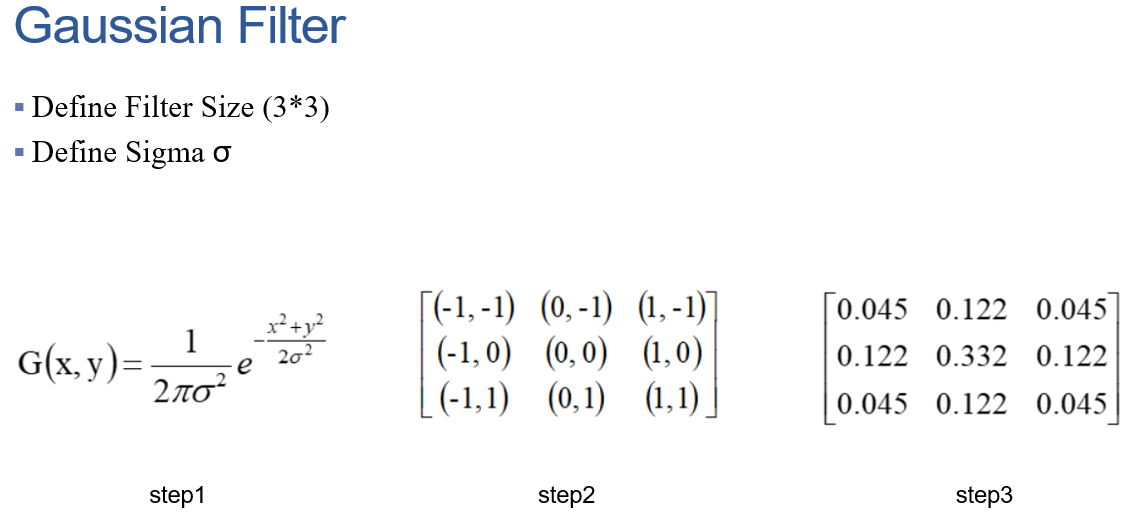
**Computer Vision**

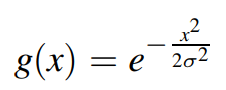
NTUT 資訊工程系111598070 賴俊霖

**Assignment 3**

**Q1. Gaussian Blur (result\_img1)**

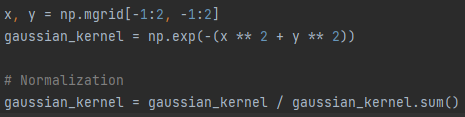
**Gaussian Filter：**

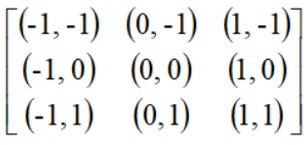
****

Although the equation above has 2\*pi\*σ,we can ignore the scale factor , and the equation will look like this 

to simplify our calculations.

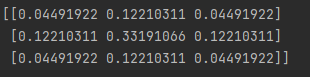
And it will be like below.

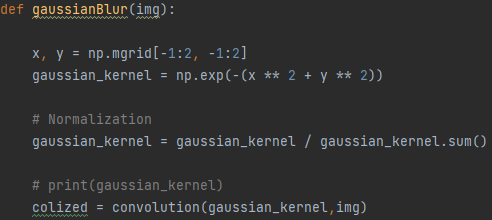


As you can see , np.mgrid[-1:2,-1:2] is used to gernerate the matrix like this .

And we use np.exp(-(x\*\*2+y\*\*2)) which is the formula we simplified.

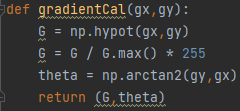
After using np.exp, we still need to normalize the matrix, and get the matrix below.

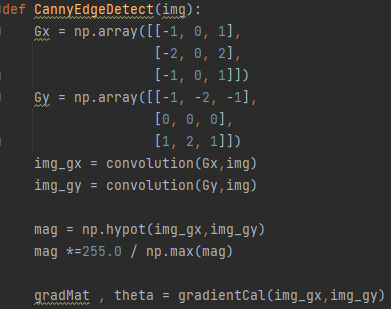




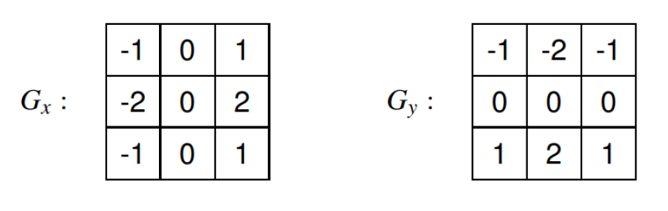
**Q2. Canny Edge Detection (result\_img2)**

**Gradient Calculation :**

****

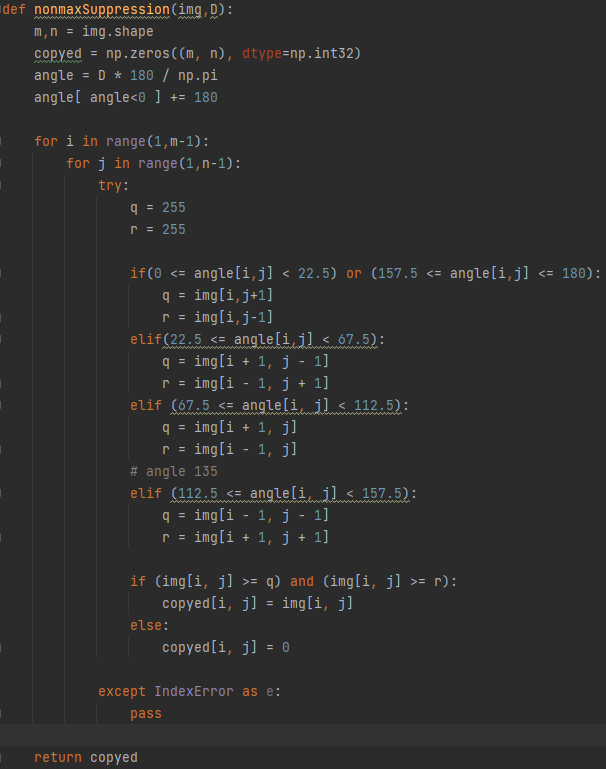
****

We can convolve the image with sobel kernel to get the horizontalImg and verticalImg. And they can help us to get magnitude and directions for each pixel in the image.



**Non-Maximum Suppression**

There are some problems about the output of Gradient Calculation , some edges are thick , but we wanna thin edges , so we gonna use nonmaximum suppression to fix them.



We create a matrix initialized to zero of the same size of the gradient magnitude matrix.

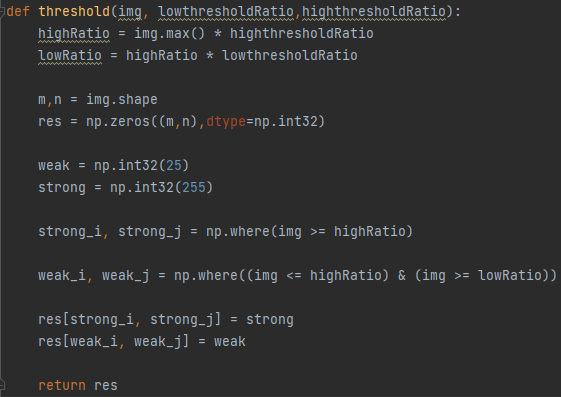
And identify the edge direction based on the angle value from the angle matrix. And we check if the pixel in the same direction has a higher magnitude than other pixels that is currently processed.

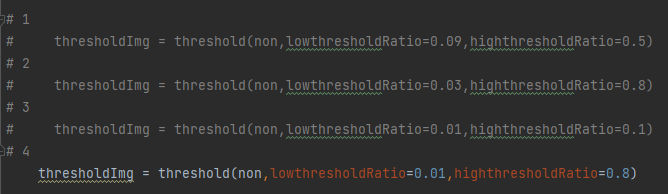
After processing, we get a same image but with thinner edges.

**Double Threshold**

Strong pixels are those with an intensity so high that we are sure they should be the final edge in the image.

And weak pixels may not enough to be considered as strong ones, but not small enough to be considered as null.

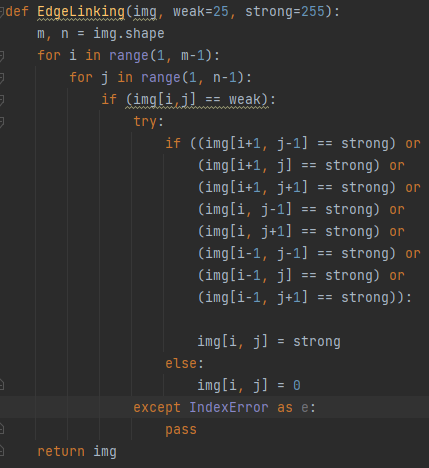




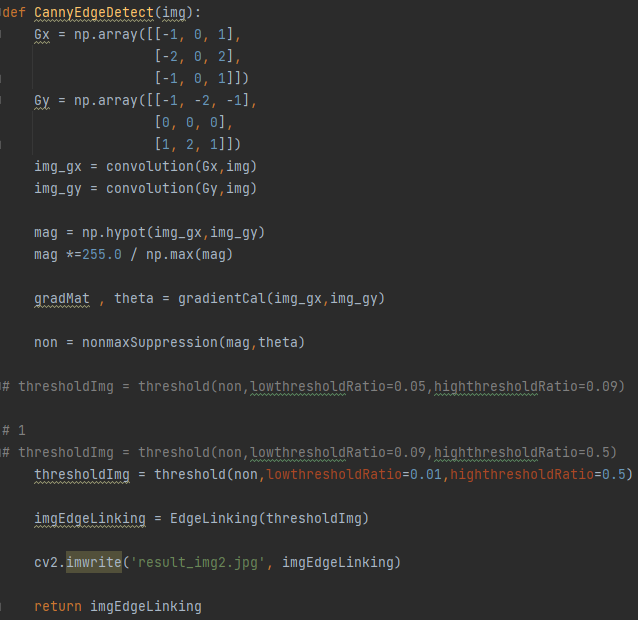
Pic above are the low / high threshold for every pics.

**Edge Linking**

After thresholding, the Edge Linking consists of transforming weak pixels into strong ones, if and only if at least one of the pixels around the one being processed is a strong one

****

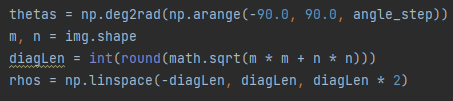
**Canny**

****

**Q3. Hough Transform(result\_img3)**

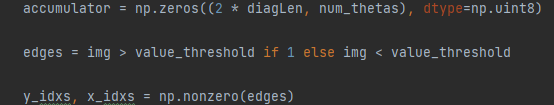
First we need to convert the traditional Cartesian coordinate system to polar coordinates, so that the computer will not encounter infinite size when computing.

Then we have to define the scope of Rho and Theta.

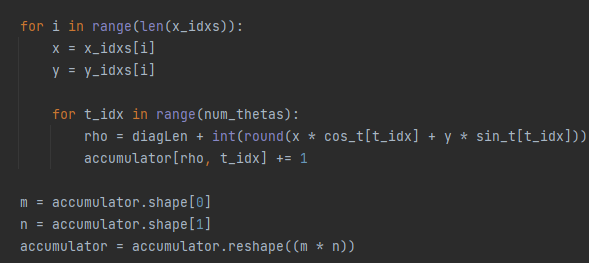


And we create a 2D matrix initialized to zero and the rows and columns are equal to the number of ros values and the number of thetas

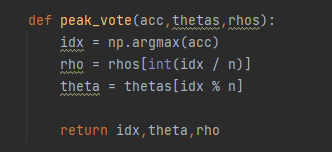
The values higher than threshold should be indexes to edges.

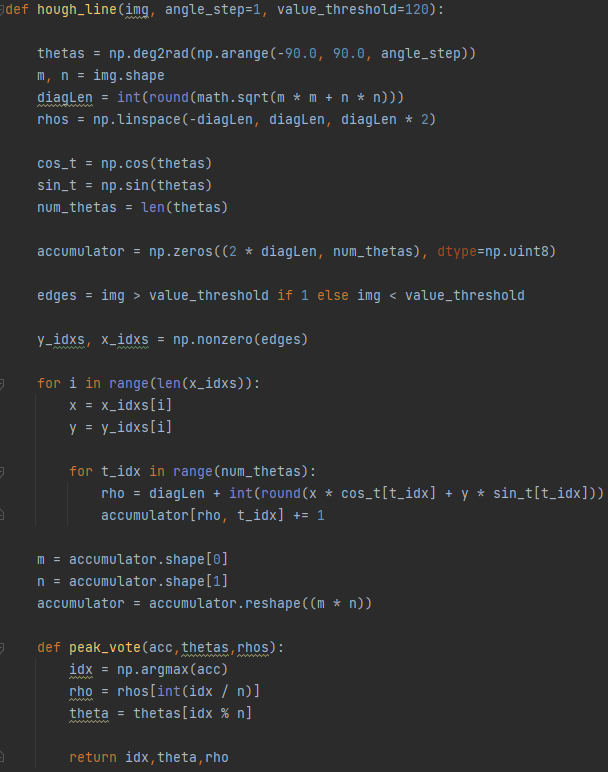


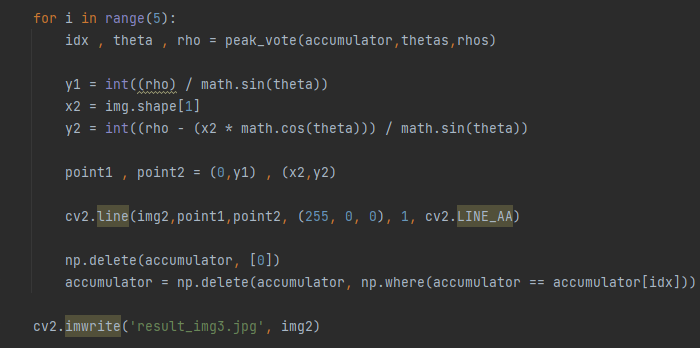
Voting in the accumulator, for each edge point and for each theta value, find the nearest rho value and increment that index in the accumulator.



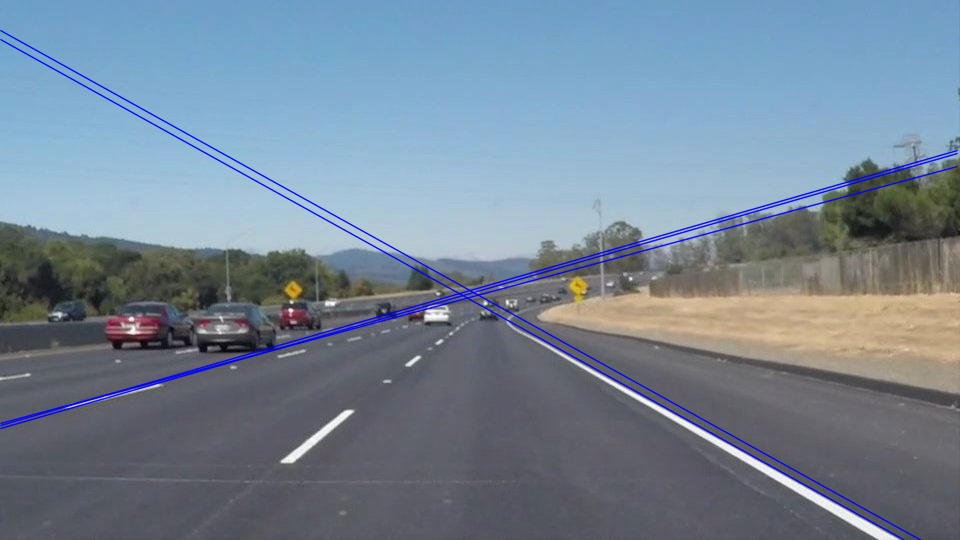
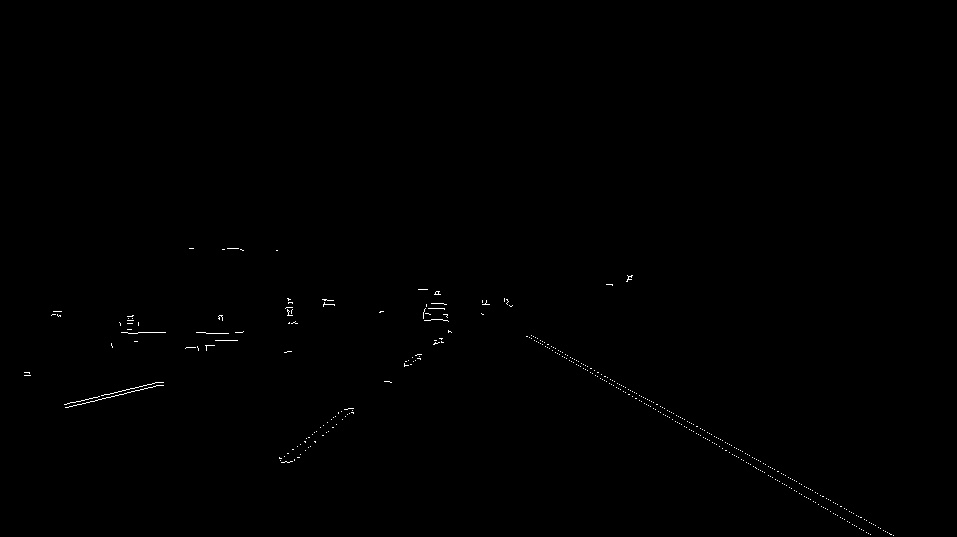
Local maximum in the accumulator indicates the parameters of the most prominent lines in the images we input. And the maximum maybe the possible lines for every index in accumulator matrix.



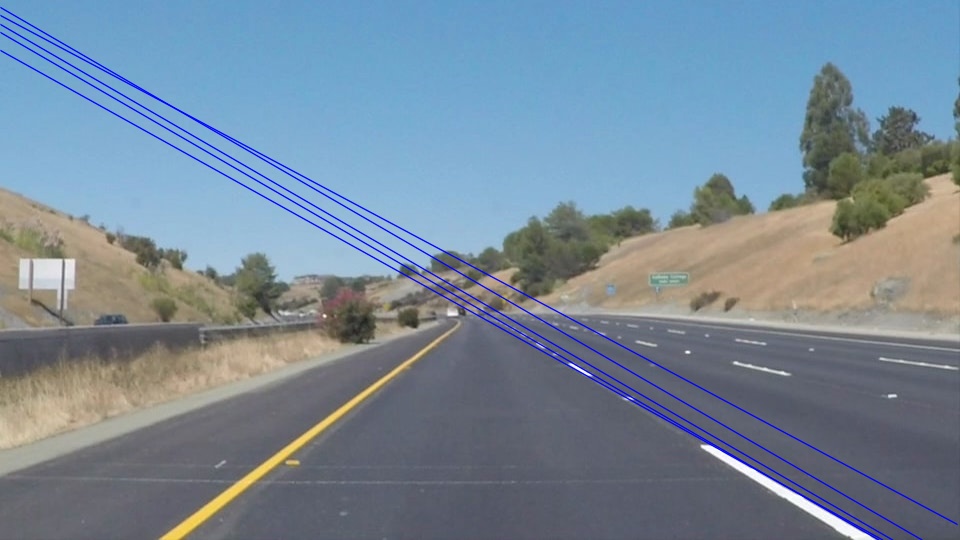
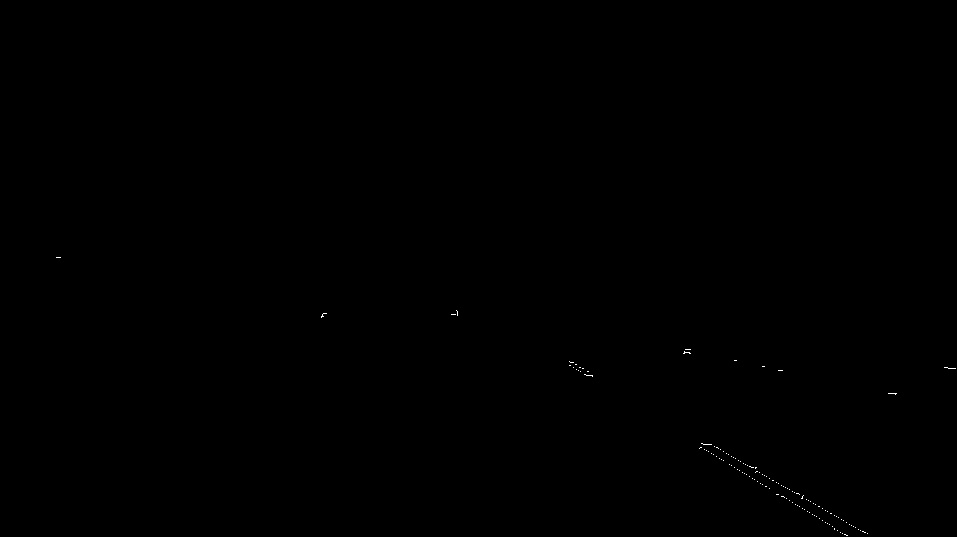
****



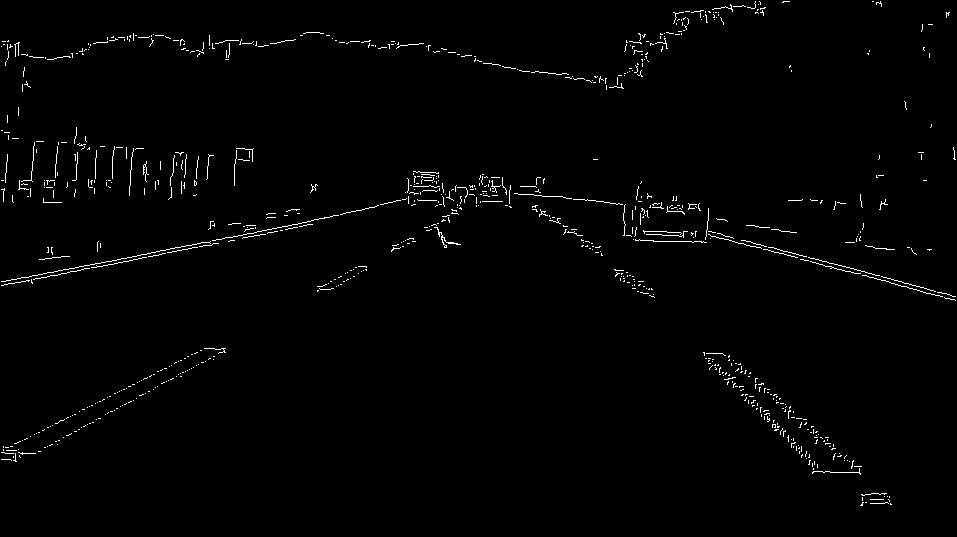
**Output of Pic 1**

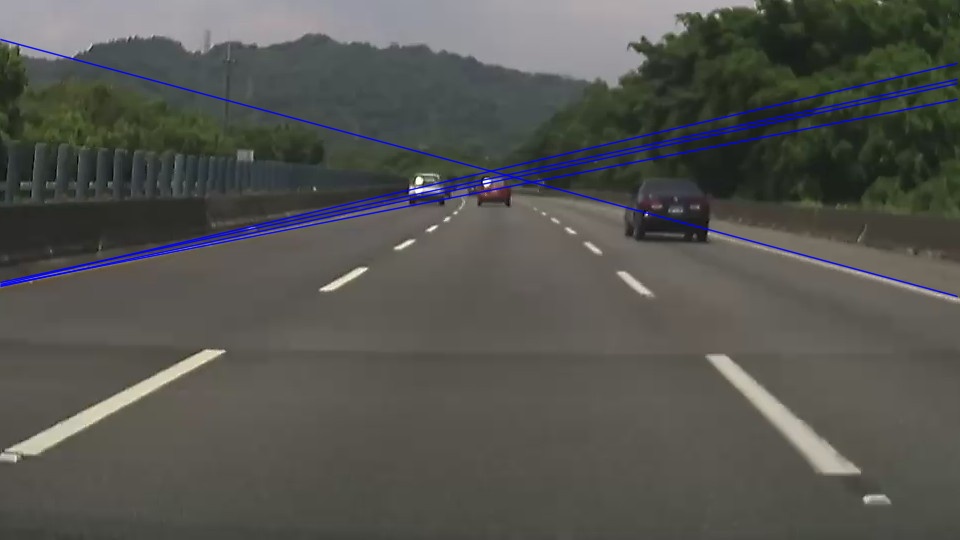
****

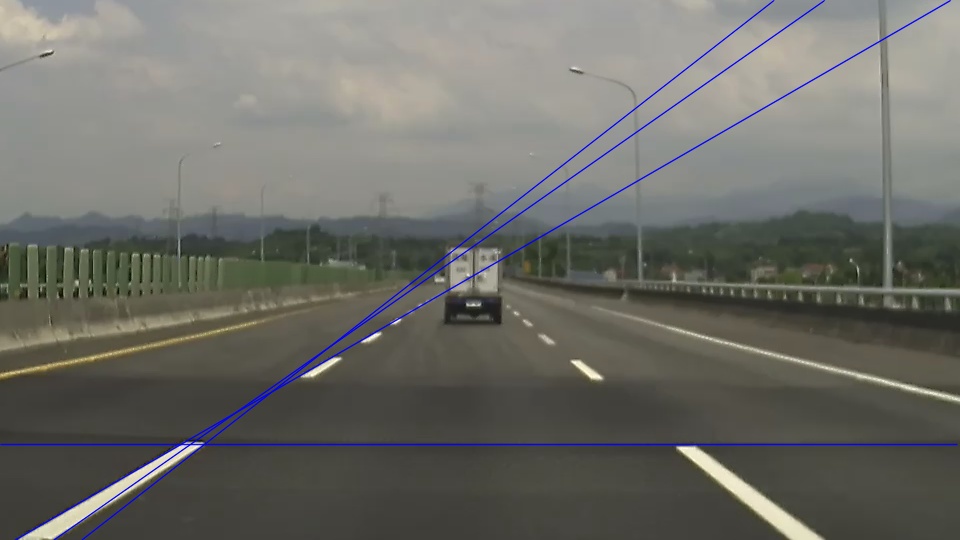
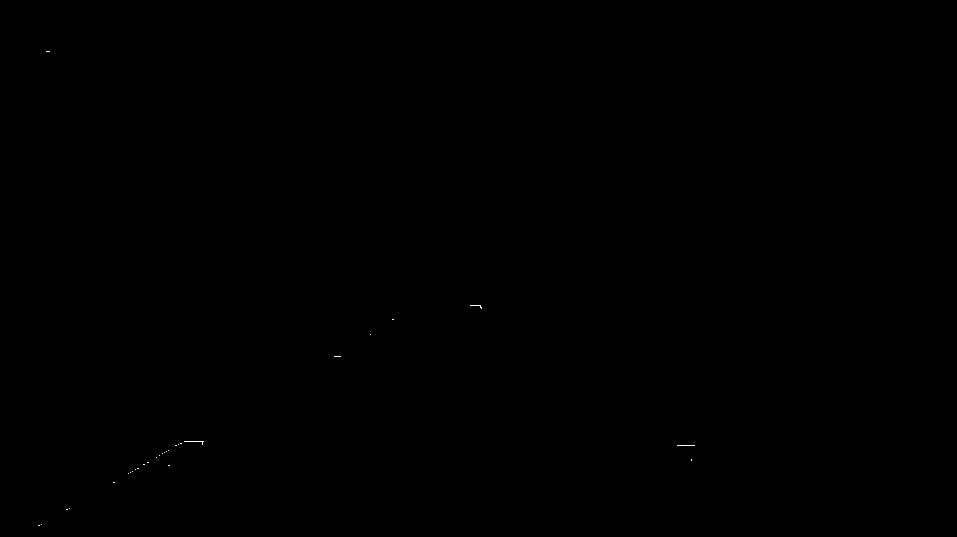
**Output of Pic 2**

****

**Output of Pic 3**

****

****

**Output of Pic 4**