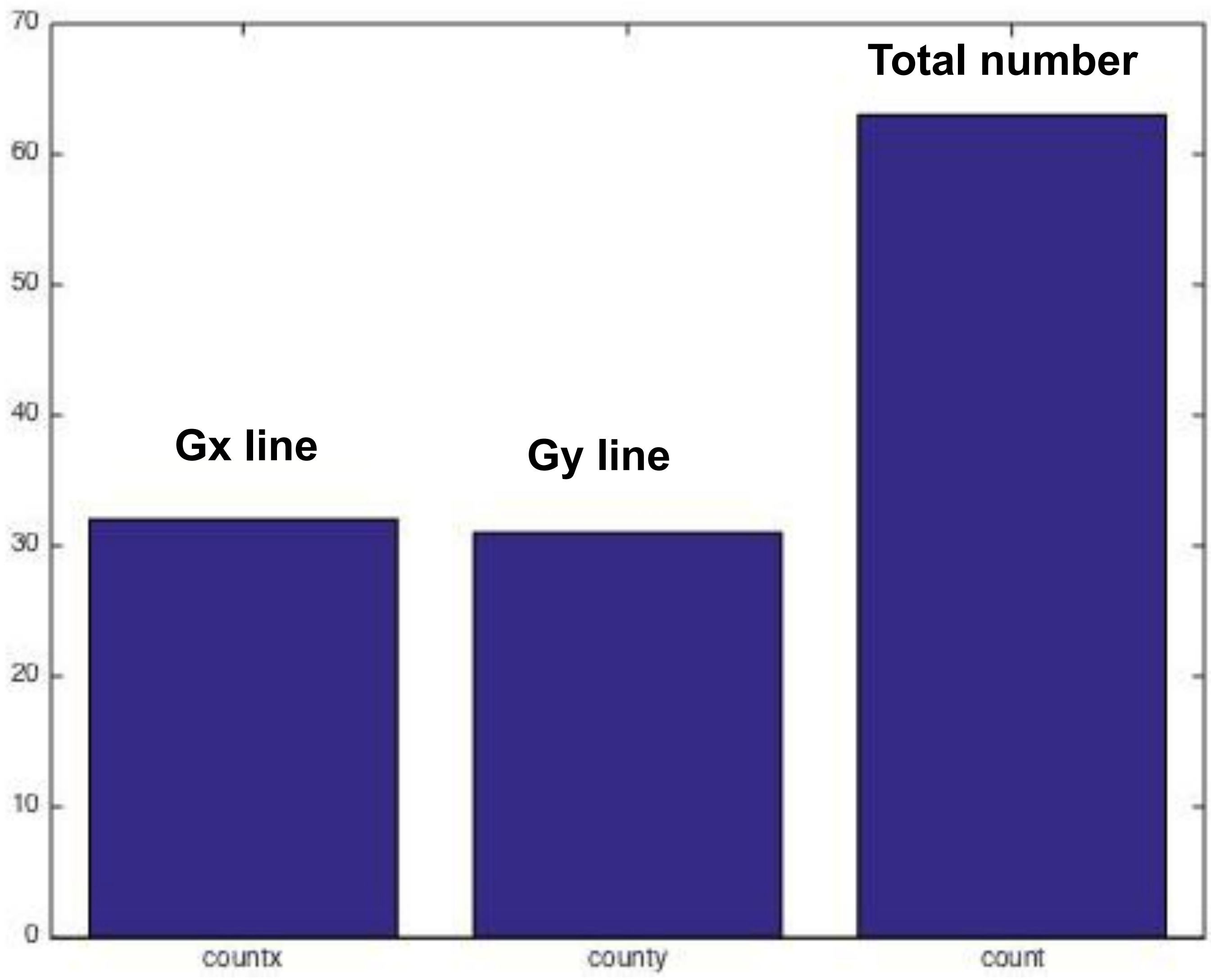


Gy



Results: Demo & Evaluation

Final statistics



Summary & Future Work

- We basically accomplished all of our goals. We remove all the objects not a parking line and remain most available parking lines.

The cost time is only width * height of the image. So we can process it very fast. That's feasiable.

- Shortcome

1.We do not check some lines that was coverred by a car. Like this:

2.Can not check diagonal parking line. Nowadays, we can only check horizentaland vertical lines.

Future work: We can apply it in the Google Map. It can show how many parking lots in a parking lot. It can show the statistics of parking lots every hours. It is an effective infomation for department of traffic and construction. They can show guide that which space is lacked of parking lots and when it is lacked parking lots to make a good decision to guide traffic and do construction of parking lot.

Appendix: assignment#2 and assigment#4 in DIP course. The other algorithm is created by ourselves.