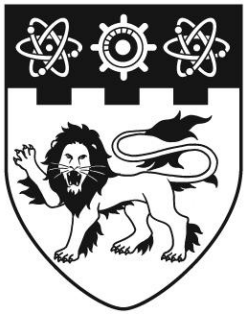


NANYANG TECHNOLOGICAL UNIVERSITY

EE4208 Intelligent System Design

Assignment 2: Canny edge detection

Submitted by:

Ng Yi Rong (U1820156H)

Wang Jue (U1820975L)

Zheng Yicheng (U1822468G)

School of Electrical & Electronics Engineering

Academic Year 2020/2021

Table of Contents

Table of Figures

Chapter 1 Introduction

The object of this project is to develop an edge detector based on the Canny edge detection algorithm. An additional sub-pixel accuracy canny edge detector is also required, and the performance from both algorithms will be tested with a real-life image shot by handphone.

This report will cover the flowchart of the programme, the process of the Canny edge detection algorithm, the approach for the sub-pixel accuracy, and the result and discussion.

The resources used in this project are mainly software applications and libraries. Python programming language is used with support libraries such as NumPy for faster mathematical calculation and OpenCV libraries for displaying the image.

The chessboard image which is used in this project is shown in Figure 1-1below.

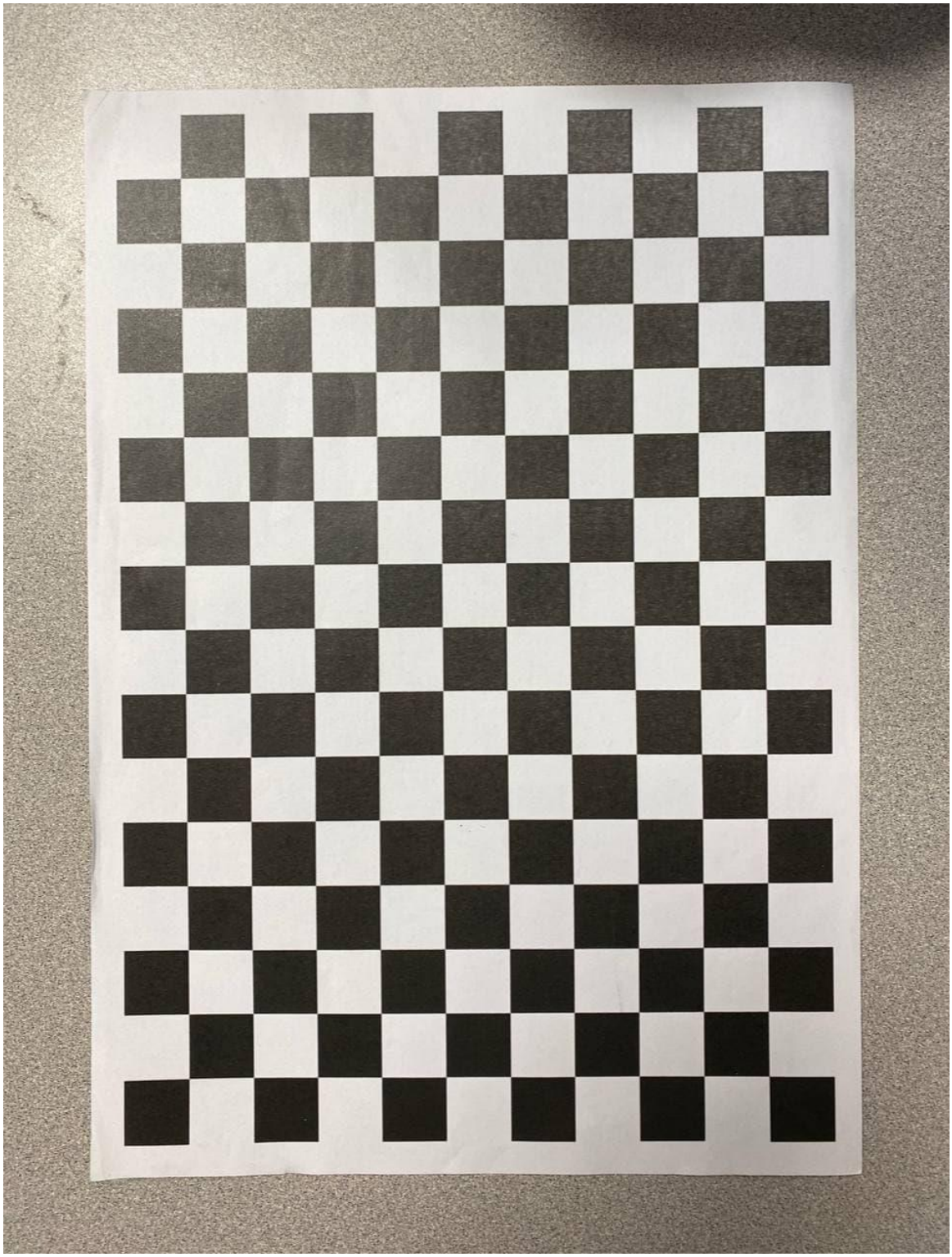


Figure 1-1: The Original chessboard image shot by handphone

Chapter 2 Flowchart

The flowchart for the Canny edge detector and the Canny edge detector with sub-pixel accuracy is shown in Figure 2-1 and Figure 2-2 below.

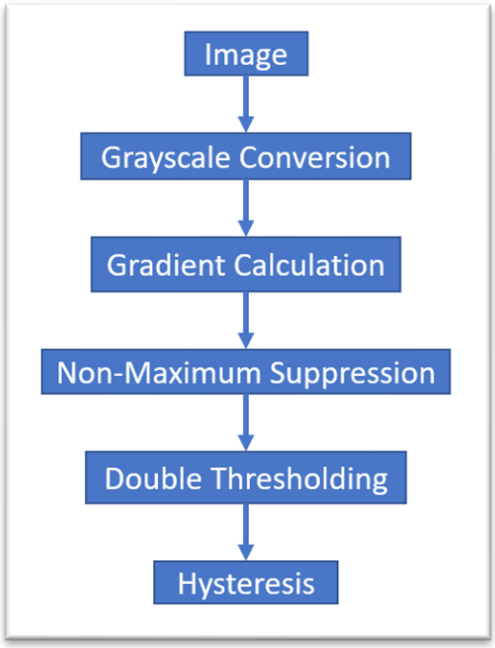


Figure 2-1: The flow chart of the Canny edge detector

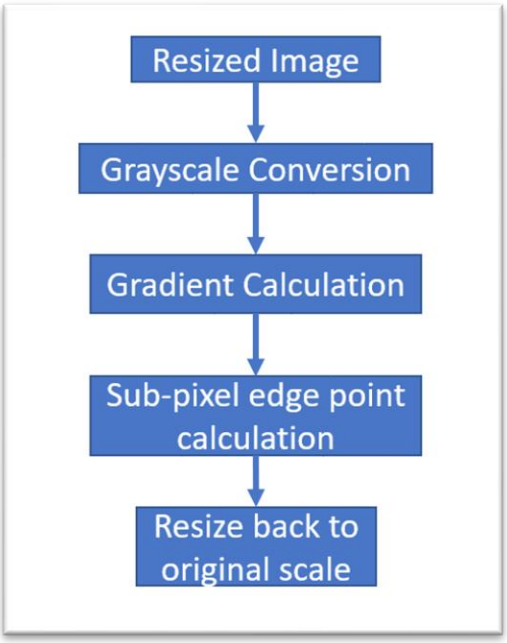


Figure 2-2: The flowchart of the Canny edge detector with sub-pixel accuracy

Chapter 3 Canny

The canny edge detector is an image processing method that used a multi-stage algorithm to detect edges in an image while suppressing noise.

Canny established three quality measures of a given edge detector which are good detection, good localization of the edge pixels and the unique response of the filter. [1]

The main process of Canny can be separated into 6 parts.

3.1 Grayscale Conversion

Since the purpose of this project is to find the edge of the original image, so instead of the RGB colour space, the original image needs to be converted into grayscale for better performance.

According to OpenCV, the conversion from RGB image to grayscale image requires three colour channels from RGB channel to calculate based on equation Y = Y = 0.299R + 0.587G + 0.114B.

The resulting image is as follows.

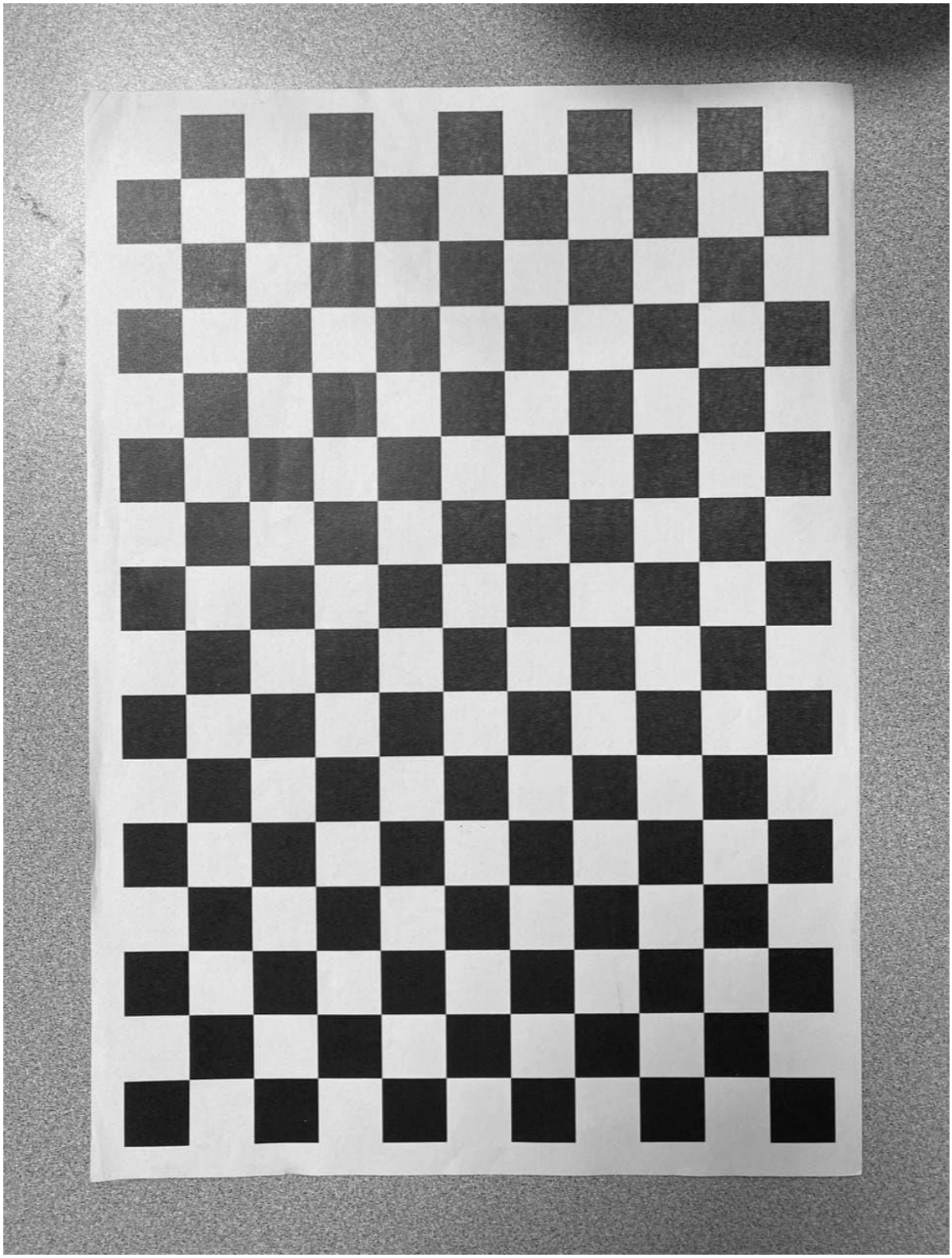


Figure 3-1: The grayscale image

3.2 Gaussian Blur

Since the gradient calculation in section 3.3 relies on the first derivative, it is highly sensitive to the image noise. In other to remove the unwanted spikes or noise in the original image, the Gaussian kernel is used in this step to filter out the unwanted noises.

The 2-D Gaussian convolution kernel is calculated based on the following formula.

$$f(x,y) = A\exp\left(-\left(\frac{(x-x_o)^2}{2\sigma_X^2} + \frac{(y-y_o)^2}{2\sigma_Y^2}\right)\right).$$

Here the coefficient A represents the amplitude, x_o and y_o represents the centre and σ_x and σ_y are the spreads of the blob.

For a 5*5 size, the Gaussian kernel is calculated as follows.

$$\frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

However, since the Gaussian kernel is known for its symmetry and its separability, the Gaussian convolution process can be done using a one-dimensional Gaussian kernel from the x and y direction, this will save the computational cost for a large image. Here the 2-D Gaussian kernel is used.

The resulting image is shown below.

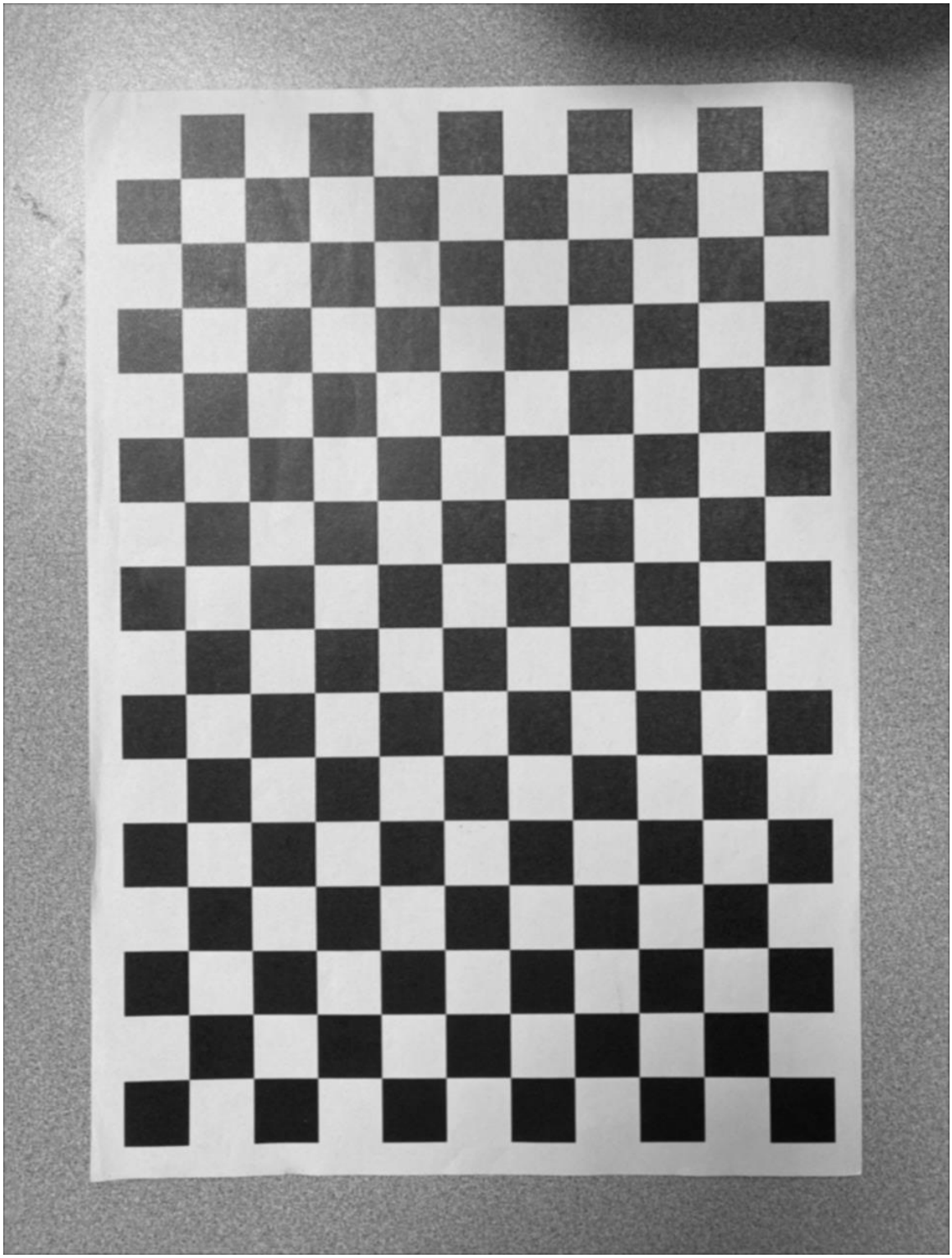


Figure 3-2: The Gaussian blur image

3.3 Gradient and Edge-orientation Calculation

Since edge can be defined as the sharp variation of the pixel intensity from one pixel to its neighbour pixel, so the first step is to calculate the first derivative of the grayscale image to determine the location of the edges.

Here the Sobel kernel is used.

The Sobel kernel can be classified into two categories, for the X direction and the Y direction. The kernel is shown as follows.

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

After applying them to the grayscale image, the two results from 2-D convolution gives the first derivative of the image from two directions. The edge magnitude can be calculated by $\sqrt{Ix^2 + Iy^2}$, and the normal of the edge can be calculated by $\arctan\left(\frac{Iy}{Ix}\right)$.

The following Figure shows the magnitude of the grayscale image.

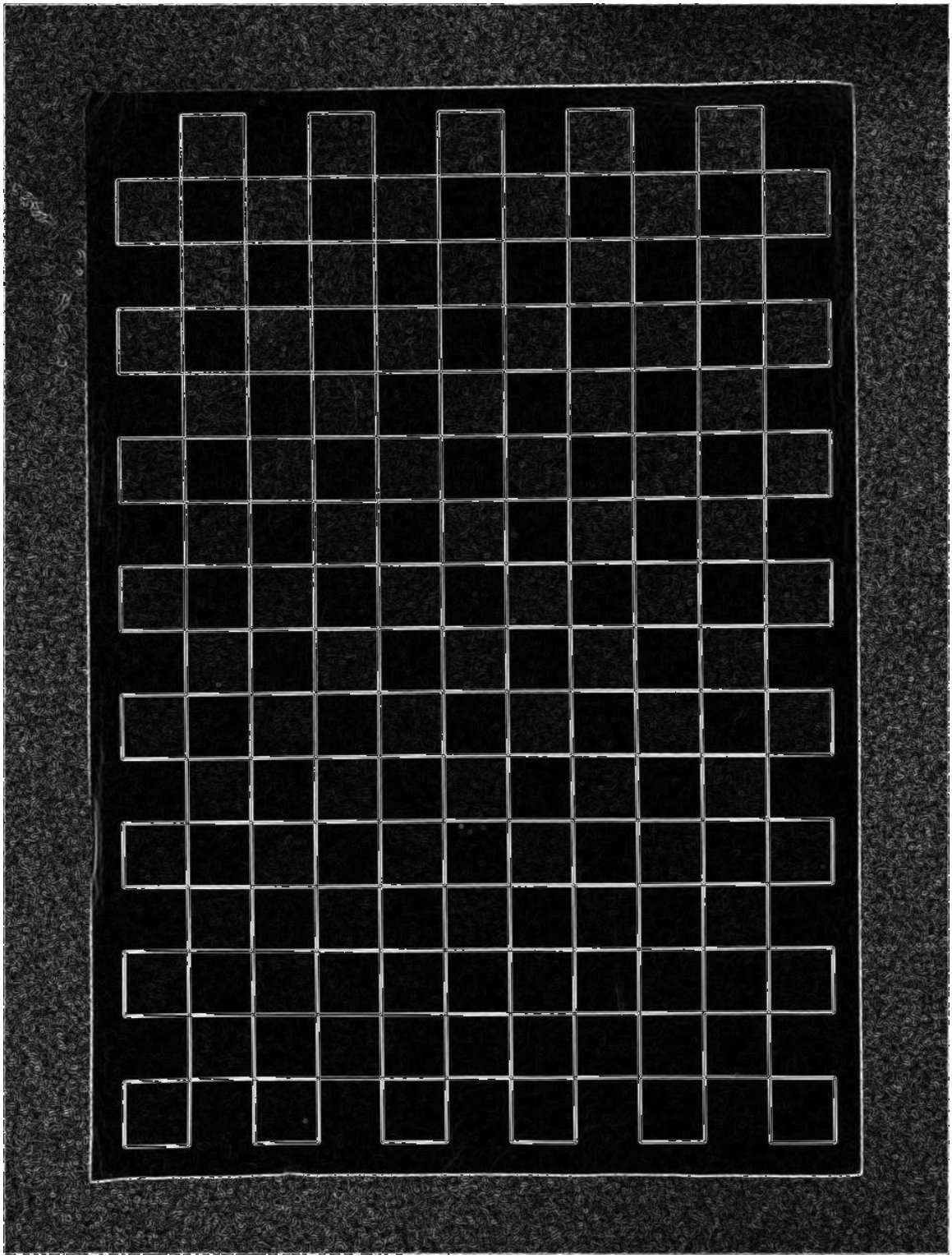


Figure 3-3: the Gradient Magnitude image

3.4 Non-Maximum Suppression

As the ideal edge should be thin, which means single-pixel width so that the non-maximum suppression is performed to thin out the edges.

The algorithm will go through all the points on the gradient magnitude image and finding the pixels with the maximum value in the gradient directions and usually, two neighbour points will be compared with the centre point to determine whether the centre point is the edge point.

Traditionally, the gradient direction will be round to eight different angles which are 0, 45, 90, 135, 180, 225, 270, 315 degrees, this is because the centre point is only surrounded by eight pixels so that practically only these eight points will be chosen to compare with the centre point.

However, since the gradient directions are usually not exactly equalling to the eight angles so that it has the potential to incorrectly determine a wrong pixel as the edge point. Here we assume that the changing rate between the two nearest points is linear and continuous. Based on this assumption, the solution used in this project is to use interpolation to calculate a sub-pixel level value and use it to determine whether the centre point is the edge point. An image illustration is as follows.

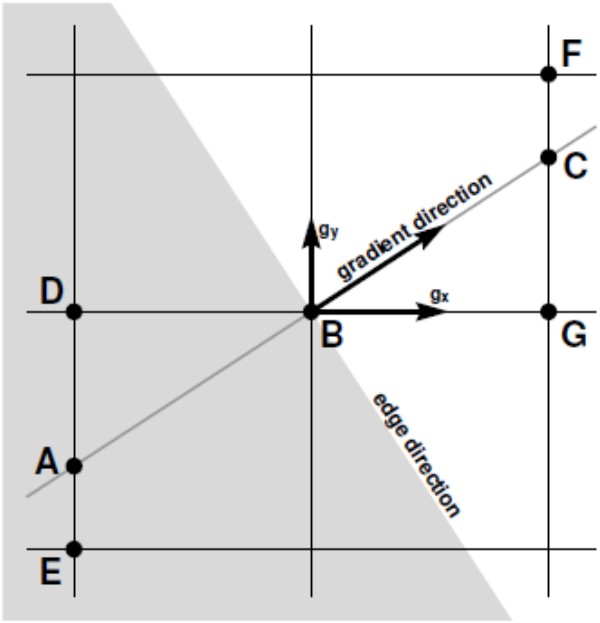


Figure 3-4: The image illustration of the interpolation NMS

The formula for calculating the interpolation value is as follows, the letters are with respect to the previous illustration.

$$||g(A)|| \approx \frac{g_x(B) - g_y(B)}{g_x(B)}||g(D)|| + \frac{g_y(B)}{g_x(B)}||g(E)||,$$
$$||g(C)|| \approx \frac{g_x(B) - g_y(B)}{g_x(B)}||g(G)|| + \frac{g_y(B)}{g_x(B)}||g(F)||.$$

The resulting image for this step is shown below. It can be seen that the left upper corner of the image has missing lines, this is because the environment light has the strongest intensity at the left upper corner in the original image. This problem will be fixed in the following procedures.

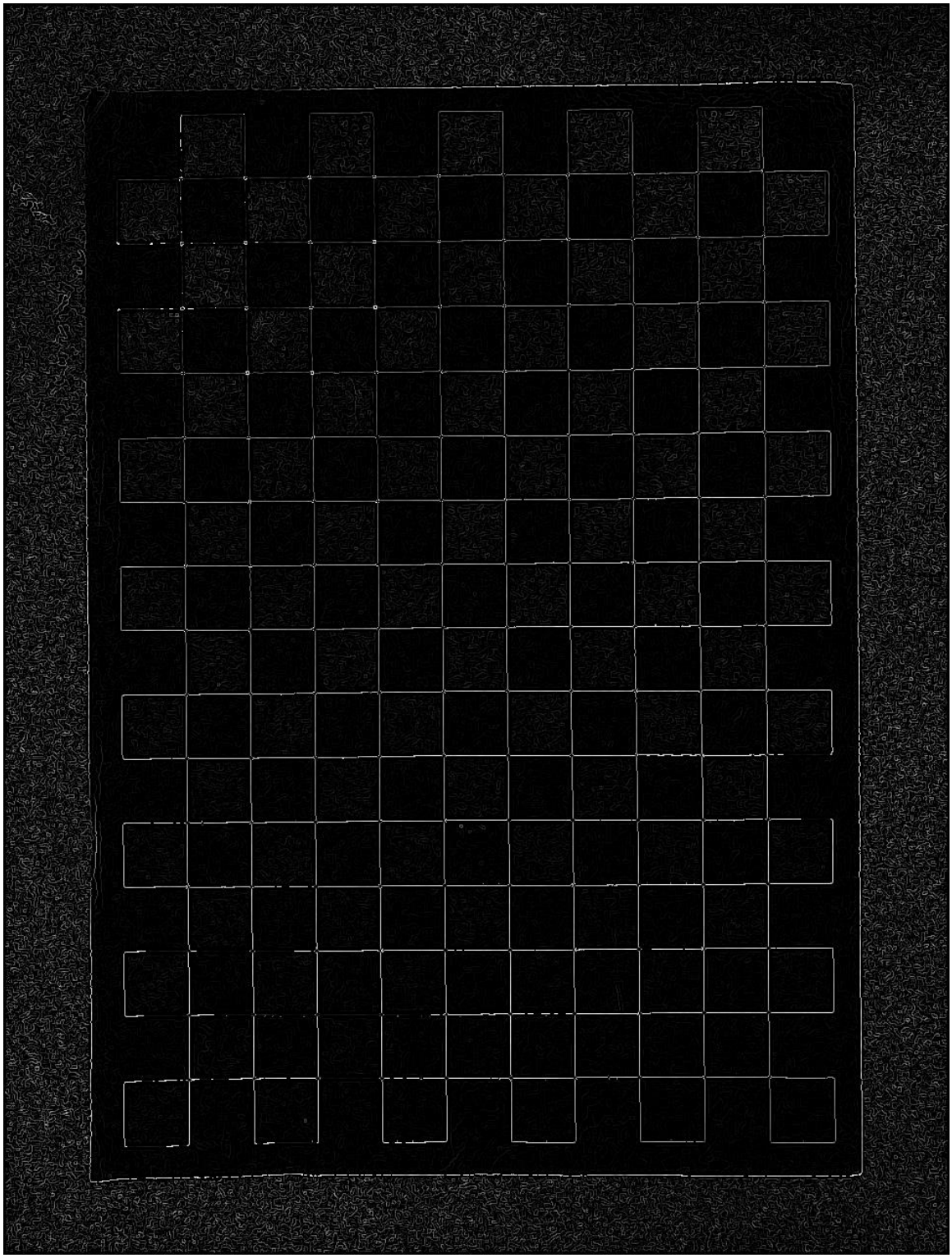


Figure 3-5: The image after NMS

3.5 Double Thresholding

As there is still variation with the intensity of the edges where some pixels seem to be brighter than others, double thresholding is used.

This step aims at classifying the pixels into 3 kinds: Strong, weak, and non-relevant.

Strong pixels are pixels that have an intensity that is high enough to contribute to the final edge, which is higher than the high threshold. Non-relevant pixels the pixels that do not have enough intensity which is lower than the low threshold, so that it can be considered as noise.

Weak pixels are pixels leftover pixels. They have an intensity value that is not enough to be considered as strong pixels yet big enough to be still considered for edge detection.

The low and the high thresholds are set by the ratio of 0.1 and 0.3 to the maximum intensity of the image.

After the edge points classification, the result is an image with 3 values, which are 0, 50, 255.

The resulting image is shown below.

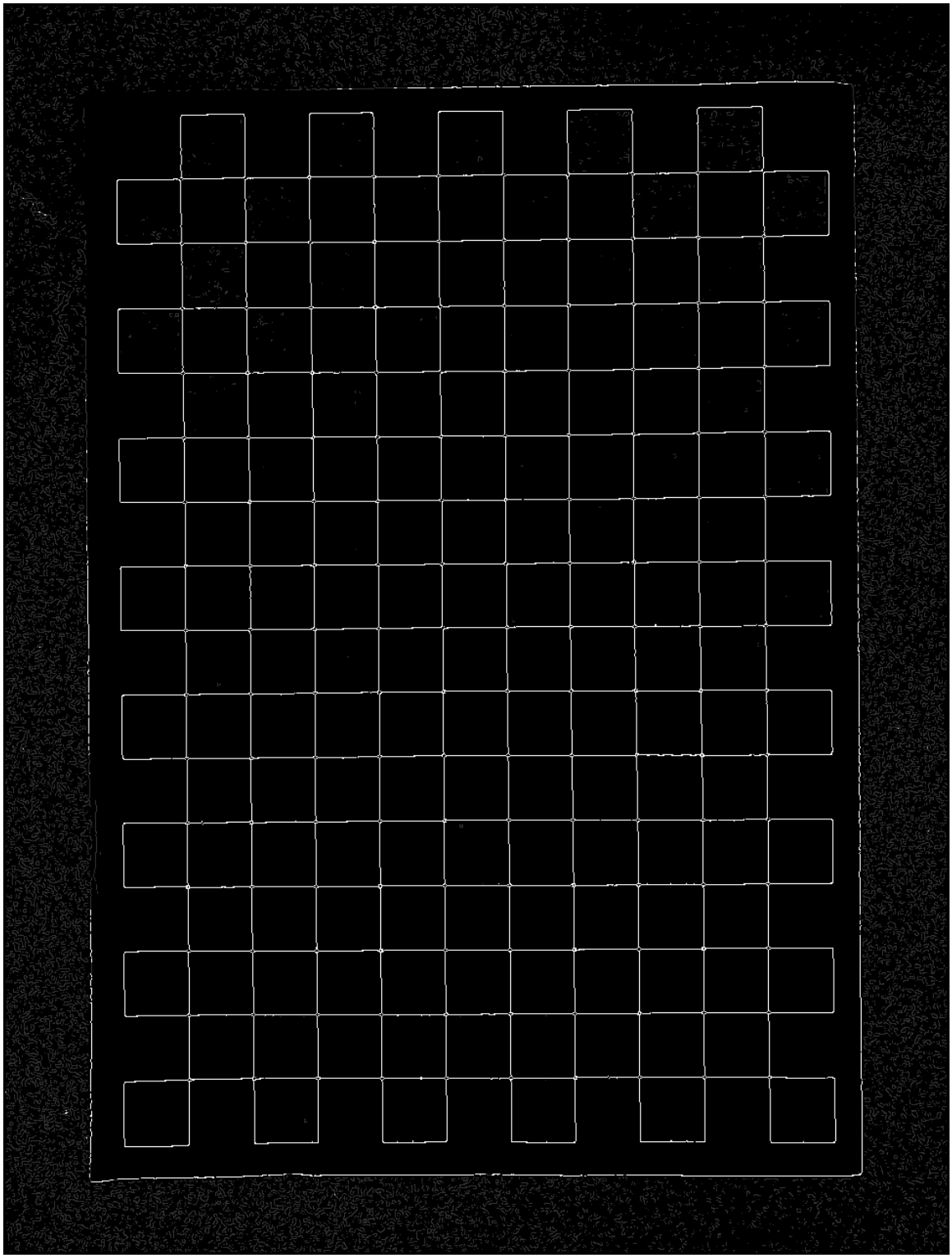


Figure 3-6: The image result after double thresholding

3.6 Edge tracking by Hysteresis

Based on the results obtained from the previous step, the only remaining undefined points are the weak edge points.

This hysteresis process will determine whether the weak pixels are edge or noise. It marks a weak edge point as a strong edge point if and only if this weak edge point is connected with a strong pixel. An image illustration of this step is shown below.

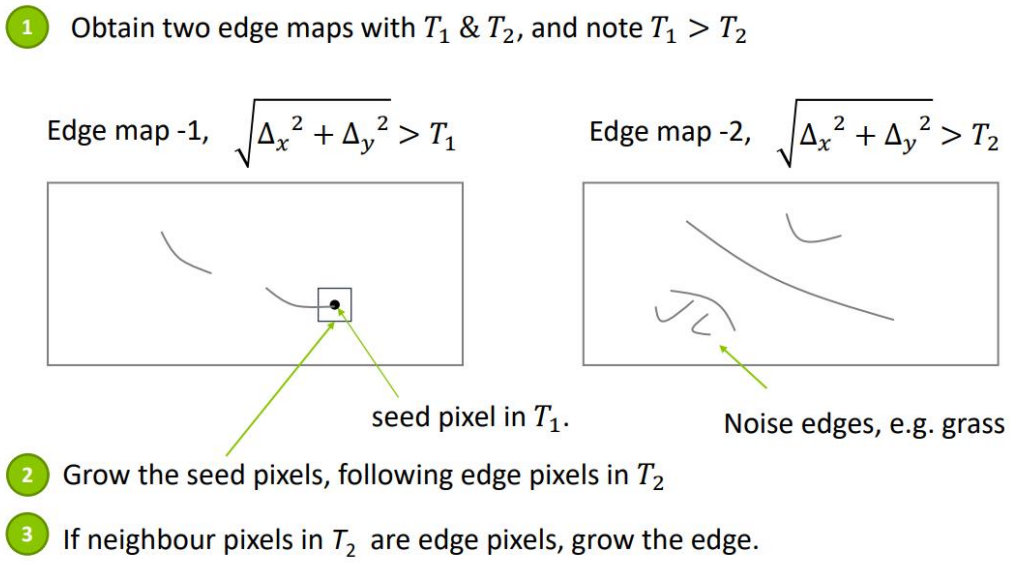


Figure 3-7: The illustration of the idea of Hysteresis

The result of this step is shown below.

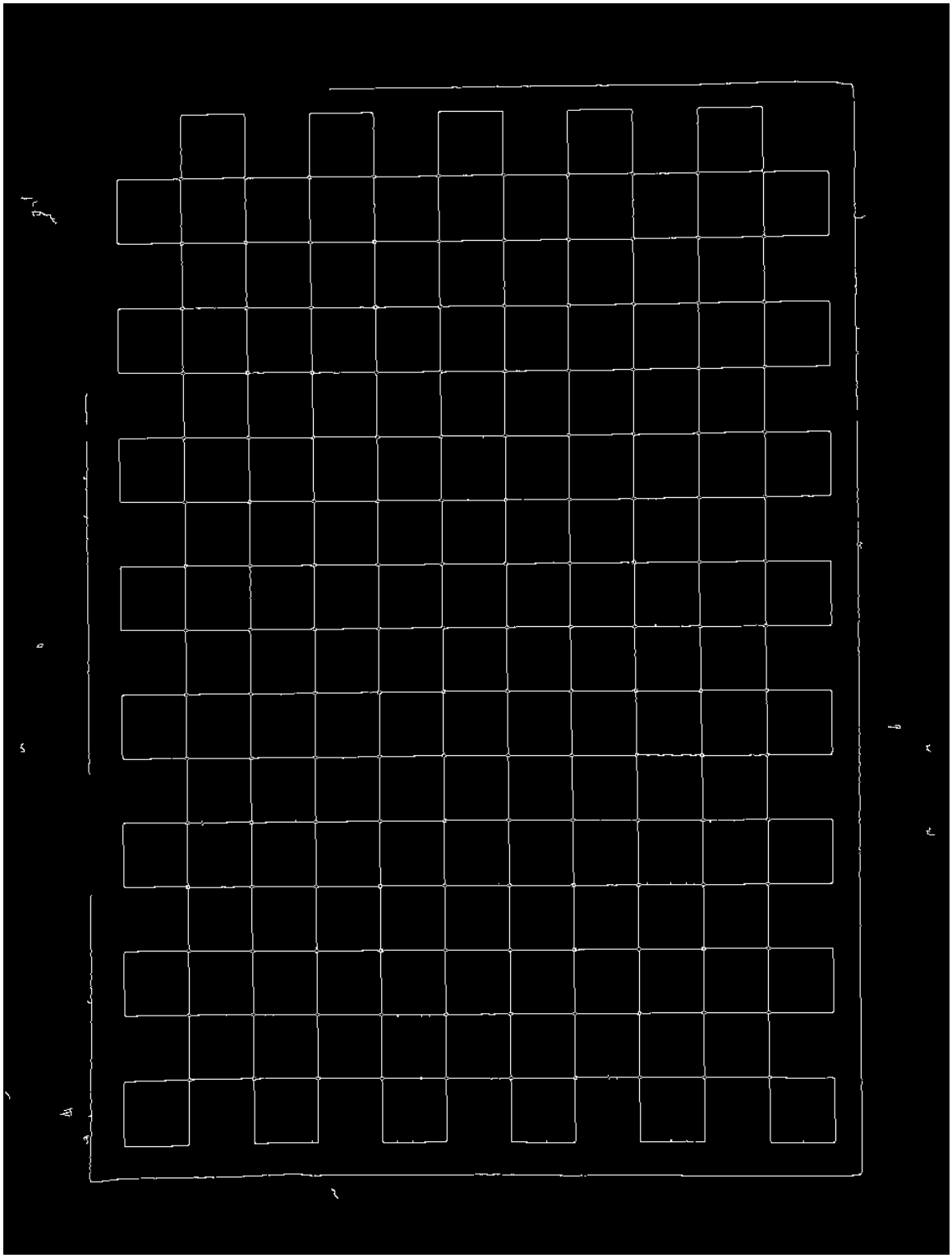


Figure 3-8: The result of the Canny edge detector

Chapter 4 Canny with sub-pixel accuracy

Canny's edge detection method will extract the edge points up to the pixel grid precision, however, many other applications, such as in a low-resolution image, requires a better location of the edge point. Therefore it is necessary to design an algorithm that provides sub-pixel accuracy of the edge points. This chapter will be focusing on the approach for sub-pixel accuracy edge detection.[2]

In this chapter, the original image mentioned in Chapter 1 is resized to its ¼, and the output of this algorithm is expected to be a list of coordinates. Grayscale Conversion, Gaussian Blur, and Gradian calculation in this chapter are as same as the previous chapter. The picture will resize back to its original scale based on the sub-pixel coordinates.

4.1 Sub-pixel accuracy

Assume an edge pixel, the gradient of this edge pixel and the surrounding pixels along the gradient direction are continuous instead of discrete so that the two pixels next to the edge and the edge pixel itself can be fitted into a parabola.

So that the position of the actual edge point can be defined as the maximum of the interpolation of the gradient norm, and the location can be calculated by solving the quadratic equation. An example of this method is shown below.

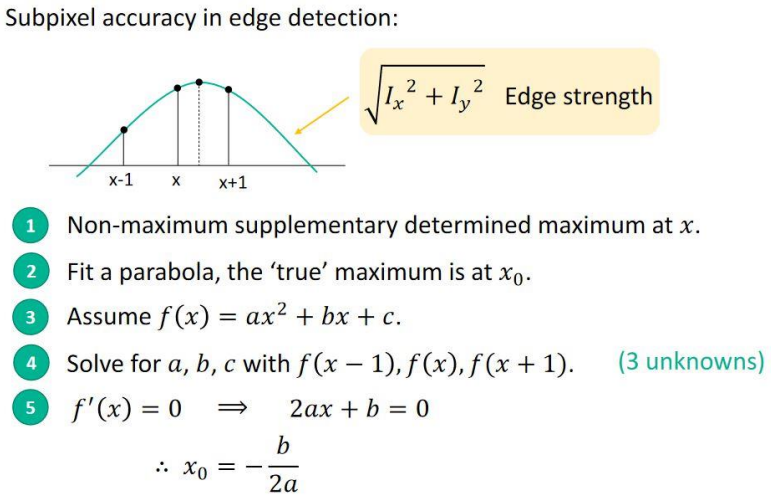


Figure 4-1: The illustration of the idea of parabola interpolation

Where X represents the known edge pixel, and x-1 and x+1 are the neighbouring pixels. The location X_0 can be calculated by the parameter of the quadratic equation. Or the location can also be calculated as follows:

$$\eta = \frac{1}{2}\frac{||g(A)|| - ||g(C)||}{||g(A)|| + ||g(C)|| - 2||g(B)||}$$

With this information, the actual location of the edge points can be calculated.

4.2 Resize to the original scale

Since the current image is shirked to its ¼ to create a low-resolution image, to resize it back to its original scale, the sub-pixel edge points are multiplied by 4, and the floor of the float value will be used as the new location for these edge pixels in the new image.

The gradient for the resized chessboard image and the screenshot of the output sub-pixel location are shown below.

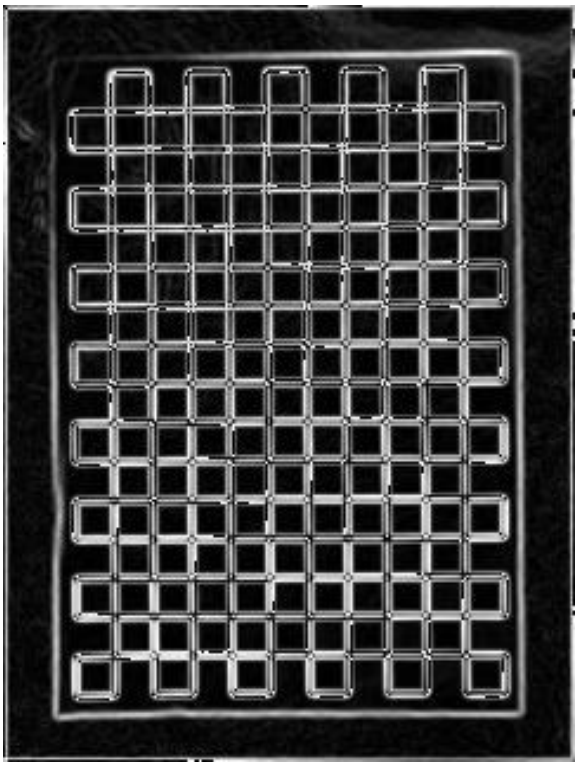


Figure 4-2: The gradient magnitude of the resized image

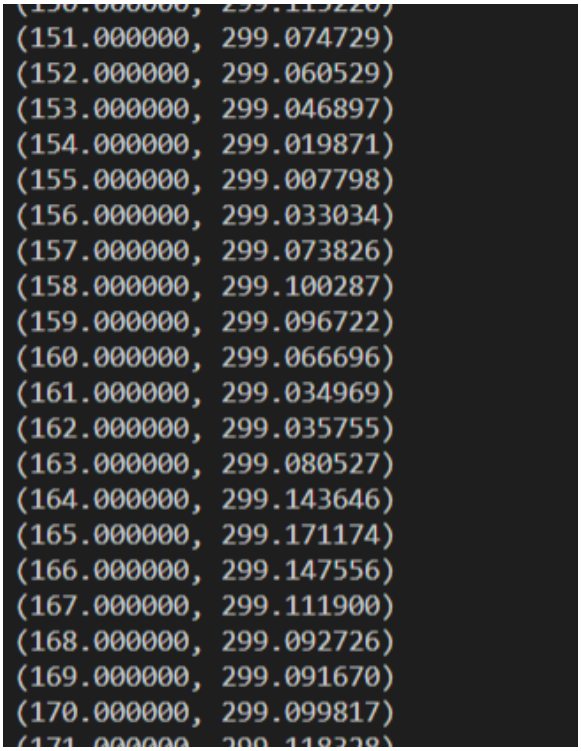


Figure 4-3: The screenshot of the output of sub-pixel location

The output of this image is as follows.

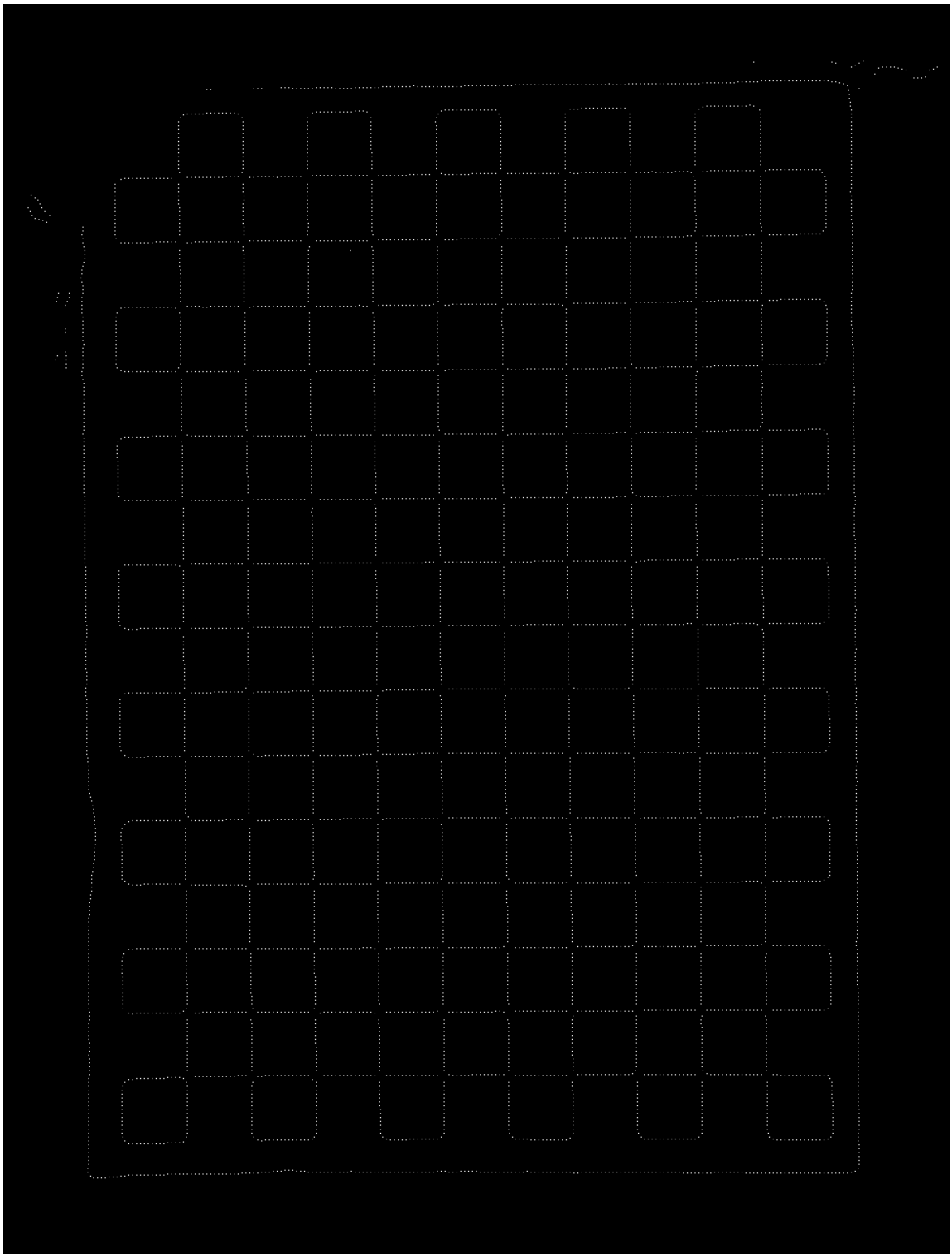


Figure 4-4: The resulting image with an original scale based on the sub-pixel location

As shown in the Figure above, it can be found that the sub-pixel is able to provide a similar-looking result to the result of the canny detector. However, the points are discrete, due to the reason that the interpolation processes are not performed for this project.

4.3 Evaluation

To test for the accuracy of our cany detector, the reference image is set as the resulting image by applying the canny detector on the original sized image. If the location of the edge points is found in the sub-pixel image, the value at the same location in the reference image will be checked. If there also exists an edge pixel, it will be marked as correct, otherwise, it will be marked as incorrect. The Accuracy can be calculated by $\frac{correct}{correct+incorrect} * 100\%$.

However, the accuracy for this point-to-point check gives a very low accuracy, so a 5*5 tolerance is given to it. This means instead search for the exact location on the reference image, a 5*5 range around the location will also be checked.

The accuracy after giving the tolerance for the chessboard image is 75.6%.

Chapter 5 Findings and performance analysis

The program is also applied to another real-life image. The details are shown below.

Figure 5-1: Original Lena image

Figure 5-2: Grayscale Lena image

Figure 5-3: Gaussian blurred Lena image

Figure 5-4: Gradient magnitude of Lena image

Figure 5-5: Lena image after NMS

Figure 5-6: Lena image after double thresholding

Figure 5-7: The Canny edge detector result

Figure 5-8: The Gaussian blurred for the resized image

Figure 5-9: The gradient magnitude of the resized image

Figure 5-10: The restored image based on sub-pixel location

The accuracy for this image is 56.2%.

Chapter 6 Canny edge detection's typical issue

6.1 Problem

Canny has a typical issue at the crossing of the chessboard pattern. Propose a solution to overcome it.

6.2 Observation

From the result of the Canny edge detection algorithm as well as the Canny edge detection with sub-pixel accuracy, we can observe that the edge pixel at the crossing of the chessboard is missing, which is shown below.

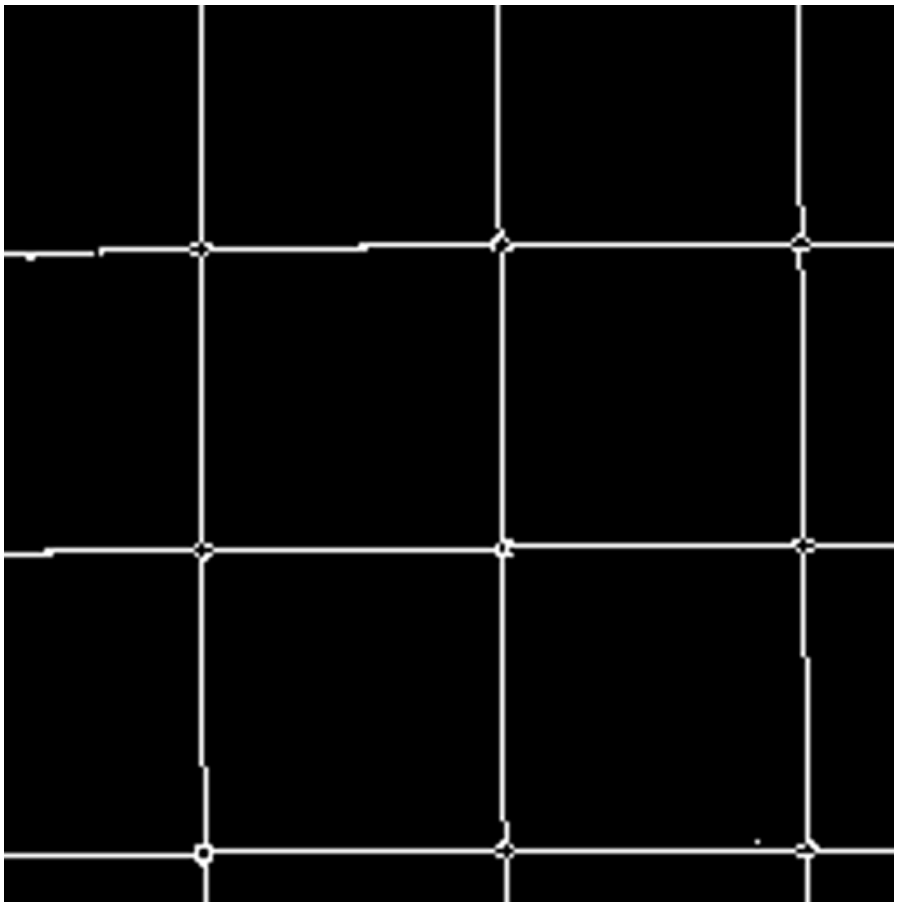


Figure 6-1: Screenshot from the Canny edge detector with chessboard crossing issue

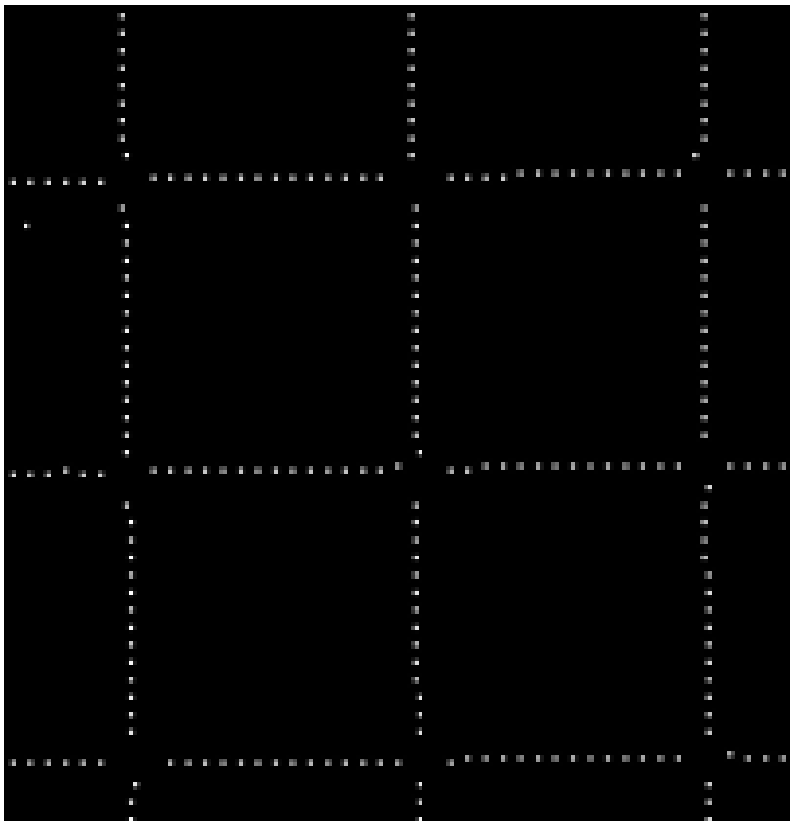


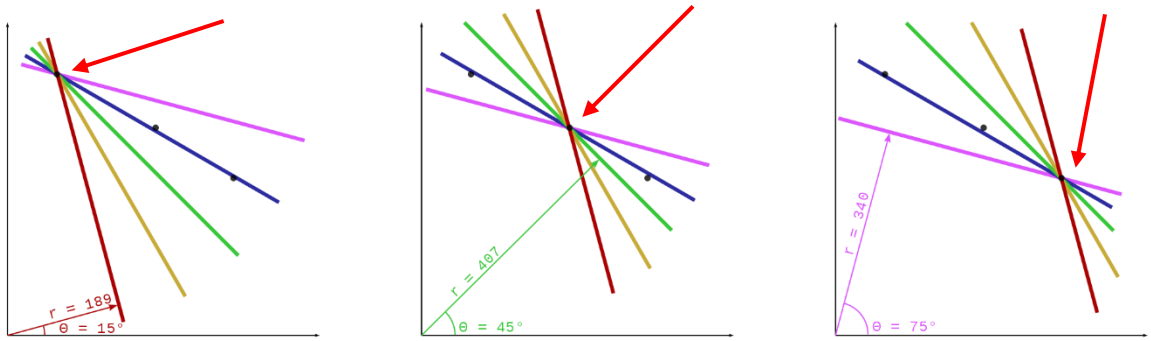
Figure 6-2: Screenshot from the sub-pixel accuracy Canny edge detector with chessboard crossing issue

6.3 Proposed solution

Based on the characteristic of the result, we believed that one of the solutions to this problem can be by using Hough Transform.

Hough transform is a feature extraction method which is to recreate the shape of the object by using imperfect edge information. The idea of Hough transform is that a change in representation converts a point grouping problem into a peak detection problem.

In this case, the edge pixels from the Canny edge detector can be treated as edgels in image space, so the set of lines that passes through the edge pixel (x, y) can be represented y = ax + b. so that if these pixels are all transformed into parameter space, it's easy to find an appropriate parameter pair (a, b) where the line in image space can pass through all edge pixels. This kind of parameter pair can be found at the intersection in parameter space. An example can be found in the Figure below.



θ	r
15	189.0
30	282.0
45	355.7
60	407.3
75	429.4

θ	r
15	318.5
30	376.8
45	407.3
60	409.8
75	385.3

θ	r
15	419.0
30	443.6
45	438.4
60	402.9
75	340.1

Figure 6-3: Image illustration of the parameter space in Hough Transform

With this information, it's easy to redraw and recreate the lines on the chessboard, thus the crossing problem is solved.

Reference

[1] CANNY, J. (1987). A computational approach to edge detection. *Readings in Computer Vision,* 184-203. doi:10.1016/b978-0-08-051581-6.50024-6

[2] Grompone von Gioi, R., & Randall, G. (2017). A sub-pixel edge detector: An implementation of the canny/devernay algorithm. *Image Processing On Line, 7,* 347-372. doi:10.5201/ipol.2017.216