Introductuion

This program implemented some basic image processing algorithms and putting them together to build a Hough Transform based line detector. The program is able to find the start and end points of straight line segments in images. Like most vision algorithms, the Hough Transform uses a number of parameters whose optimal values are data dependent (i.e., a set of parameter values that works really well on one image might not be best for another image). By running code and tuning parameters on the test images, it obtained optima value for each image which gives a good performance.

Implementation

3.1 Convolution (15 points)

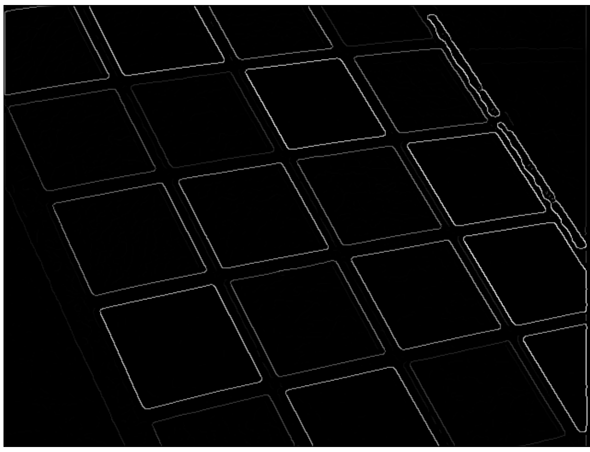
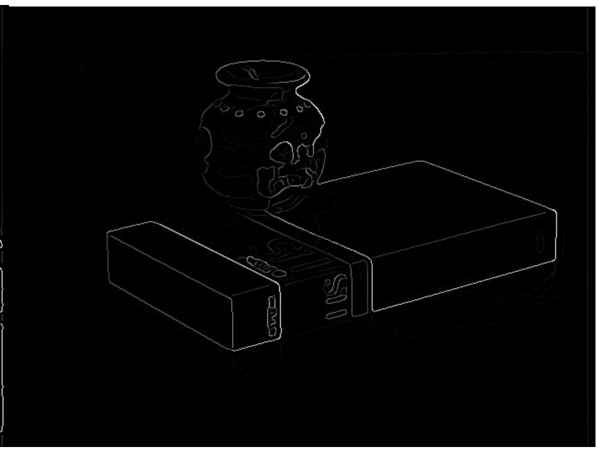
Write a function that convolves an image with a given convolution filter

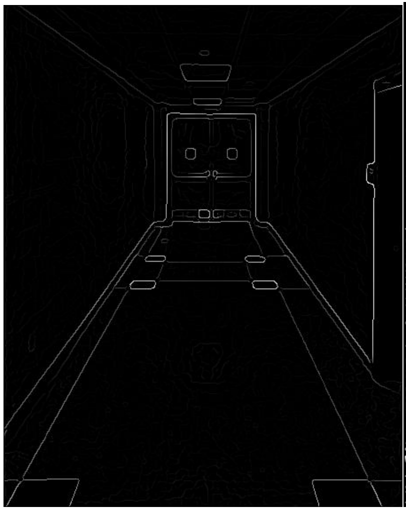
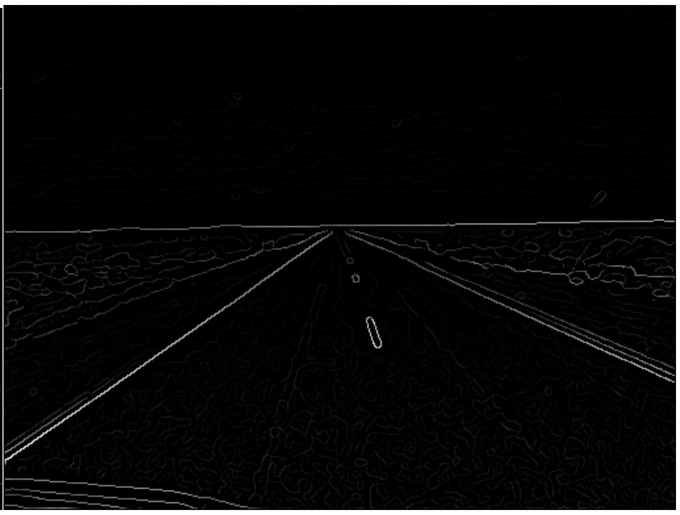
```
function [img1] = myImageFilter(img0, h)
```

As input, the function takes a greyscale image (`img0`) and a convolution filter stored in matrix `h`. The output of the function should be an image `img1` of the same size as `img0` which results from convolving `img0` with `h`. You can assume that the filter `h` is odd sized along both dimensions. You will need to handle boundary cases on the edges of the image. For example, when you place a convolution mask on the top left corner of the image, most of the filter mask will lie outside the image. One possible solution is to pad the image such that pixels lying outside the image boundary have the same intensity value as the nearest pixel that lies inside the image.

Q3.1

See as. m file.

HEINZ
Brillo
Campbell's

3.2 Convolution with One *for* Loop (extra credit: 10 points)

Please write a function that does convolution with only one for loop and save it to `ec/` directory. (If you already do so in `Q3.1`, good job, just copy one and save it to `ec/`.) Also, briefly describe how you implement it in your write-up. Illustrations helpful for understanding is highly encouraged.

```
    function [img1] = myImageFilterX(img0, h)
```

Q3.2

In order to use only one loop, I just transferred the 2D array into a 1D array, then using figured out what is the mapping form of them. Other codes are quite the same as Q3.1. See .m file under folder ec.

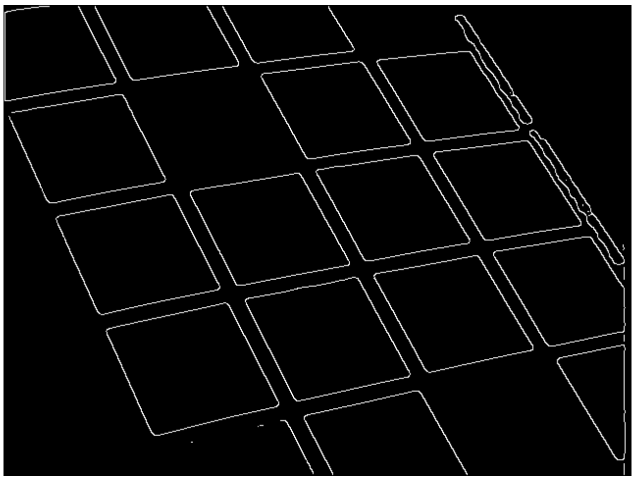
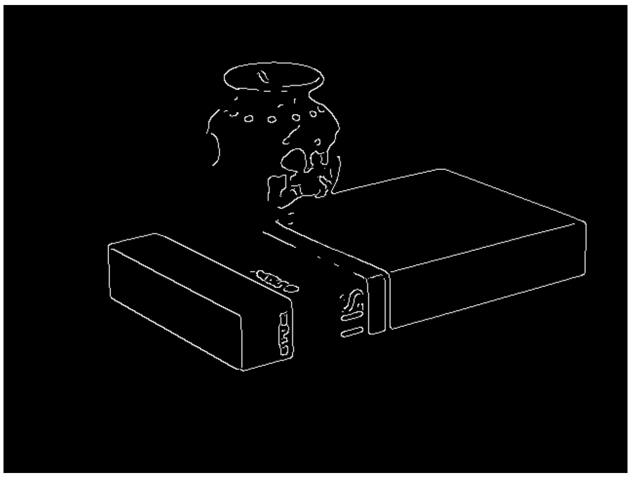
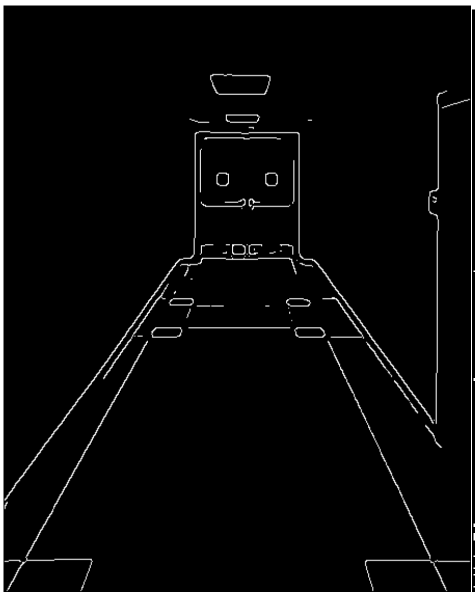
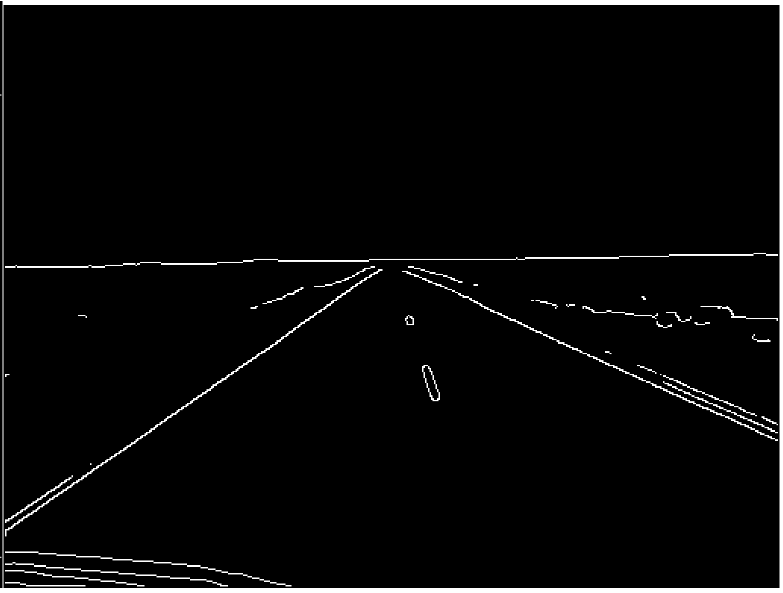
3.3 Edge Detection (15 points)

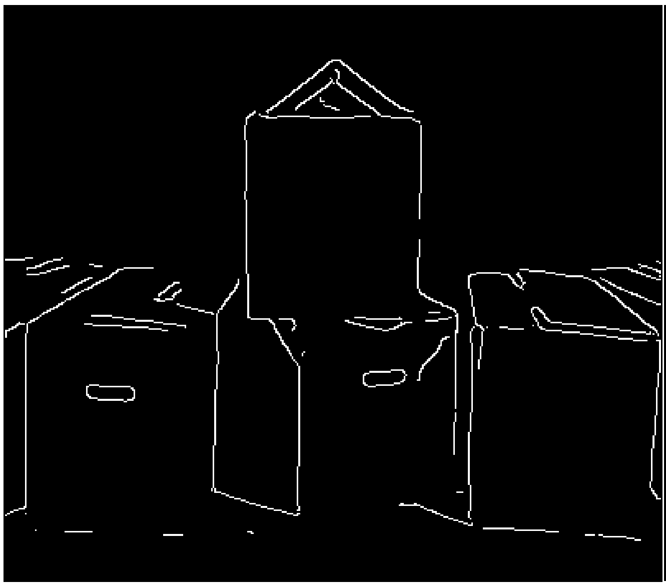
Write a function that finds edge intensity and orientation in an image. Include the output of your function for one of the given images in your writeup.

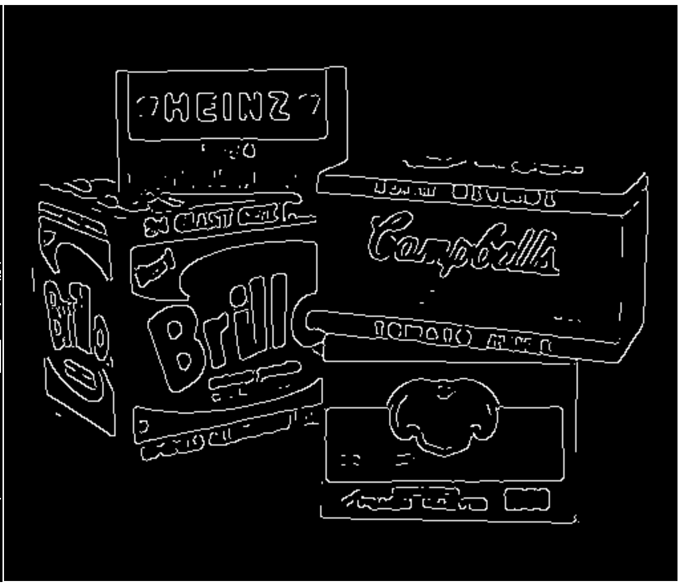
```
function [Im Io Ix Iy] = myEdgeFilter(img, sigma)
```

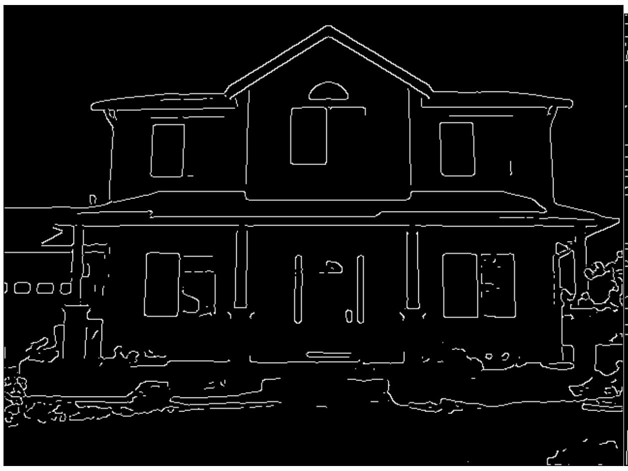
The function will input a greyscale image (`img`) and `sigma` (scalar). `sigma` is the standard deviation of the Gaussian smoothing kernel to be used before edge detection. The function will output `Im`, the edge magnitude image; `Io` the edge orientation image and `Ix` and `Iy` which are the edge filter responses in the x and y directions respectively.

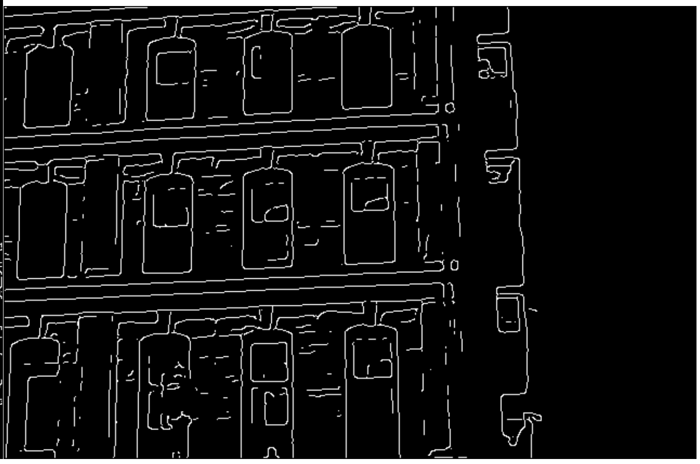
Q3.3

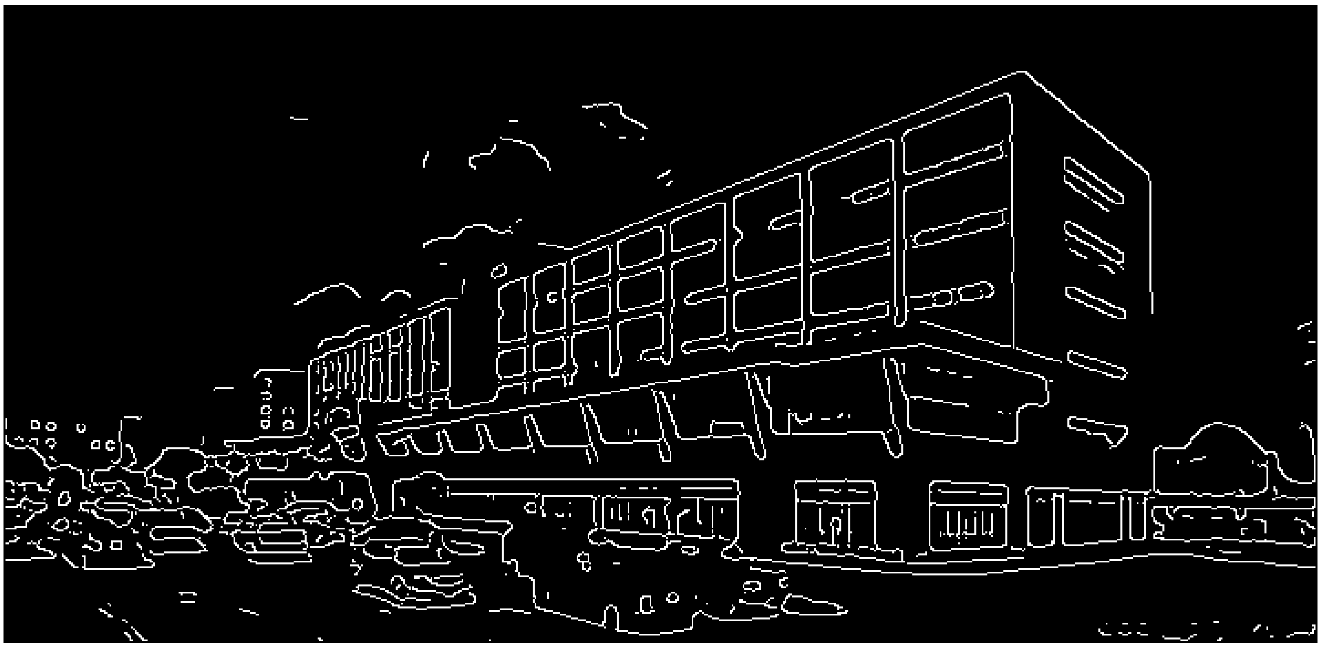








HEINZ
Brillo
Campbell's







3.4 Hough Transform (20 points)

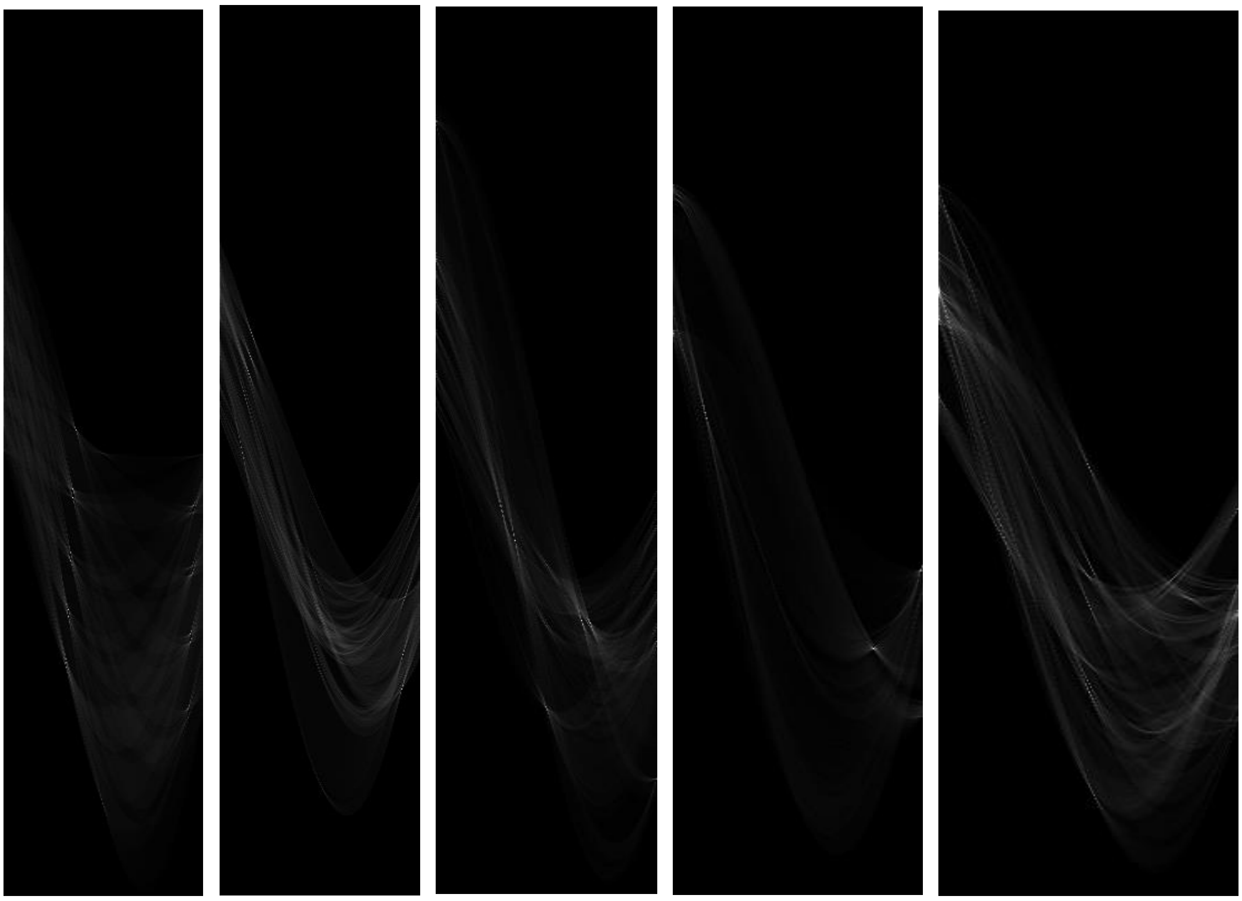
Write a function that applies the Hough Transform to an edge magnitude image.

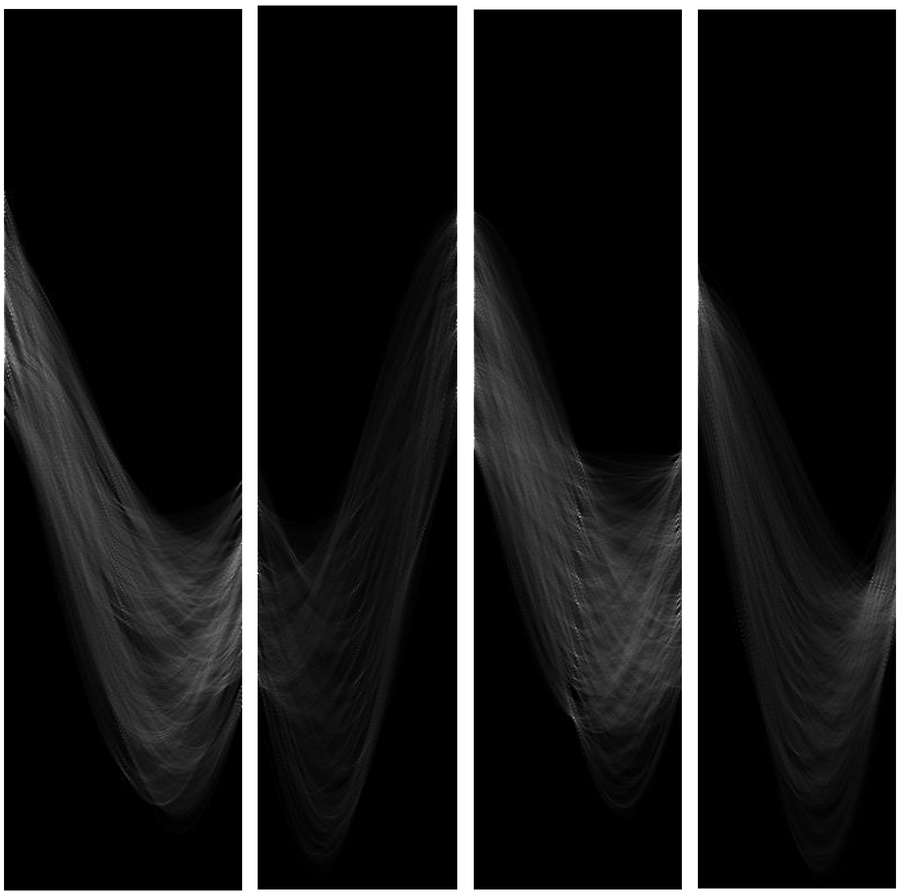
```
function [H, rhoScale, thetaScale] = myHoughTransform(Im, threshold, rhoRes, thetaRes)
```

`Im` is the edge magnitude image, `threshold` (scalar) is an edge strength threshold used to ignore pixels with a low edge filter response. `rhoRes` (scalar) and `thetaRes` (scalar) are the resolution of the Hough transform accumulator along the ρ and θ axes respectively. For example, if `thetaRes` = 5° and $\theta \in [0, 360°]$, then number of bins along θ axis is $360/5 = 72$. `H` is the Hough transform accumulator that contains the number of 'votes' for all the possible lines passing through the image. `rhoScale` and `thetaScale` are the arrays of ρ and θ values over which `myHoughTransform` generates the Hough transform matrix `H`. For example, if `rhoScale(i)` = ρ_i and `thetaScale(j)` = θ_j, then `H(i,j)` contains the votes for $\rho = \rho_i$ and $\theta = \theta_j$.

Q3.4

From image 1 to 9 (left to right)



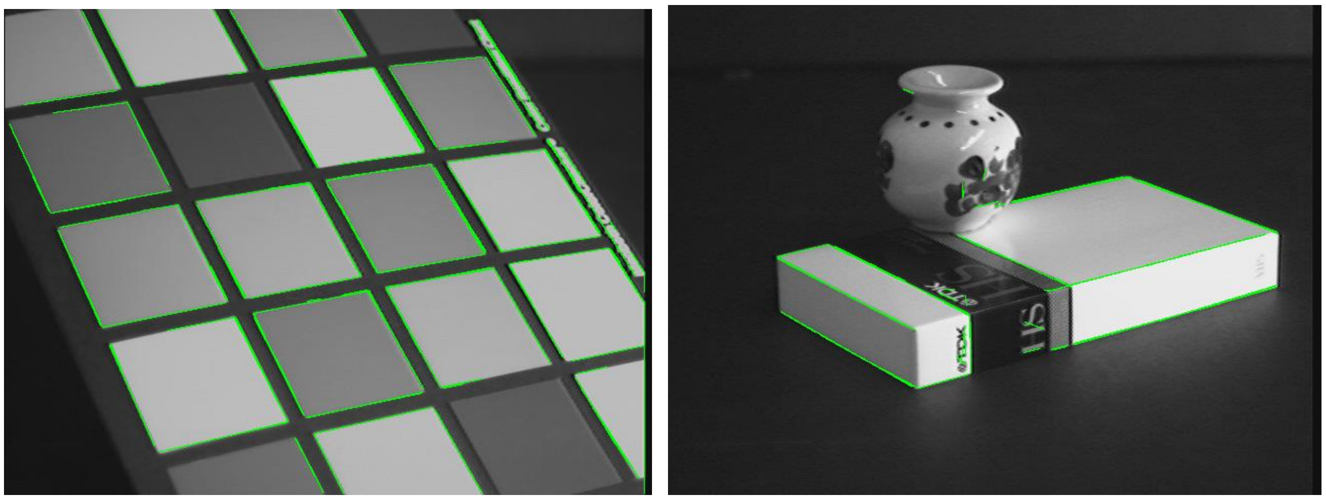


3.5 Finding Lines (15 points)

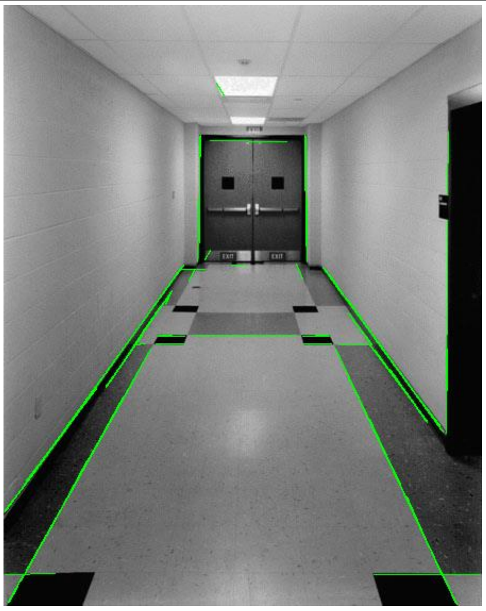
```
function [lineRho lineTheta] = myHoughLines(H, nLines)
```

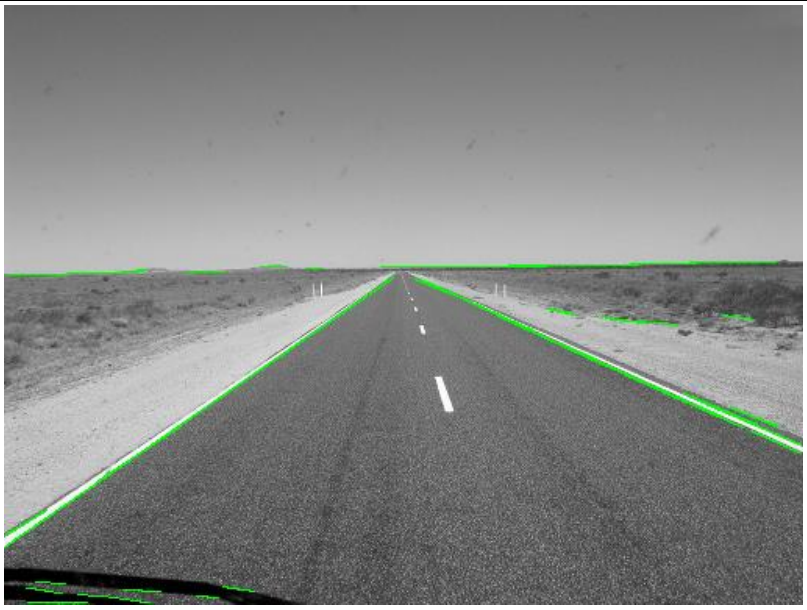
`H` is the Hough transform accumulator; `rhoRes` and `thetaRes` are the accumulator resolution parameters and `nLines` is the number of lines to return. Outputs `lineRho` and `lineTheta` are both `nLines` $\times$ 1 vectors that contain the parameters (ρ and θ respectively) of the lines found in an image.

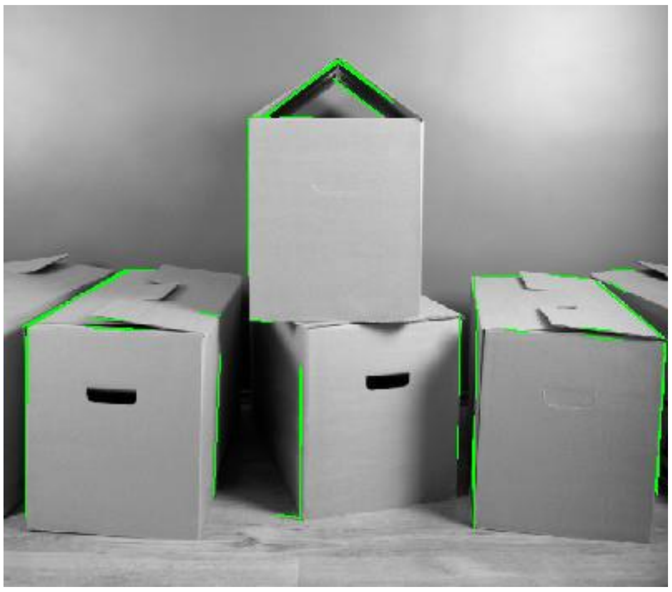
Q3.5 (WITH GIVEN PARAMETERS)

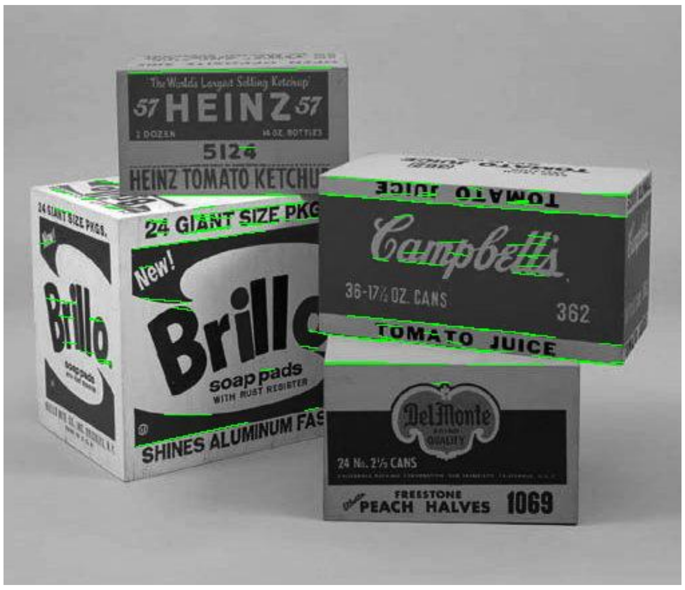


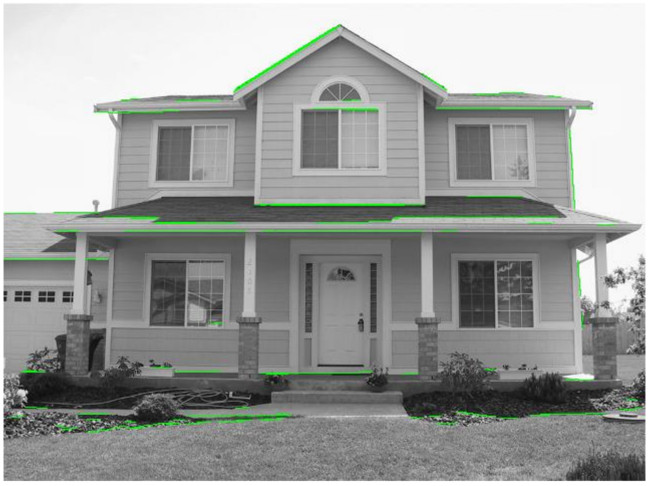
Name: Jiayi Wang ID: jiayiw2

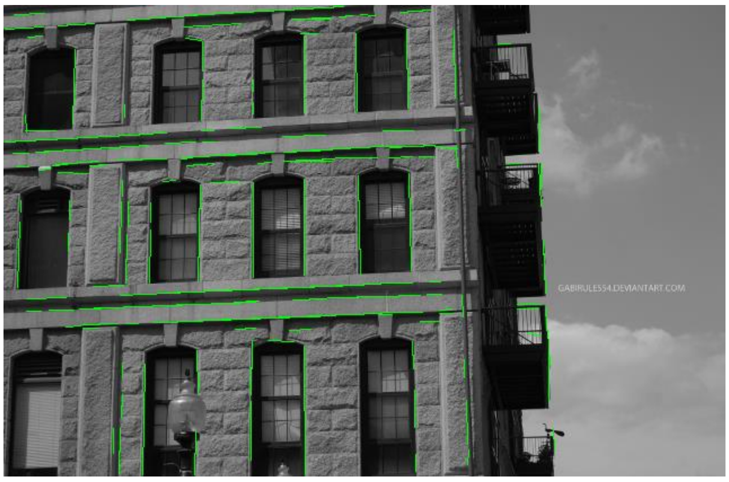


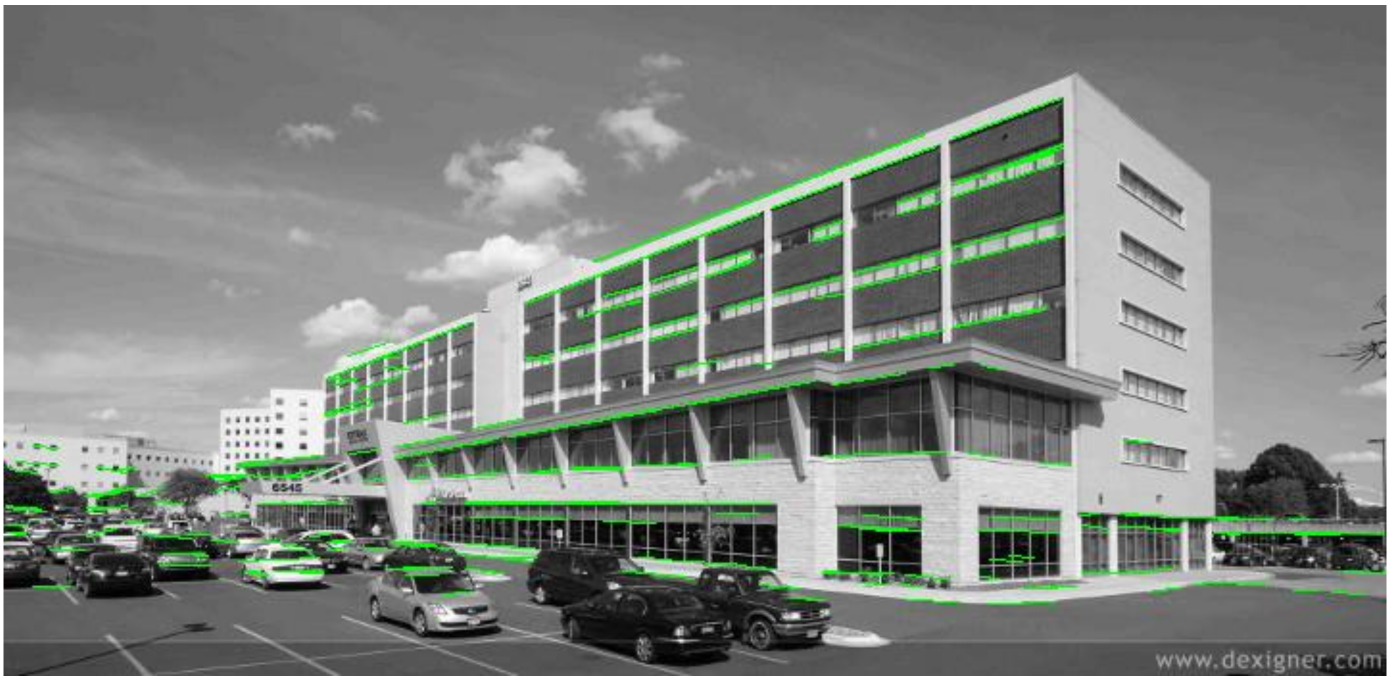








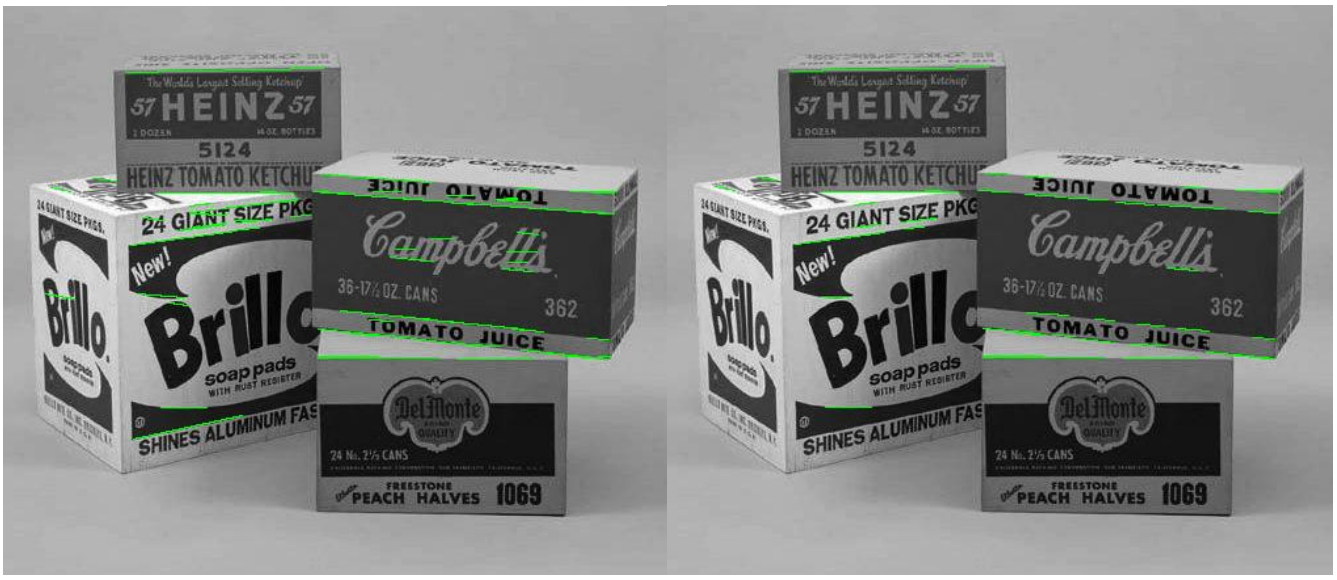



6546
www.dexigner.com

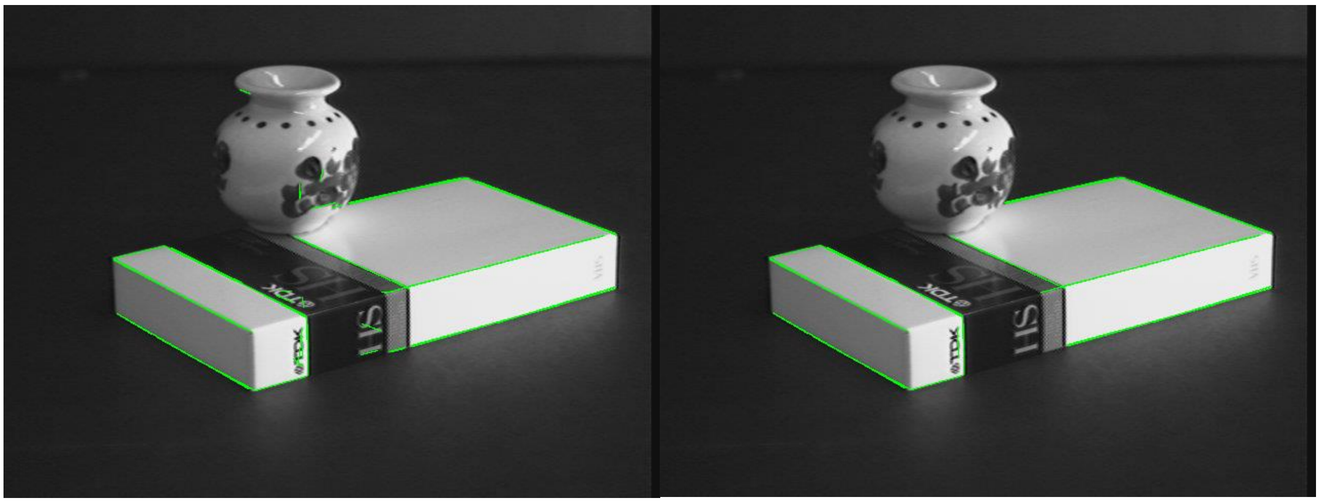
Q3.6

(1) Fillgap: If there's discontinuous lines which should be joint together, then set fillgap to a relatively high value. On the other hand, if there's lines near each other which should not be connected, then set fillgap to a small value like what we did in image 6.
(2) Minlength: if the image only has long lines instead of couple of short lines, then minlength should be large, vise versa.

Changing 'fillgap':
Fillgap = 5 (left) and fillgap = 2 (right), we can see here the letters on the box should not be connected, under this condition, decreasing fillgap to 2 will give a better result.

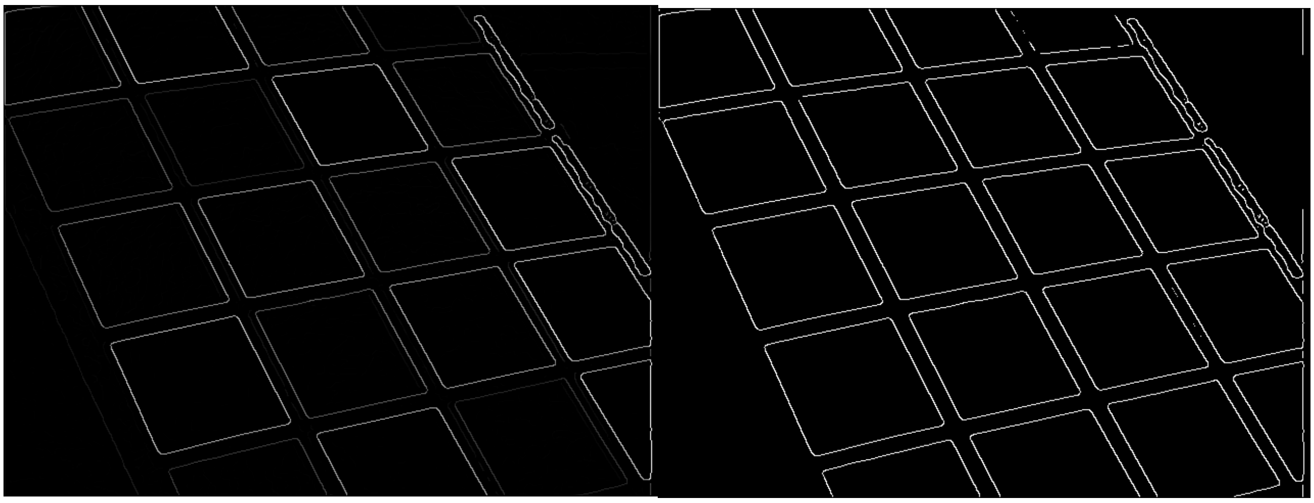


Changing 'minlength' :
Minlength = 10(left) and minlength = 30(right), by increasing minimum length of the line can eliminate the short noise lines in the picture. Circumstances varies according to different pictures.



Q4 Experiment and Discussion (best perform parameters)

Image 1 --- Parameters:
Sigma = 2; threshold = 0.017;
rhoRes = 2; thetaRes = pi/180;
nLines = 50; 'FillGap' = 5; 'MinLength' = 15;



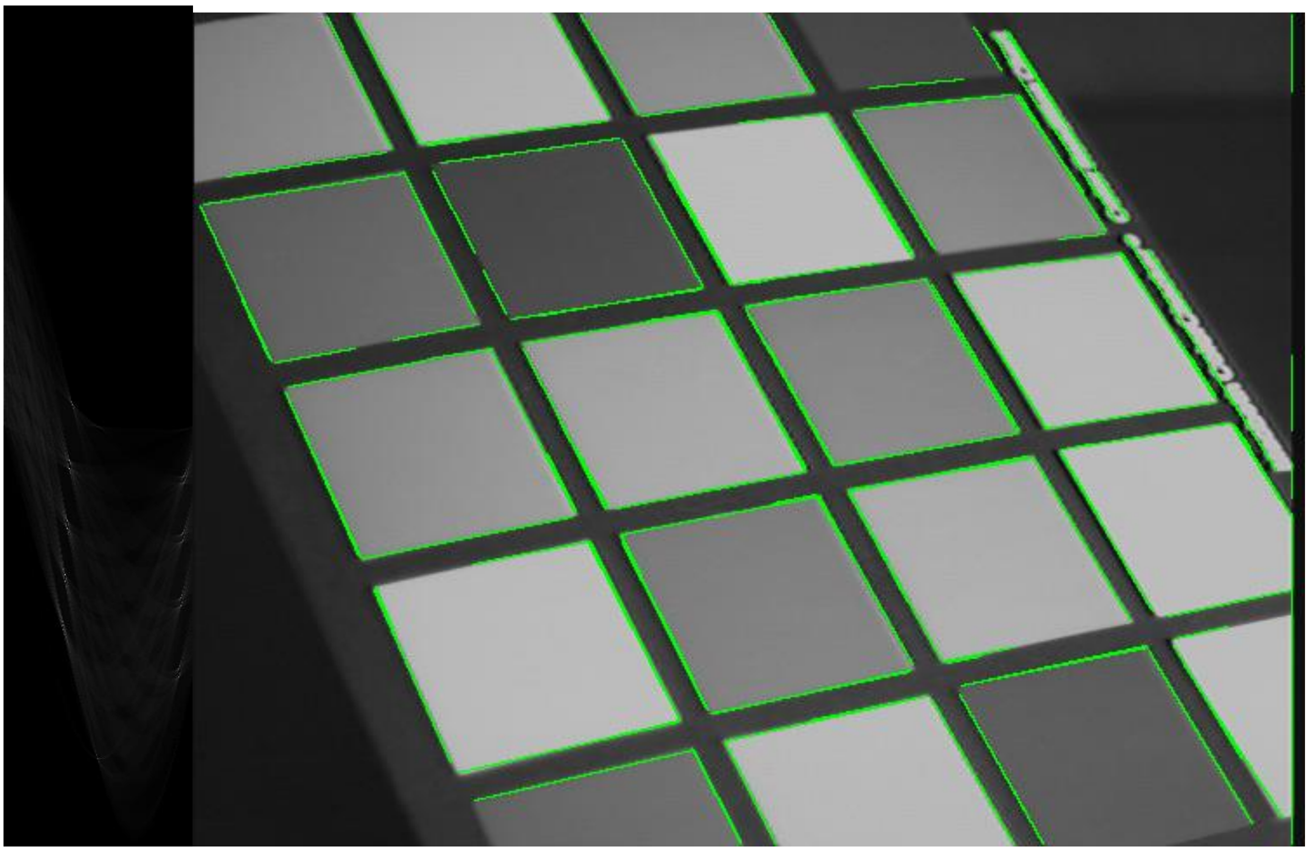


Image 2 --- Parameters:
Sigma = 2; threshold = 0.02;
rhoRes = 2; thetaRes = pi/180;
nLines = 50; 'FillGap' = 5; 'MinLength' = 15;

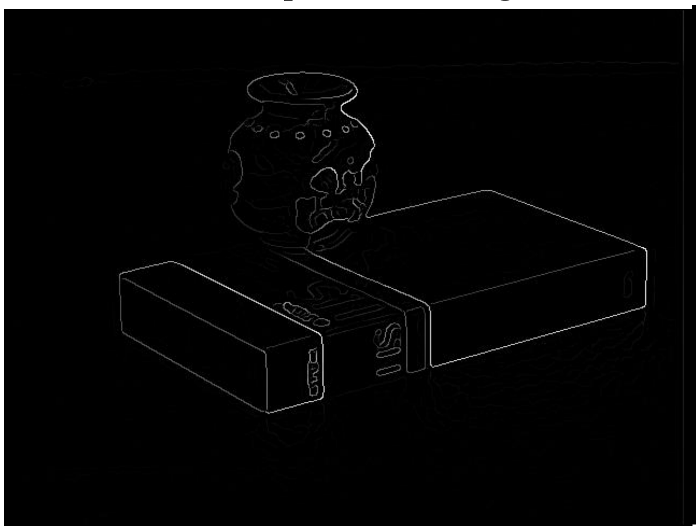
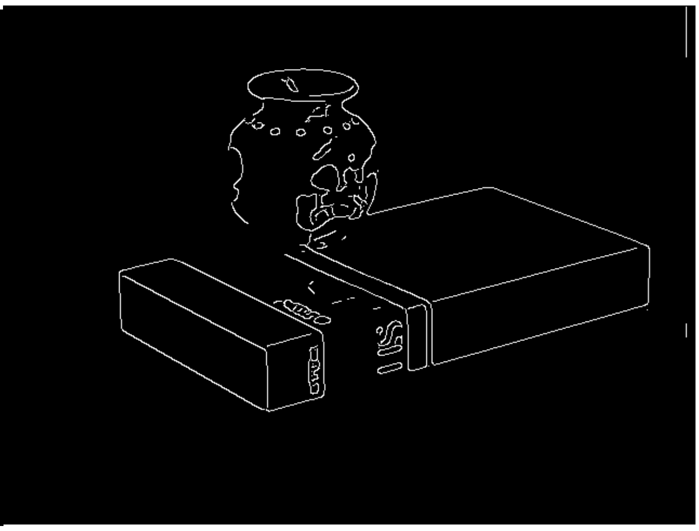
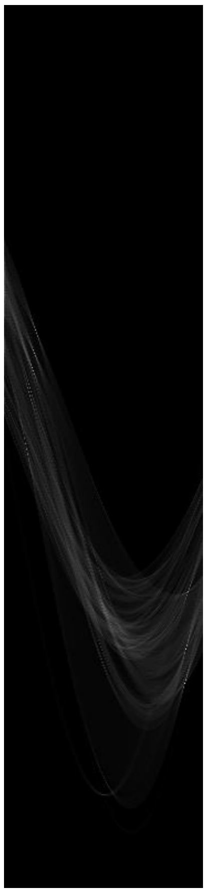
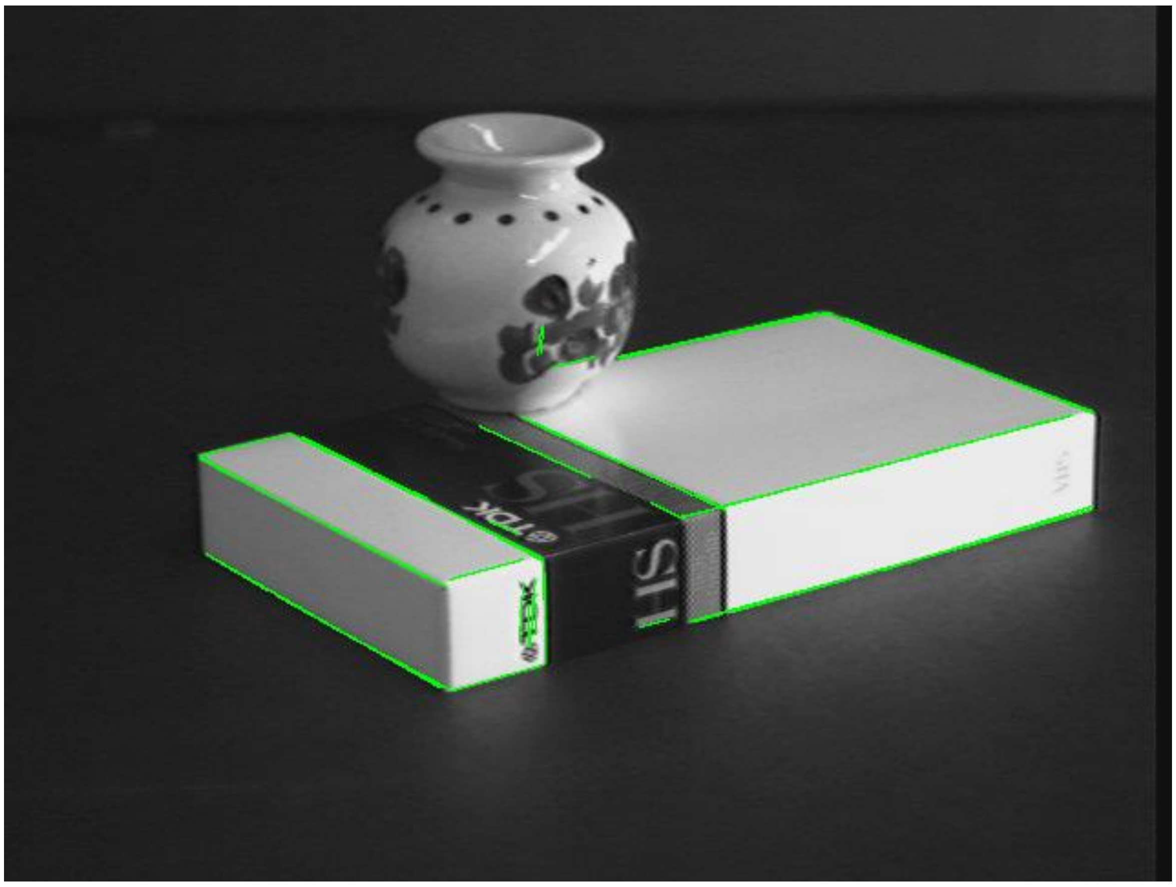
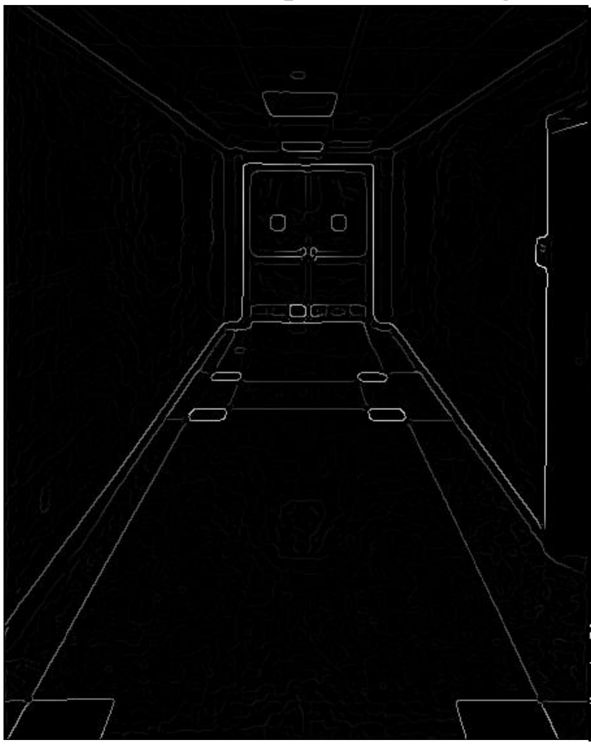
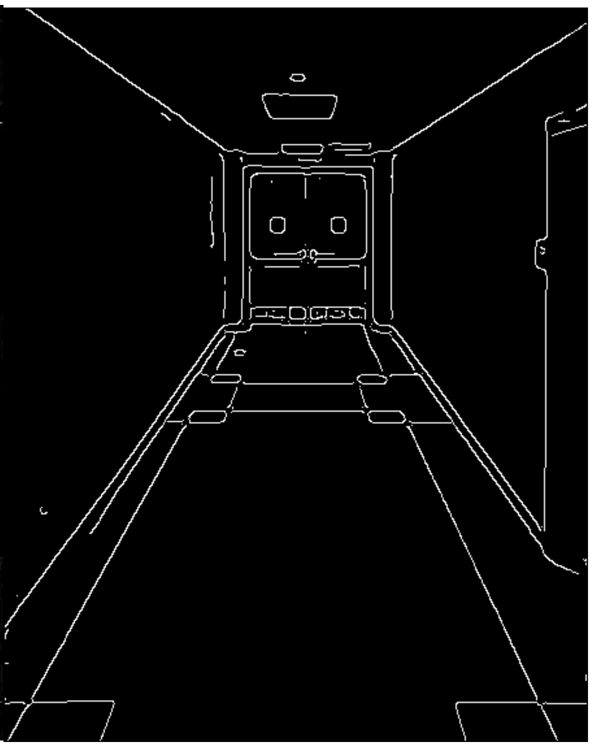
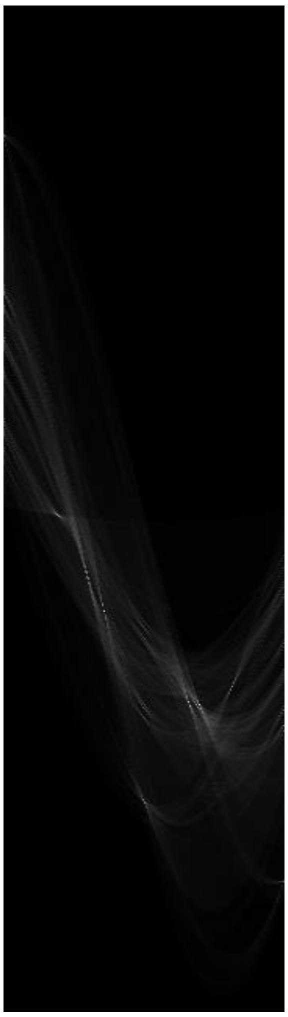





Image 3 --- Parameters:
Sigma = 2; threshold = 0.014;
rhoRes = 2; thetaRes = pi/180;
nLines = 60; 'FillGap' = 5; 'MinLength' = 15;







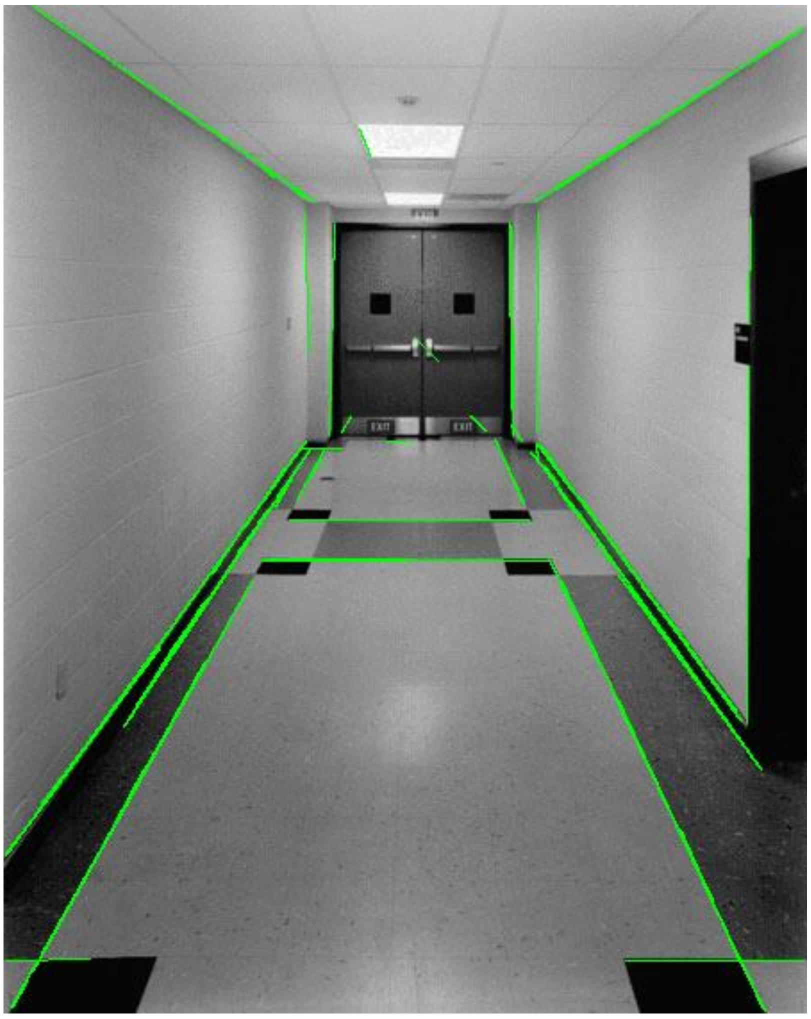
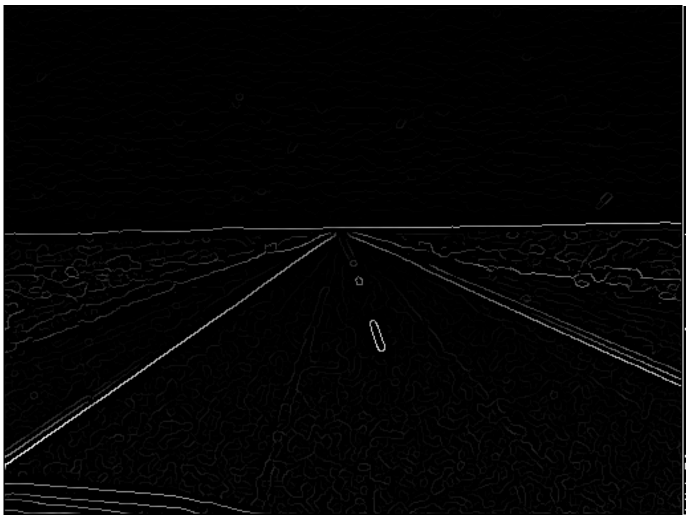
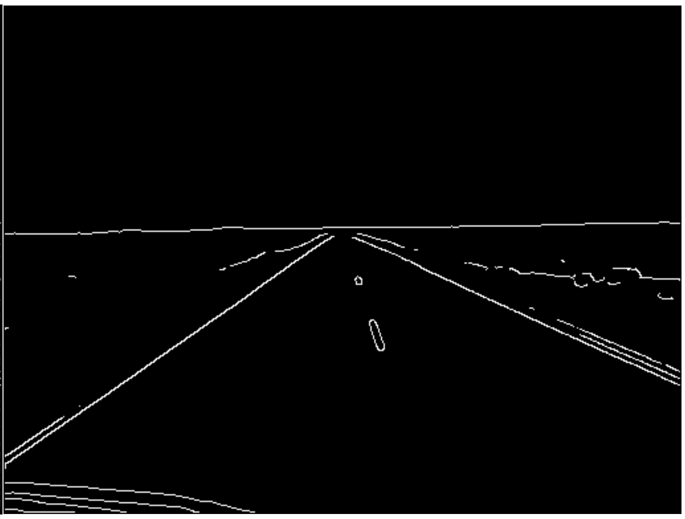


Image 4 --- Parameters:
Sigma = 2; threshold = 0.03;
rhoRes = 2; thetaRes = pi/180;
nLines = 30; 'FillGap' = 5; 'MinLength' = 15;

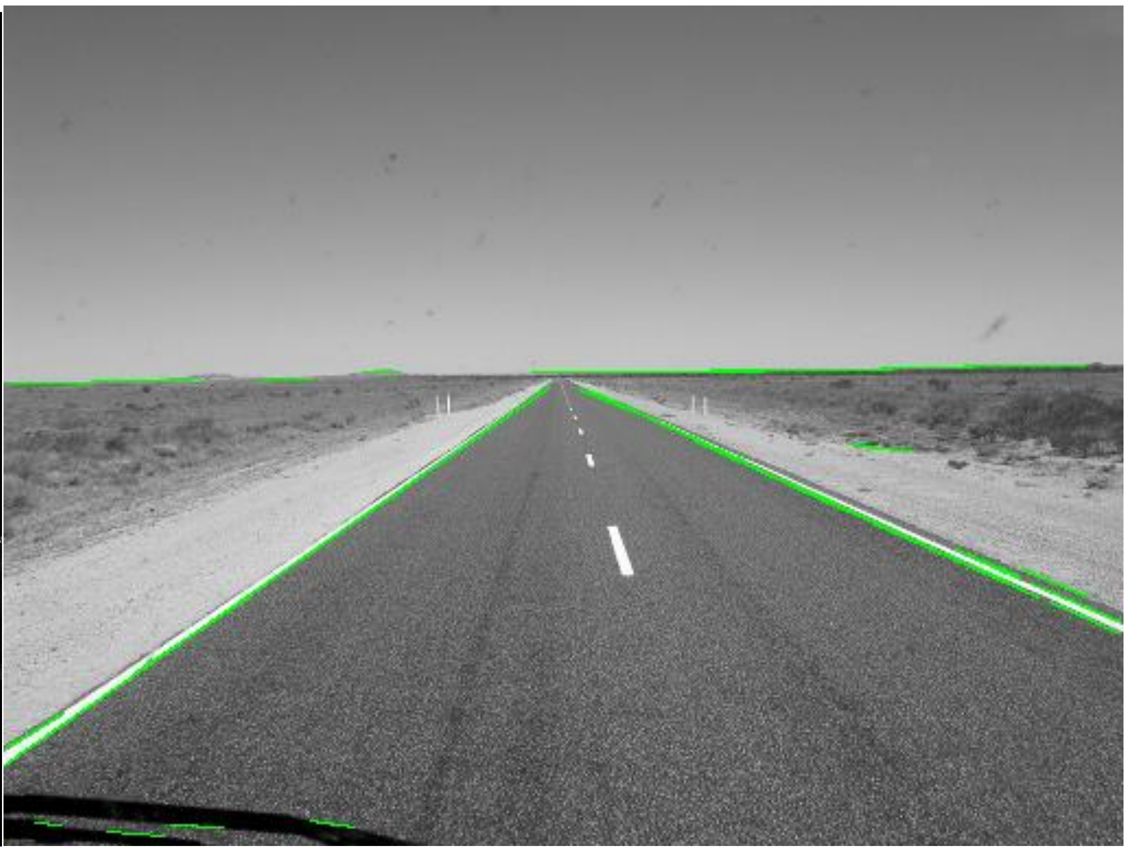
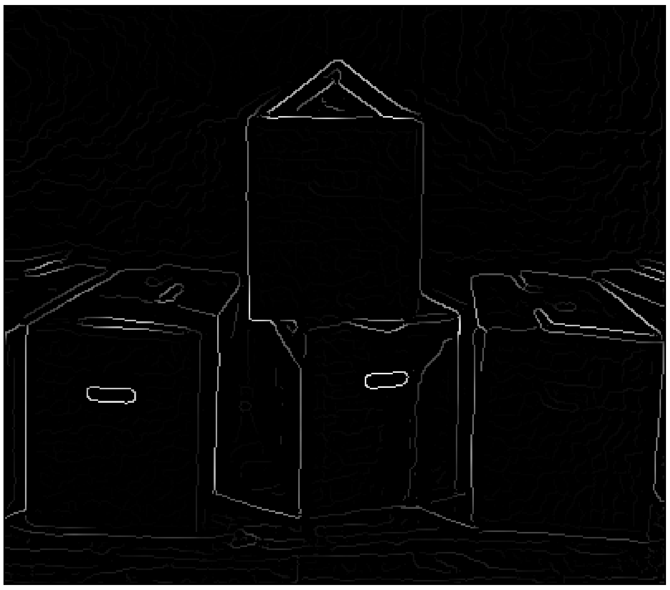
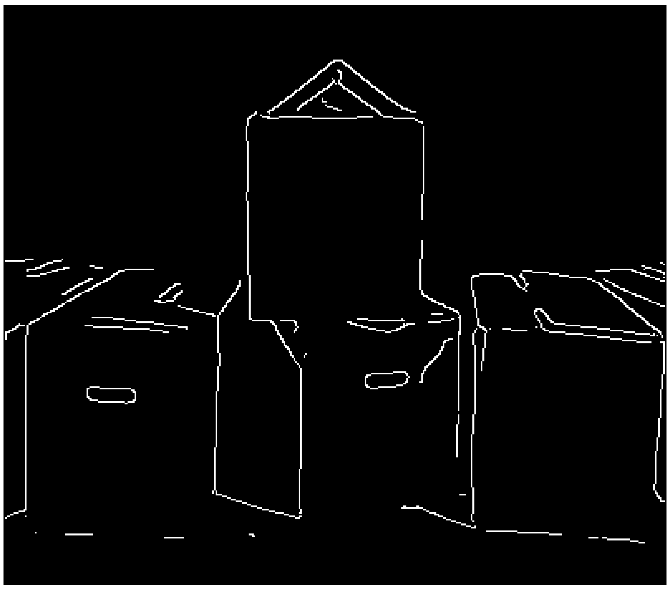


Image 5 --- Parameters:
Sigma = 2; threshold = 0.03;
rhoRes = 2; thetaRes = pi/180;
nLines = 100; 'FillGap' = 5; 'MinLength' = 10;

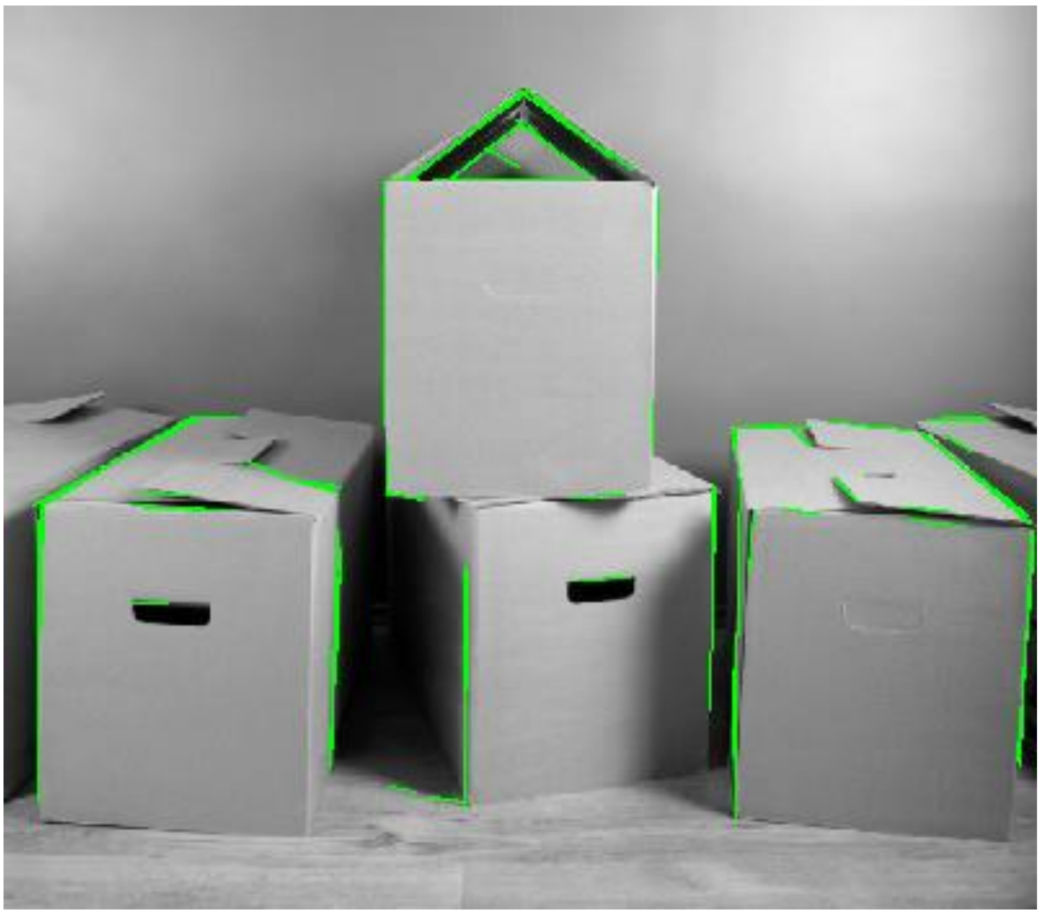
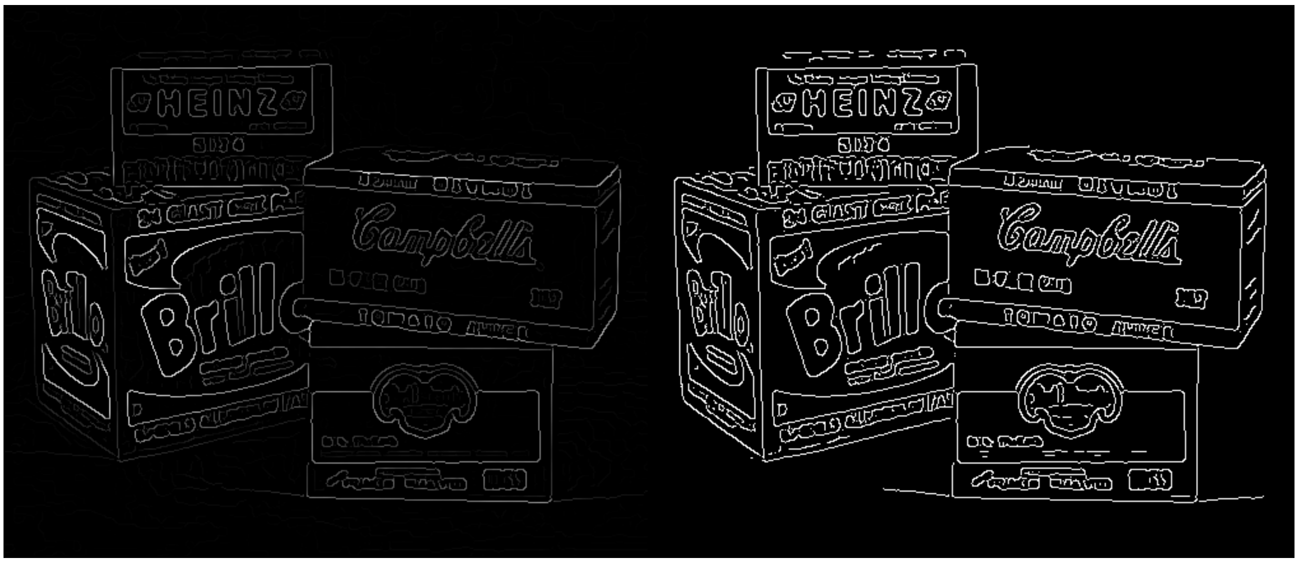


Image 6 --- Parameters:

Sigma = 2; threshold = 0.01;

rhoRes = 2; thetaRes = pi/180;

nLines = 100; 'FillGap' = 3; 'MinLength' = 25;



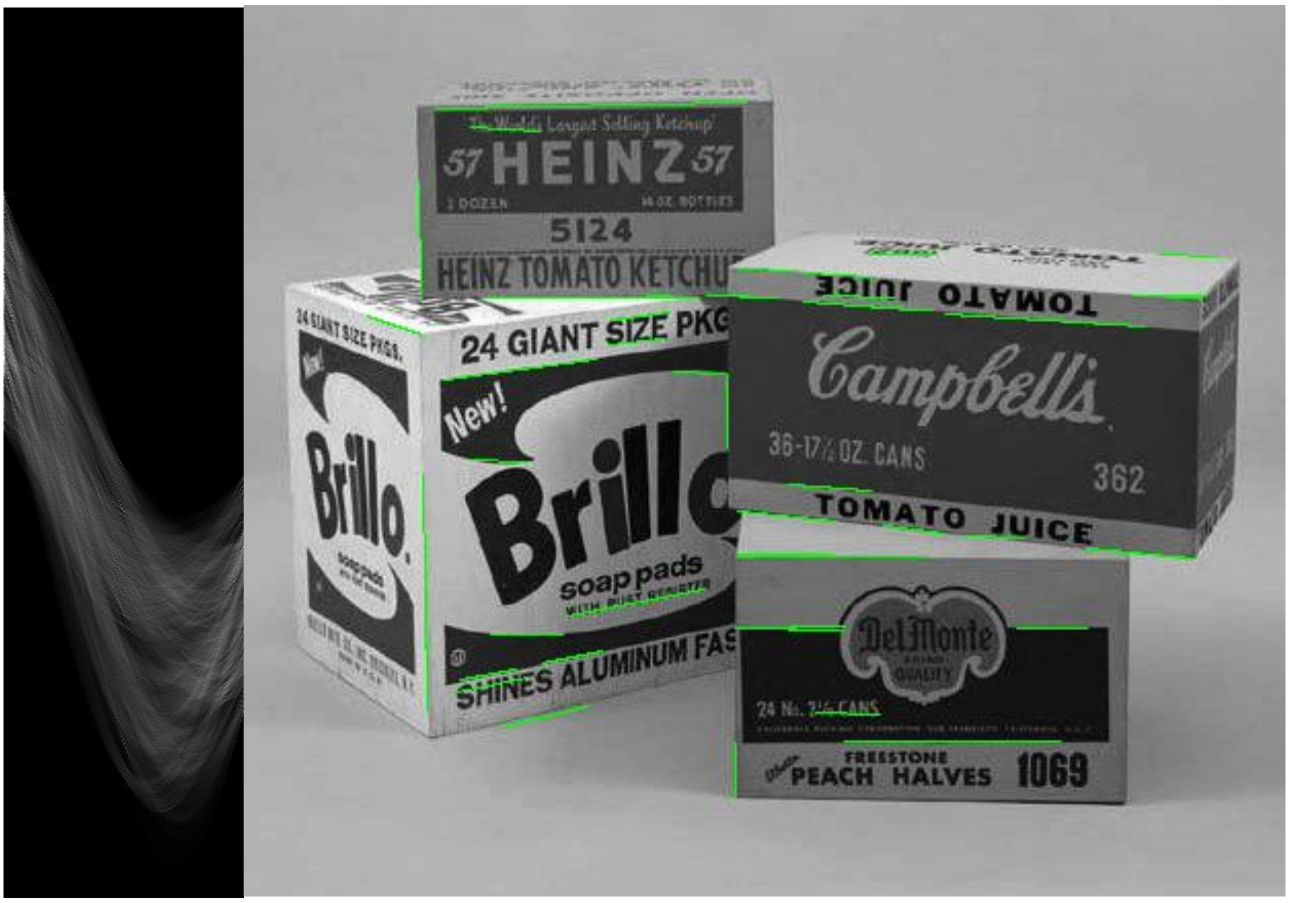
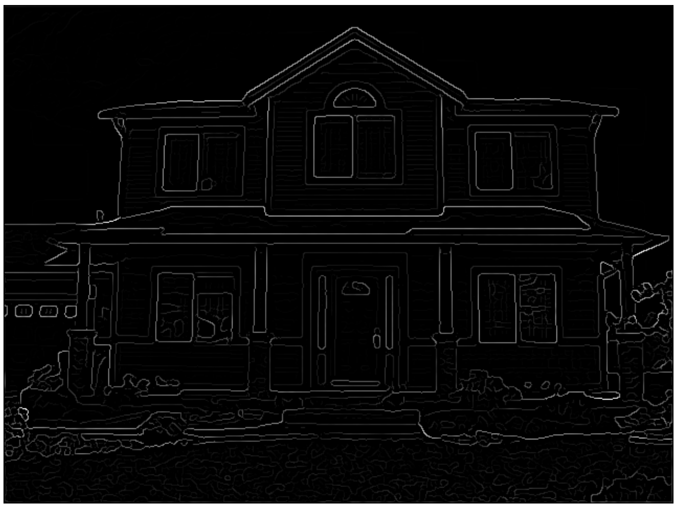
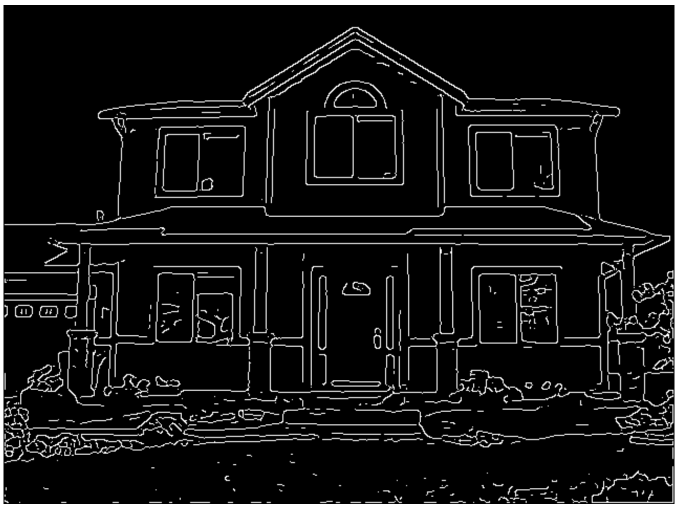


Image 7 --- Parameters:
Sigma = 2; threshold = 0.015;
rhoRes = 2; thetaRes = pi/180;
nLines = 180; 'FillGap' = 5; 'MinLength' = 20;

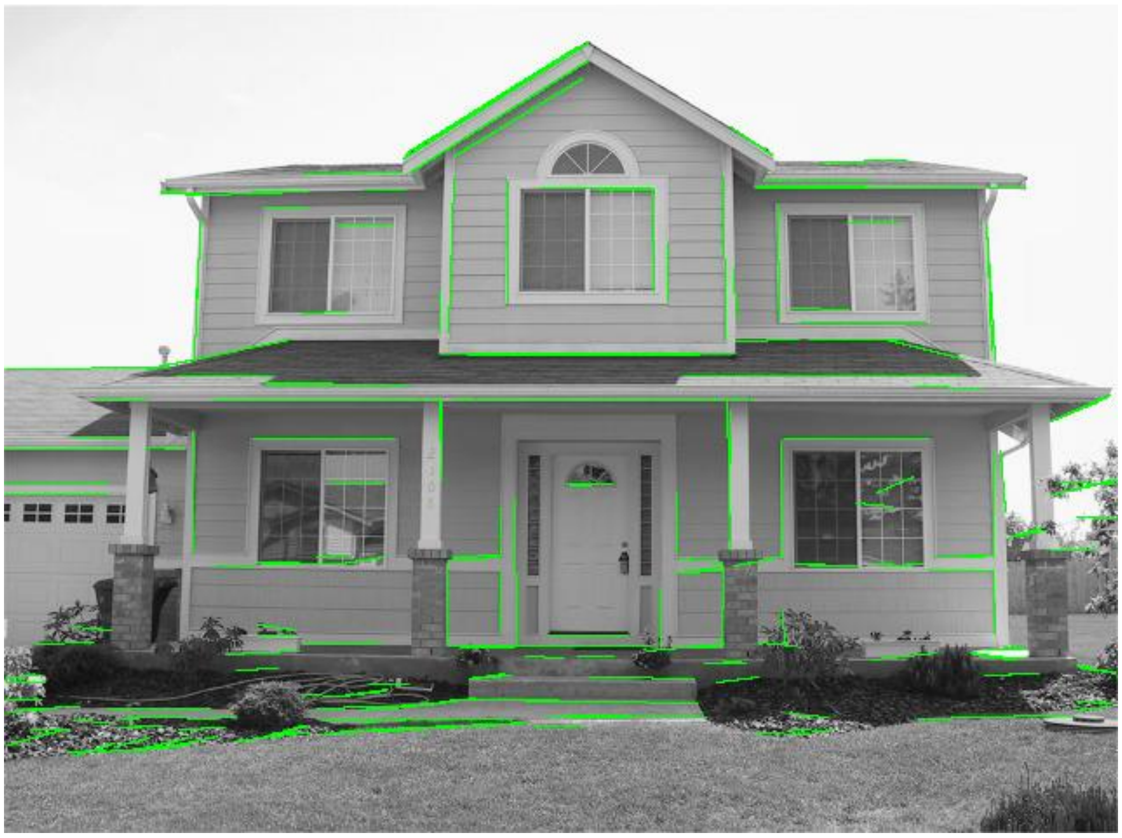
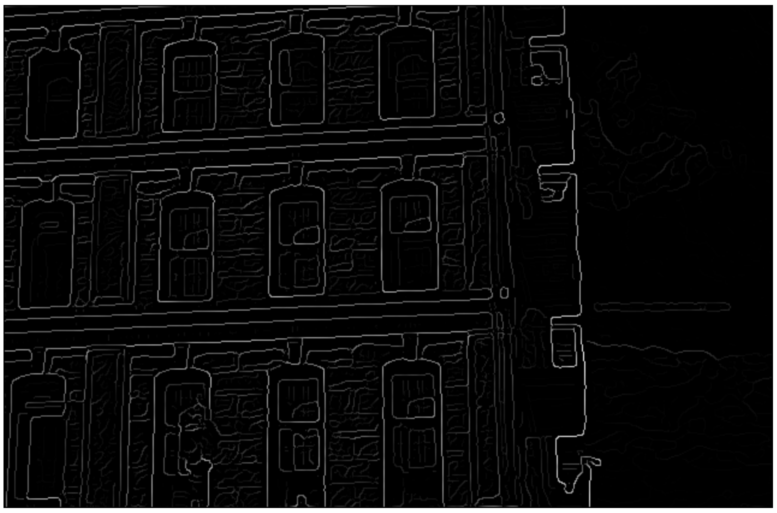


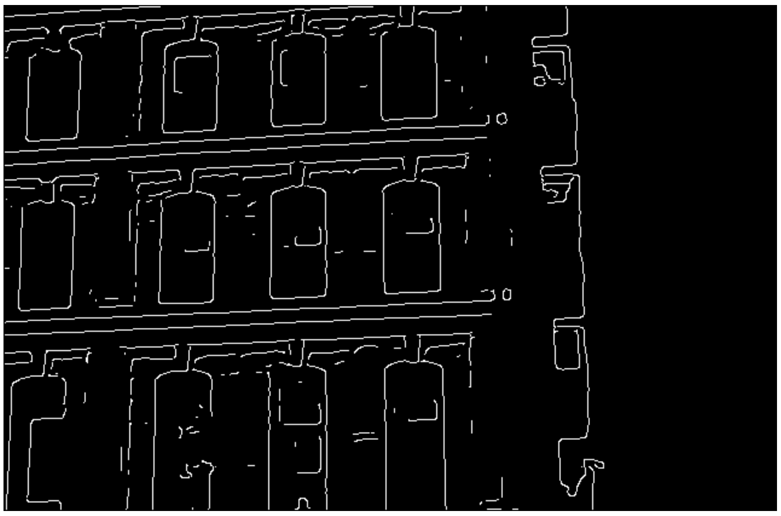
Image 8 --- Parameters:

Sigma = 2; threshold = 0.04;

rhoRes = 2; thetaRes = pi/180;

nLines = 80; 'FillGap' = 5; 'MinLength' = 10；





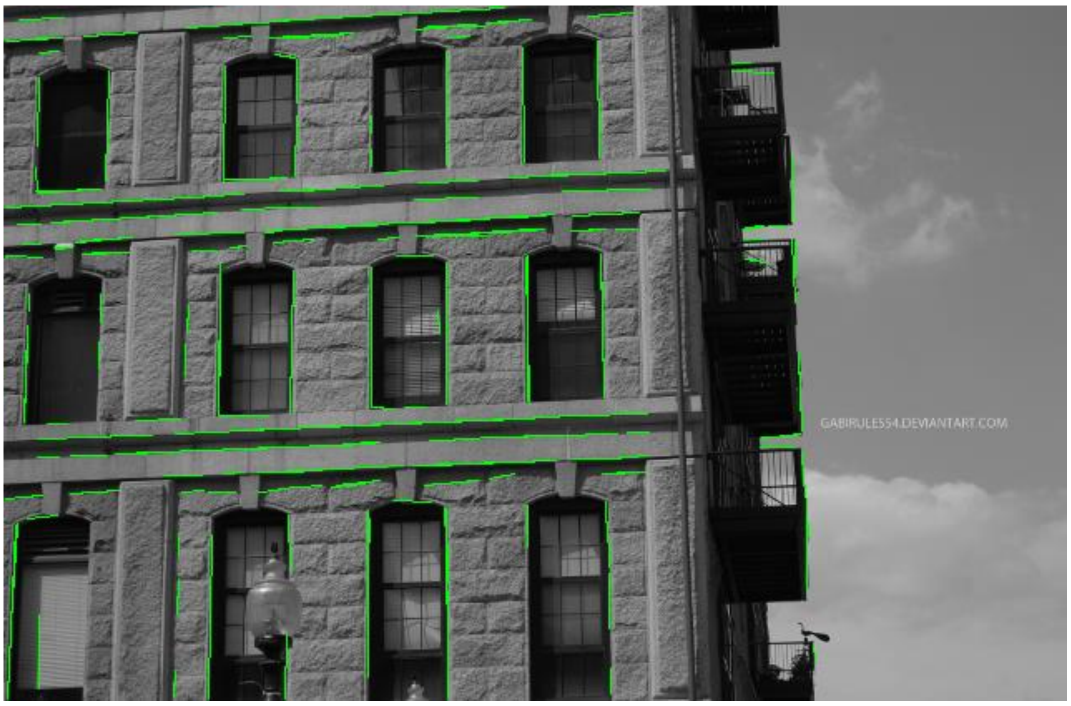
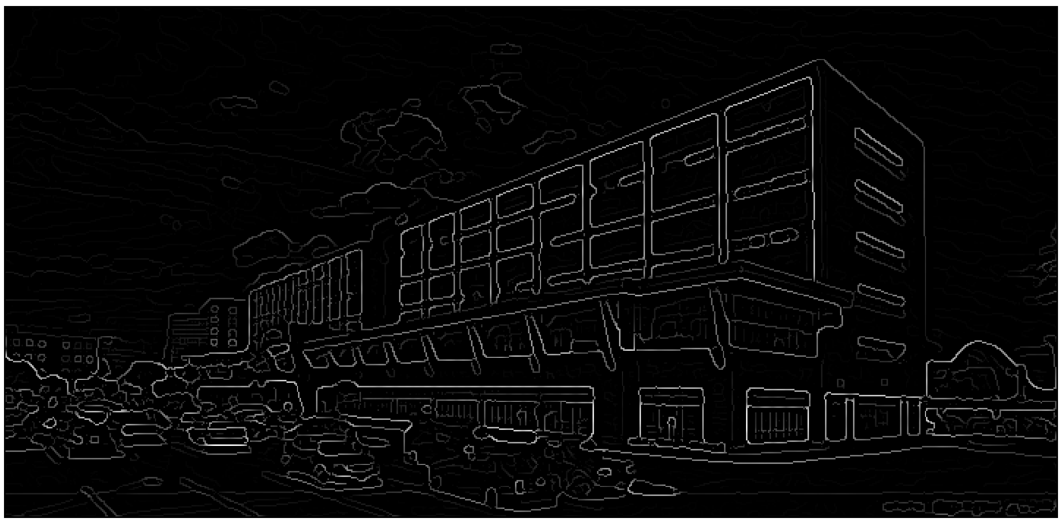


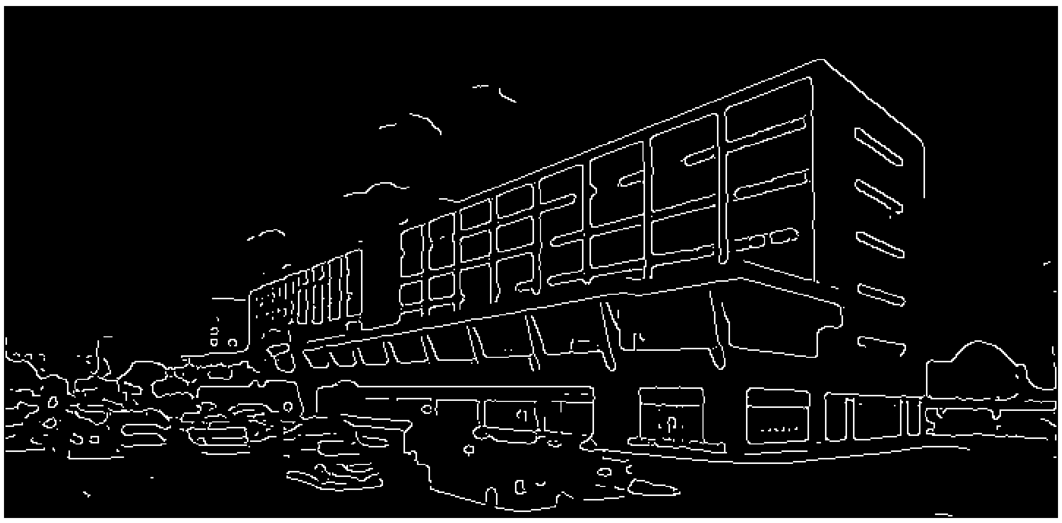
Image 9 --- Parameters:

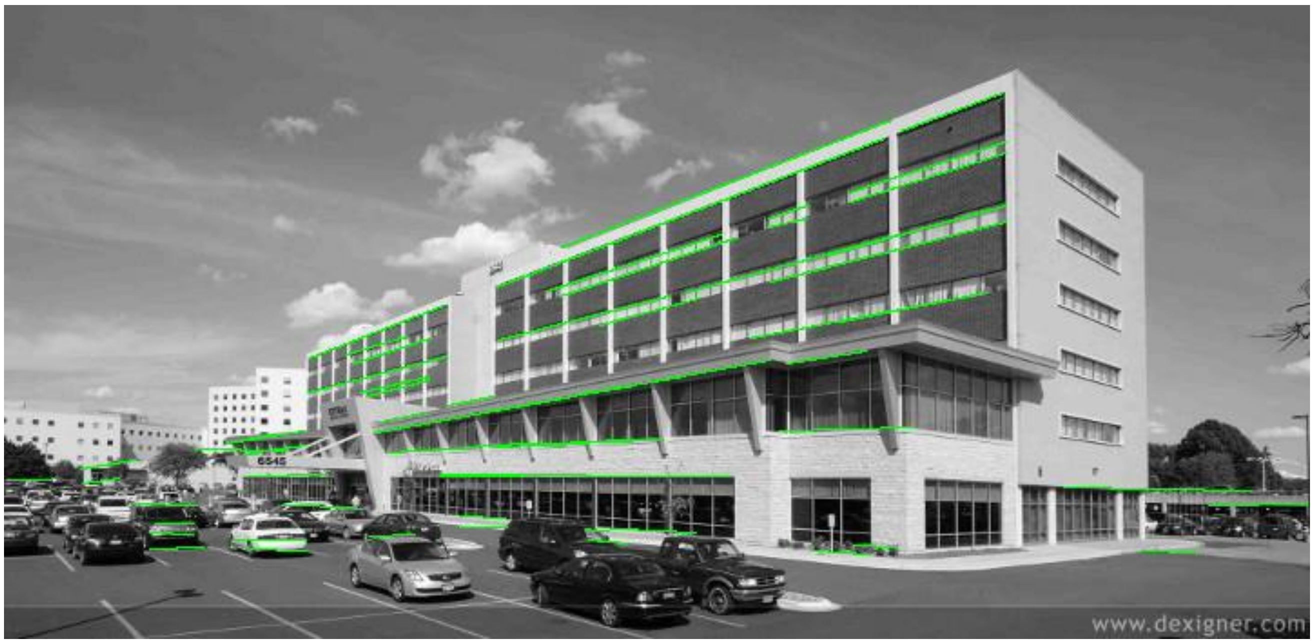
Sigma = 2; threshold = 0.04;

rhoRes = 2; thetaRes = pi/180;

nLines = 30; 'FillGap' = 8; 'MinLength' = 20;







Discussion

My code can work on all the images with a single pair of parameter but the outcome is not ideal. It can be seen in the image above that:

(3) Threshold: If the image's line are not clear in the edge picture, we should set the threshold to a relatively low value to enhance it. Similarly, we should set a higher threshold to decrease noise if the edge is already clear enough.

(4) thetaRes and rhoRes: Increasing these value gives a better resolution in the Hough voting picture, in my code rhoRes=pi/180 and thetaRes=2 works the best.

(5) nLines: If there's a lot of straight lines in the image to fit, then a larger nLine should be chosen. If there's a few lines but mostly curves, nLines should not be set too high.

(6) Fillgap: If there's discontinuous lines which should be joint together, then set fillgap to a relatively high value. On the other hand, if there's lines near each other which should not be connected, then set fillgap to a small value like what we did in image 6.

(7) Minlength: if the image only has long lines instead of couple of short lines, then minlength should be large, vise versa.

The houghscrip part is causing the most problems before change resolution. Since if the resolution is low, Hough peaks which should be near are treated as only one peak. So, some of the lines can not be found. After changing resolution, the problem got solved. However, for each element in the Hough matrix, there's still a problem that there might be a same value in its peripheral. We can only set the value to zero when there's a value lager than it in neighborhood. So it may result in a double line in final line fitting process. There's a way to solve it which is checking if there's any same element in the neighborhood then only reserve one of them, but it will need one more loop.

My best performance parameters are included in the writeup above, there's a different pair of parameters for different image. If applying only one pair of parameter to all the images, that will be:

Sigma = 2; threshold = 0.03;
rhoRes = 2; thetaRes = pi/180;
nLines = 50; 'FillGap' = 6; 'MinLength' = 15;
The effect of single parameter has already been stated above.