Haonan Zhu

hz1396

N19549395

Computer Vision Project

**1.Process**

(1) Read image and detect edge using Sobel’s detector

Use Sobel’s detector to detect edge, get magnitude map.

(2) Normalize image

Use to normalize the grayscale value and let it lie on [0, 255]

(3) Do non-maxima suppression

Do non-maxima suppression to thin the edges.

(4) Produce binary edge map

Choose a threshold, if the grayscale of the pixel is smaller than the threshold, set it to 0, else set it to 255.

(5) Hough Transform

Do Hough Transform and get accumulator of .

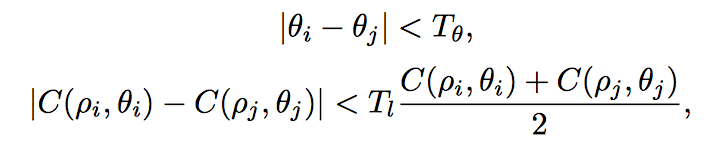
(6) Filter accumulator

Set a threshold(e.g. 0.25). If the votes of related is smaller than threshold \* maxVotes, filter out it.

(7) Merge lines

Since with little difference will generate same result, we needed to merge these lines. When the difference of and are below a threshold between lines, these lines will be merged. (e.g. difference is smaller than 2% of the range, diffrence is smaller than 3 degrees)

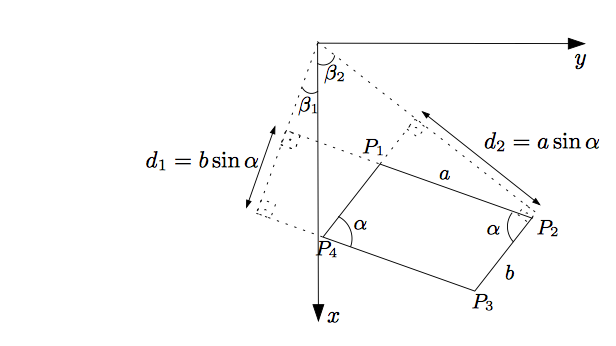
(8) Find parallel lines pair



is theta difference threshold between two lines. C is number of votes. is threshold of votes. e.g.

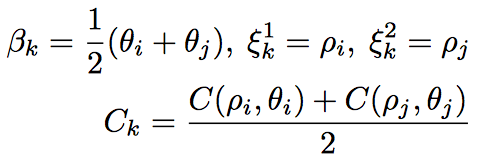
If two lines fits these two equations, then these two lines can form into a pair.

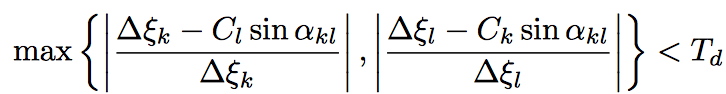
(9) Match parallel lines pairs



The vertical distances (ρ axis) between peaks within each pair are the orthogonal distances between parallel sides,

i.e., d 1 = |ρ 1 − ρ 2 | = b sin α and d 2 = |ρ 3 − ρ 4 | = a sin α. Thus, |ρ 1 − ρ 2 | = C(ρ 3 , θ 3 ) sin α = C(ρ 4 , θ 4 ) sin α and |ρ 3 − ρ 4 | = C(ρ 1 , θ 1 ) sin α = C(ρ 2 , θ 2 ) sin α.





Two pairs satisfying the conditions can form a parallelogram. is threshold. It should be approximately equal to .

(10) Get intersections of parallel line pairs

Solve .

Calculate the intersection of parallel line pairs. Filter out the intersections not in the image.

(11) Sort intersections by angles

Calculate the center point of four intersections and sort intersection based on the angle of the line generated by intersection and center point. It is for locating the sides of parallelogram. Just loop in order we will not meet diagram.

(12) Filter intersections

Calculate the fours sides’ length. If the shorted side’s length is smaller than 1/20 of image width, we filter out the intersections.

Check the side line of parallelogram based on two intersection points on binary edge map. Divide the line to many separate points. Record the points positions on the line. If one of the pixel on 3 x 3 range of the position on edge map is greater than 0, then add one on valid points. Finally we calculate validPoints / totalPoints on four side lines. If four sides percentages are all greater than a threshold value, we will pick these intersections.

(13) Merge close parallelograms

If parallelograms’ four intersections’ related distance are all below 10, there parallelograms will be merged into one.

(14) Plot parallelogram

Plot side lines of parallelogram based on the intersections. Just plot the intersection in order because we sorted it before.

**2. Programming language and instructions**

I use python as my programming language. Since I use different parameters on different figures, I attached three separate files for testing the figures. To test you just need to run the python files. Make sure that the origin figures are on the same directory of the python files.

**3. Codes**

Attached.

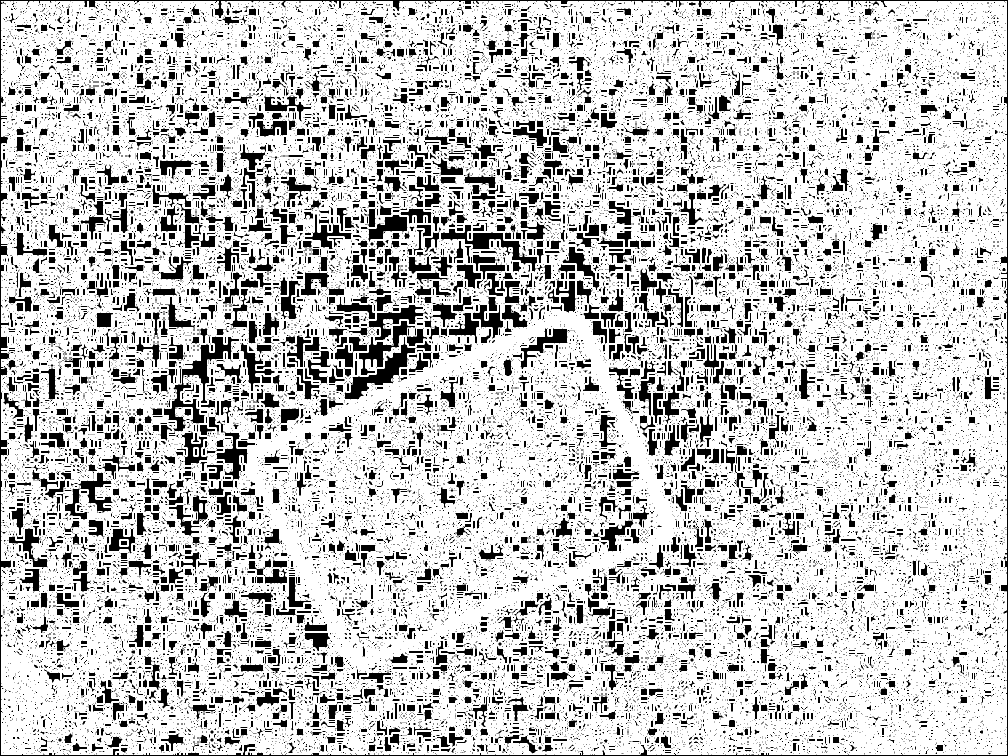
**4. Images**

(1) Figure 1

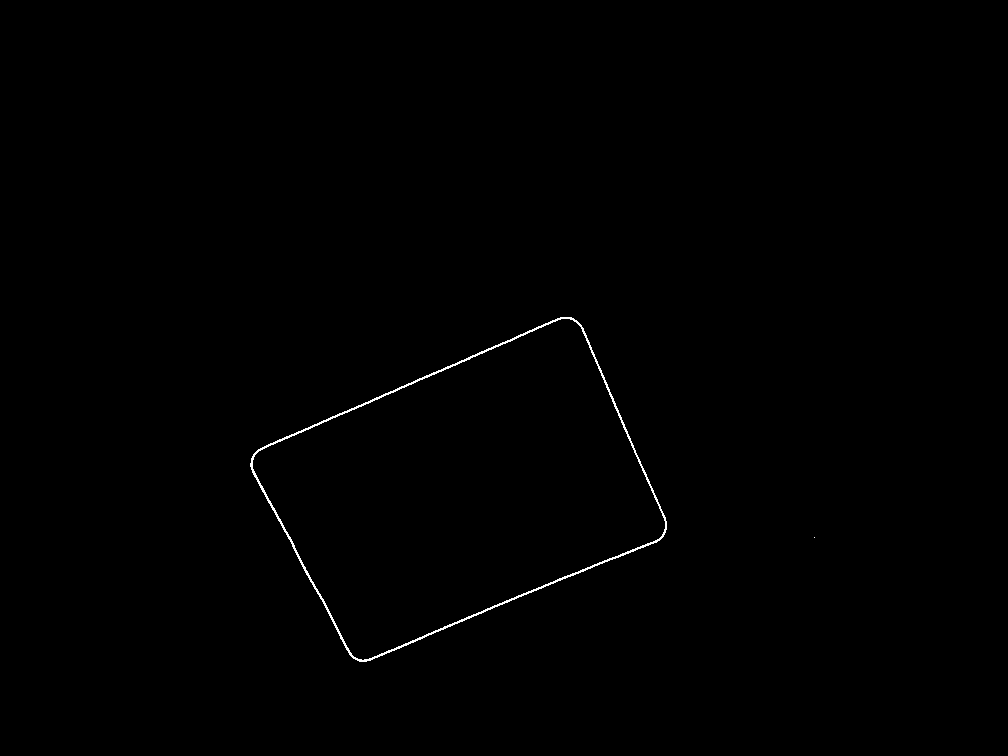
Original image



Normalized gradient magnitude



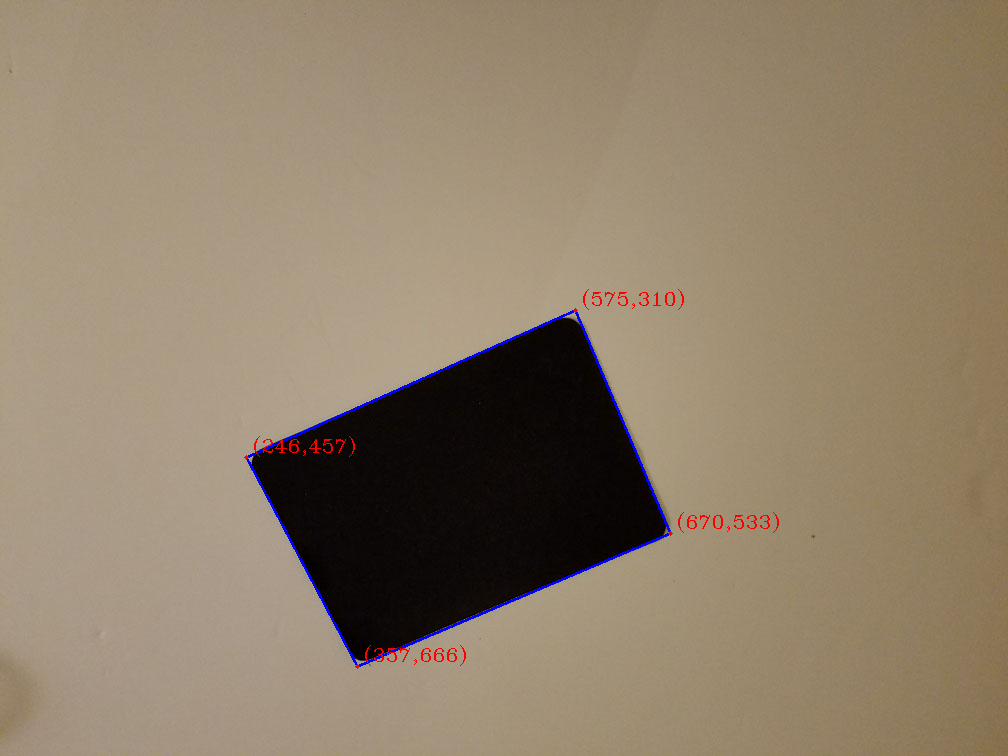
Edge map after thresholding



The threshold used is 78.

Four corners of the parallelogram: [246, 457], [357, 666], [670, 533], [575, 310]

Output image

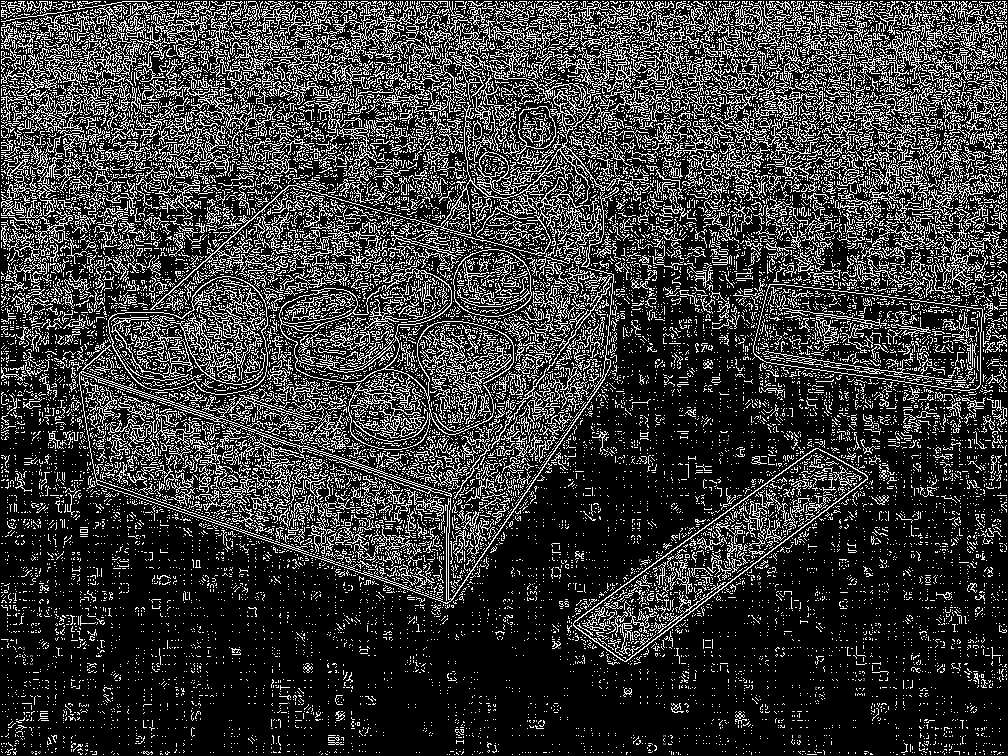


(2) Figure 2

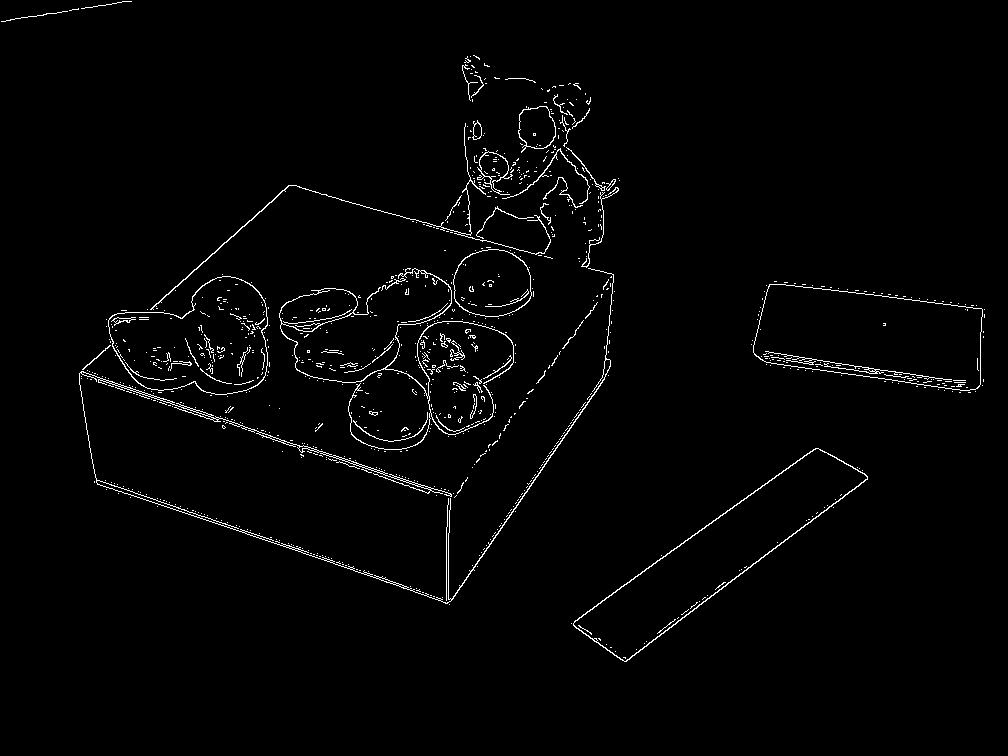
Original image



Normalized gradient magnitude



Edge map after thresholding



I used non-maxima suppression before thresholding.

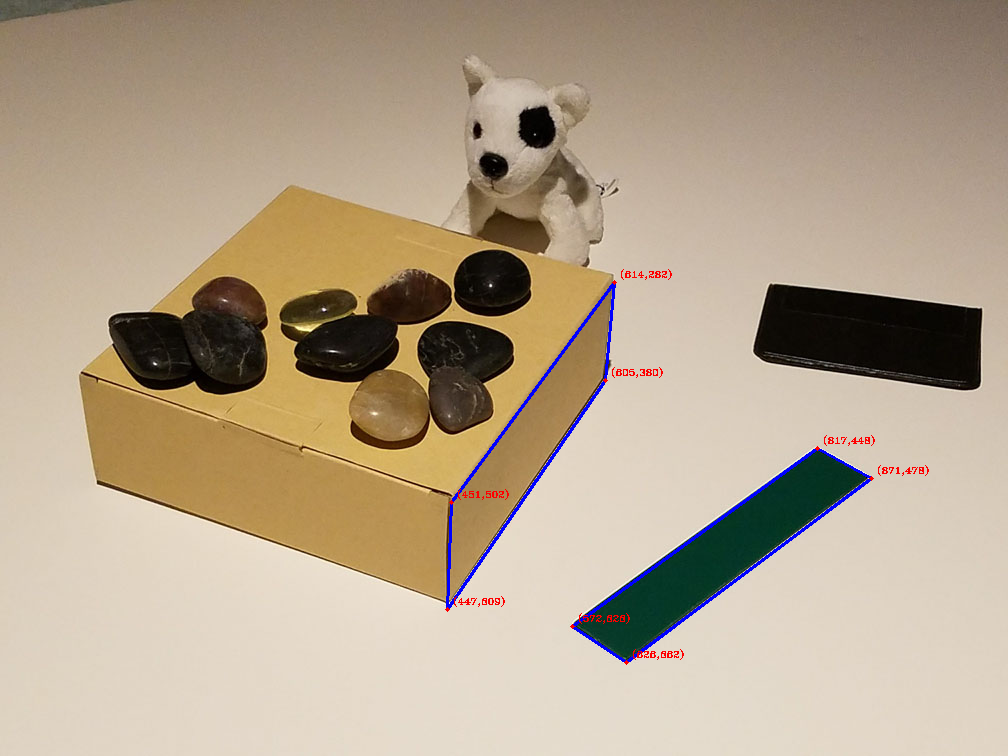
The threshold used is 22.

Four corners of the parallelogram:

[572, 626], [626, 662], [871, 478], [817, 448]

[451, 502], [447, 609], [605, 380], [614, 282]

Output image

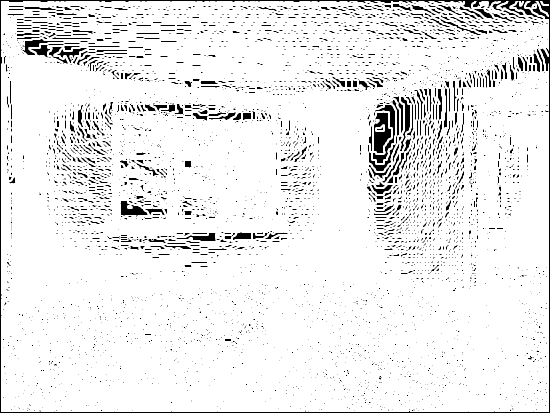


(3) Figure 3

original image



Normalized gradient magnitude



Edge map after thresholding



The threshold is 20.

Four corners of the parallelogram:

[115, 115], [117, 225], [230, 227], [230, 123]

[115, 115], [117, 225], [279, 228], [279, 127]

[116, 115], [114, 225], [230, 227], [230, 123]

[116, 115], [114, 225], [279, 228], [279, 127]

[170, 119], [168, 226], [279, 228], [279, 127]

Output image

