國立臺灣科技大學
NATIONAL TAIWAN UNIVERSITY OF SCIENCE AND TECHNOLOGY

實習二
影像輪廓的擷取

課程大綱

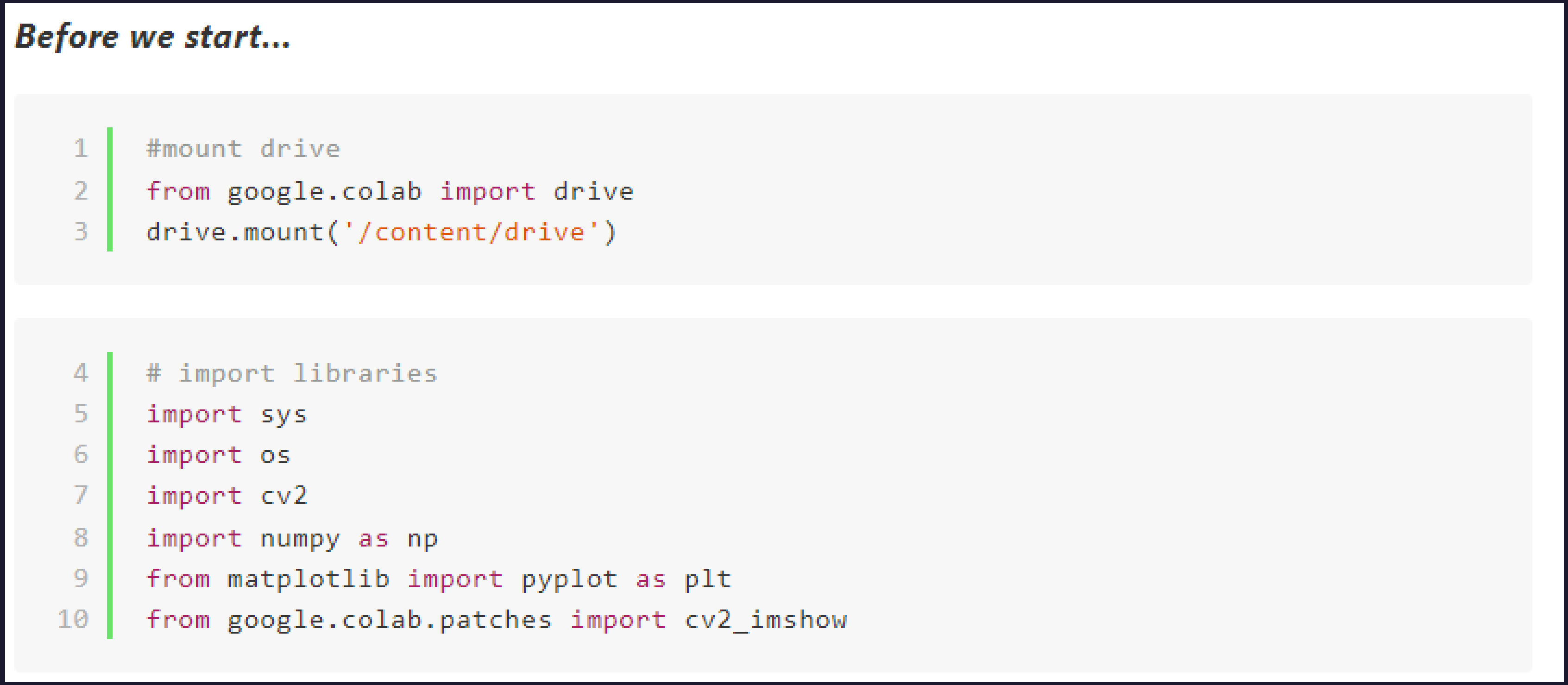
- 實習00: Colab 環境
- 實習02:影像輪廓的擷取

實習 00

Colab 環境

Colab Env.

Colab Env.



Before we start...

```python
#mount drive
from google.colab import drive
drive.mount('/content/drive')
```

```python
# import libraries
import sys
import os
import cv2
import numpy as np
from matplotlib import pyplot as plt
from google.colab.patches import cv2_imshow
```

實習 02

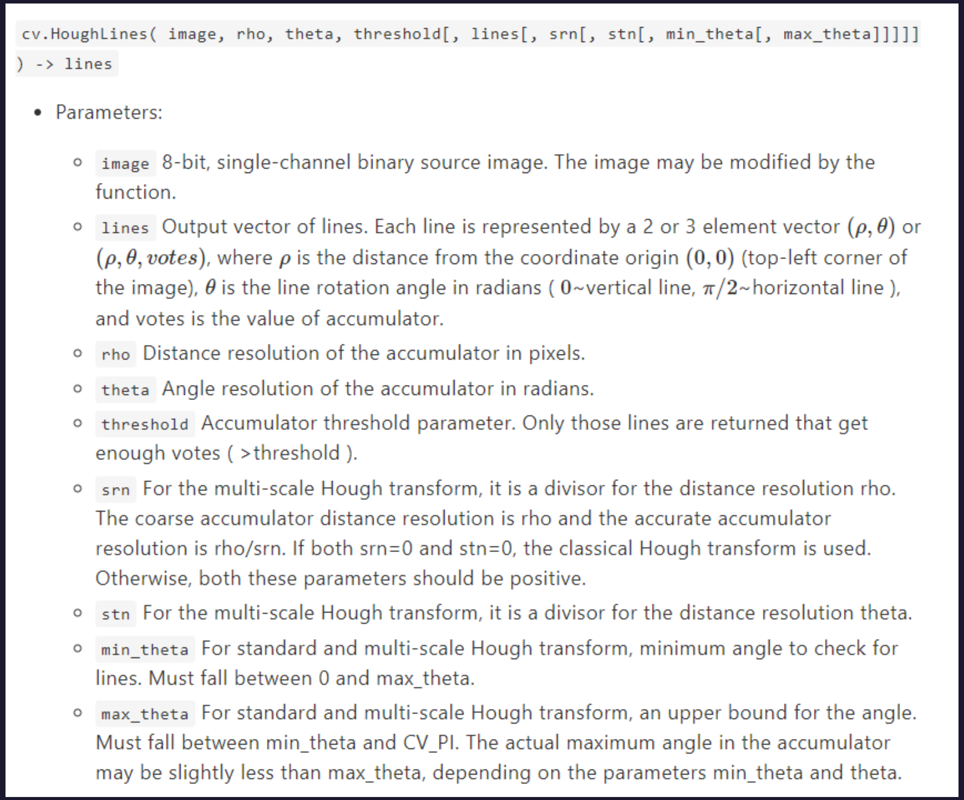
影像輪廓的擷取

Image Edge

TASK: 對影像作邊緣檢測。

實驗影像(sudoku.jpg)下載處:
- https://reurl.cc/bEryg3

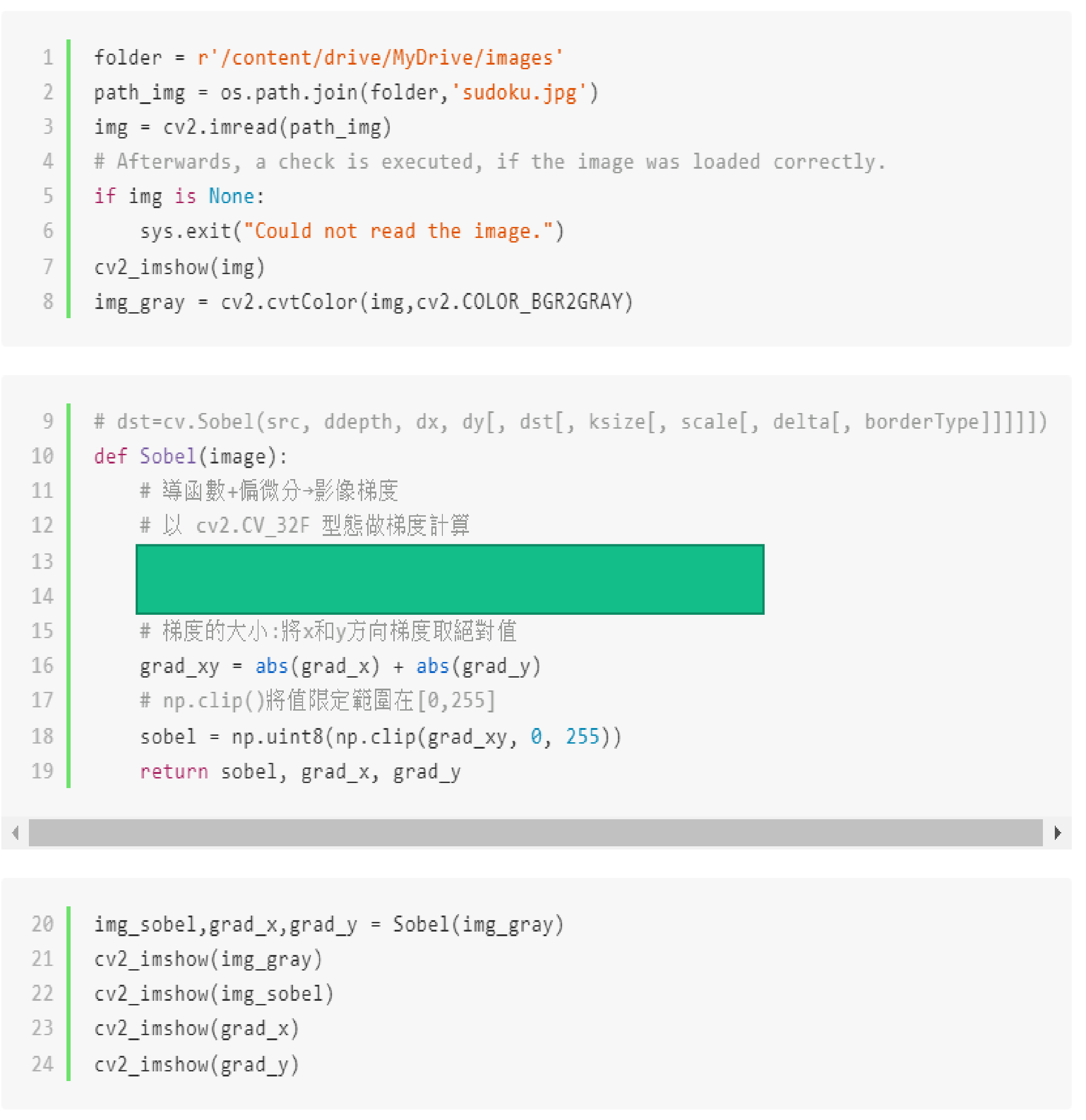
- **使用函式:**
  - **Sobel**
    - grad_x：x方向 sobel 梯度； grad_y：y方向 sobel 梯度
    - 使用 OpenCV 的 cv2.Sobel()
    - 需要使用 cv2.CV_32F 型態做梯度運算
    - Kernel size：3
  - **Laplacian**
    - OpenCV 的 cv2.Laplacian() 參數:
      - 使用 cv2.CV_8U 型態做梯度運算
      - Kernel size：3
  - **Hough: Line**
    1. 先使用 cv2.Canny() 做影像輪廓擷取(edges)
    2. 再使用 cv2.HoughLines() 做直線檢測(lines)
    3. OpenCV 的 cv2.HoughLines() 參數：
       - image：edges
       - rho：1 <累加器的距離分辨率（以像素為單位）>
       - theta：np.pi / 180.0 <累加器的角度分辨率（以弧度為單位）>
       - threshold：200



```
cv.HoughLines( image, rho, theta, threshold[, lines[, srn[, stn[, min_theta[, max_theta]]]]]
) -> lines
```

- Parameters:
  - `image` 8-bit, single-channel binary source image. The image may be modified by the function.
  - `lines` Output vector of lines. Each line is represented by a 2 or 3 element vector (ρ, θ) or $(\rho, \theta, votes)$, where ρ is the distance from the coordinate origin $(0, 0)$ (top-left corner of the image), θ is the line rotation angle in radians (0~vertical line, $\pi/2$~horizontal line), and votes is the value of accumulator.
  - `rho` Distance resolution of the accumulator in pixels.
  - `theta` Angle resolution of the accumulator in radians.
  - `threshold` Accumulator threshold parameter. Only those lines are returned that get enough votes (>threshold).
  - `srn` For the multi-scale Hough transform, it is a divisor for the distance resolution rho. The coarse accumulator distance resolution is rho and the accurate accumulator resolution is rho/srn. If both srn=0 and stn=0, the classical Hough transform is used. Otherwise, both these parameters should be positive.
  - `stn` For the multi-scale Hough transform, it is a divisor for the distance resolution theta.
  - `min_theta` For standard and multi-scale Hough transform, minimum angle to check for lines. Must fall between 0 and max_theta.
  - `max_theta` For standard and multi-scale Hough transform, an upper bound for the angle. Must fall between min_theta and CV_PI. The actual maximum angle in the accumulator may be slightly less than max_theta, depending on the parameters min_theta and theta.

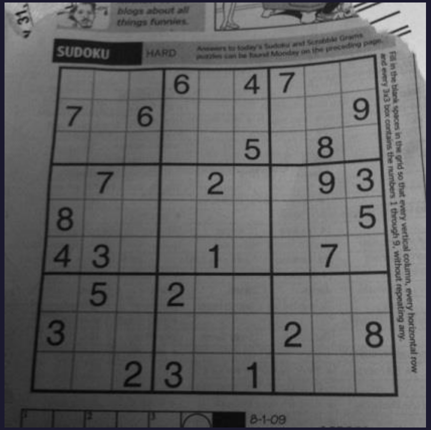
提示: 程式碼與結果 (sobel)



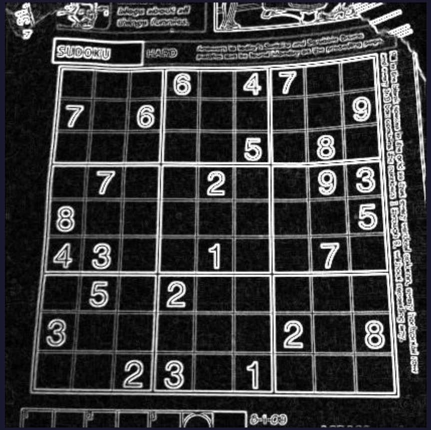
```python
folder = r'/content/drive/MyDrive/images'
path_img = os.path.join(folder,'sudoku.jpg')
img = cv2.imread(path_img)
# Afterwards, a check is executed, if the image was loaded correctly.
if img is None:
    sys.exit("Could not read the image.")
cv2_imshow(img)
img_gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
```

```python
# dst=cv.Sobel(src, ddepth, dx, dy[, dst[, ksize[, scale[, delta[, borderType]]]]])
def Sobel(image):
    # 導函數→偏微分→影像梯度
    # 以 cv2.CV_32F 型態做梯度計算
    [redacted]
    # 梯度的大小:將x和y方向梯度取絕對值
    grad_xy = abs(grad_x) + abs(grad_y)
    # np.clip()將值限定範圍在[0,255]
    sobel = np.uint8(np.clip(grad_xy, 0, 255))
    return sobel, grad_x, grad_y
```

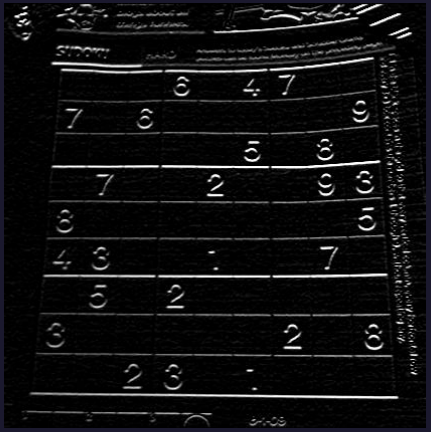
```python
img_sobel,grad_x,grad_y = Sobel(img_gray)
cv2_imshow(img_gray)
cv2_imshow(img_sobel)
cv2_imshow(grad_x)
cv2_imshow(grad_y)
```



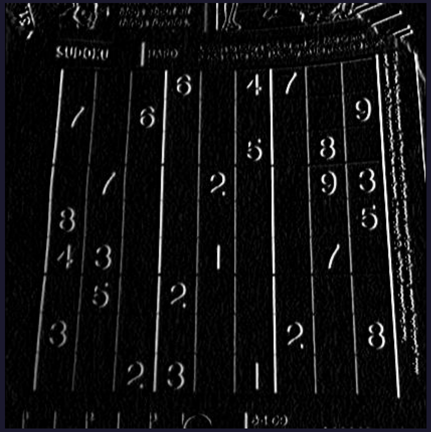
original



sobel



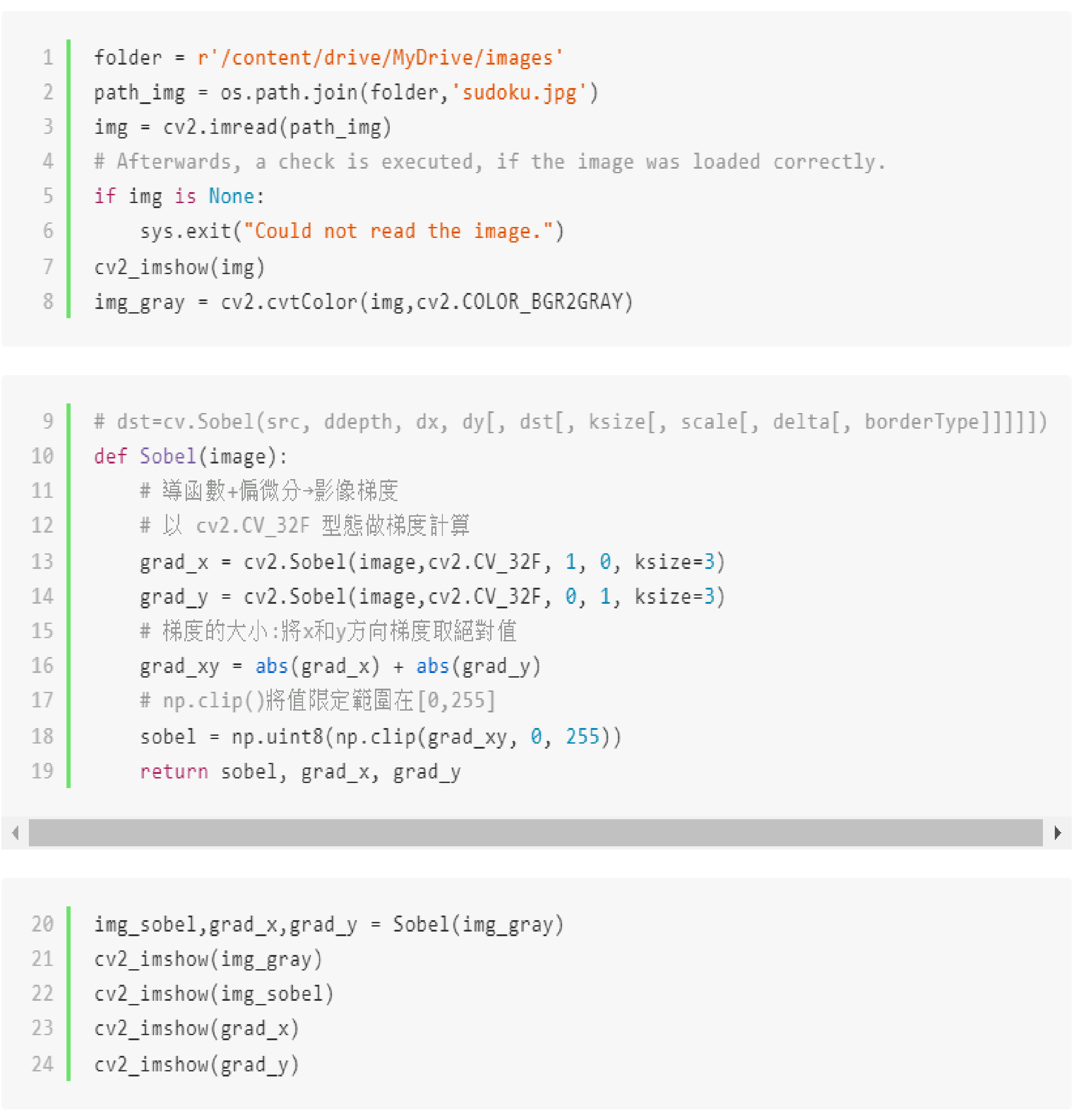
grad_x



grad_y

程式碼與結果 (sobel)

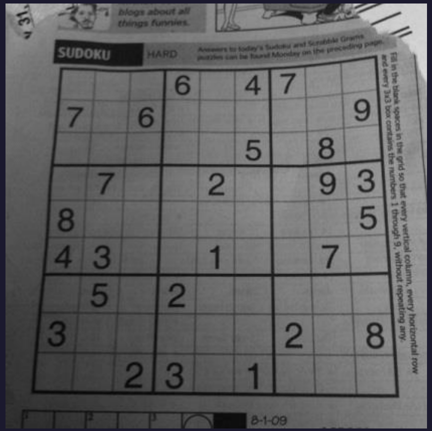
Colab : https://drive.google.com/file/d/1LwM64AchEKHW2a93lfiV1hVafrYzC5Tj/view?usp=sharing
HackMD : https://hackmd.io/@chanhts/rky7Hfwfj



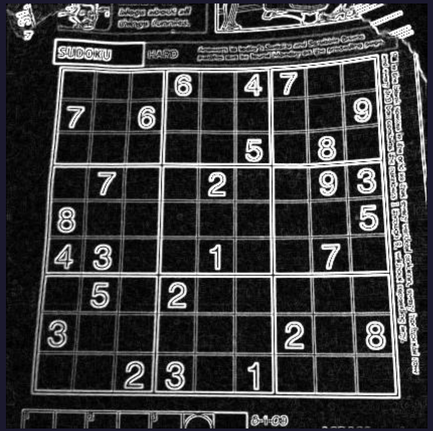
```
folder = r'/content/drive/MyDrive/images'
path_img = os.path.join(folder,'sudoku.jpg')
img = cv2.imread(path_img)
# Afterwards, a check is executed, if the image was loaded correctly.
if img is None:
    sys.exit("Could not read the image.")
cv2_imshow(img)
img_gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
```

```
# dst=cv.Sobel(src, ddepth, dx, dy[, dst[, ksize[, scale[, delta[, borderType]]]]])
def Sobel(image):
    # 導函數+偏微分→影像梯度
    # 以 cv2.CV_32F 型態做梯度計算
    grad_x = cv2.Sobel(image,cv2.CV_32F, 1, 0, ksize=3)
    grad_y = cv2.Sobel(image,cv2.CV_32F, 0, 1, ksize=3)
    # 梯度的大小:將x和y方向梯度取絕對值
    grad_xy = abs(grad_x) + abs(grad_y)
    # np.clip()將值限定範圍在[0,255]
    sobel = np.uint8(np.clip(grad_xy, 0, 255))
    return sobel, grad_x, grad_y
```

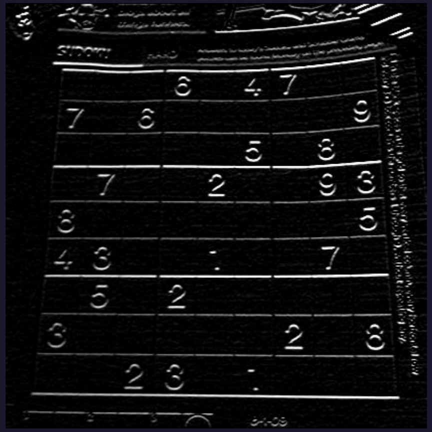
```
img_sobel,grad_x,grad_y = Sobel(img_gray)
cv2_imshow(img_gray)
cv2_imshow(img_sobel)
cv2_imshow(grad_x)
cv2_imshow(grad_y)
```



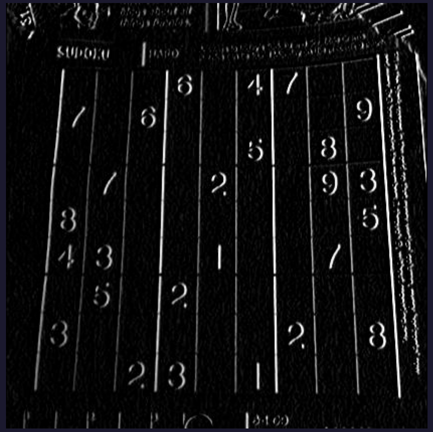
original



sobel

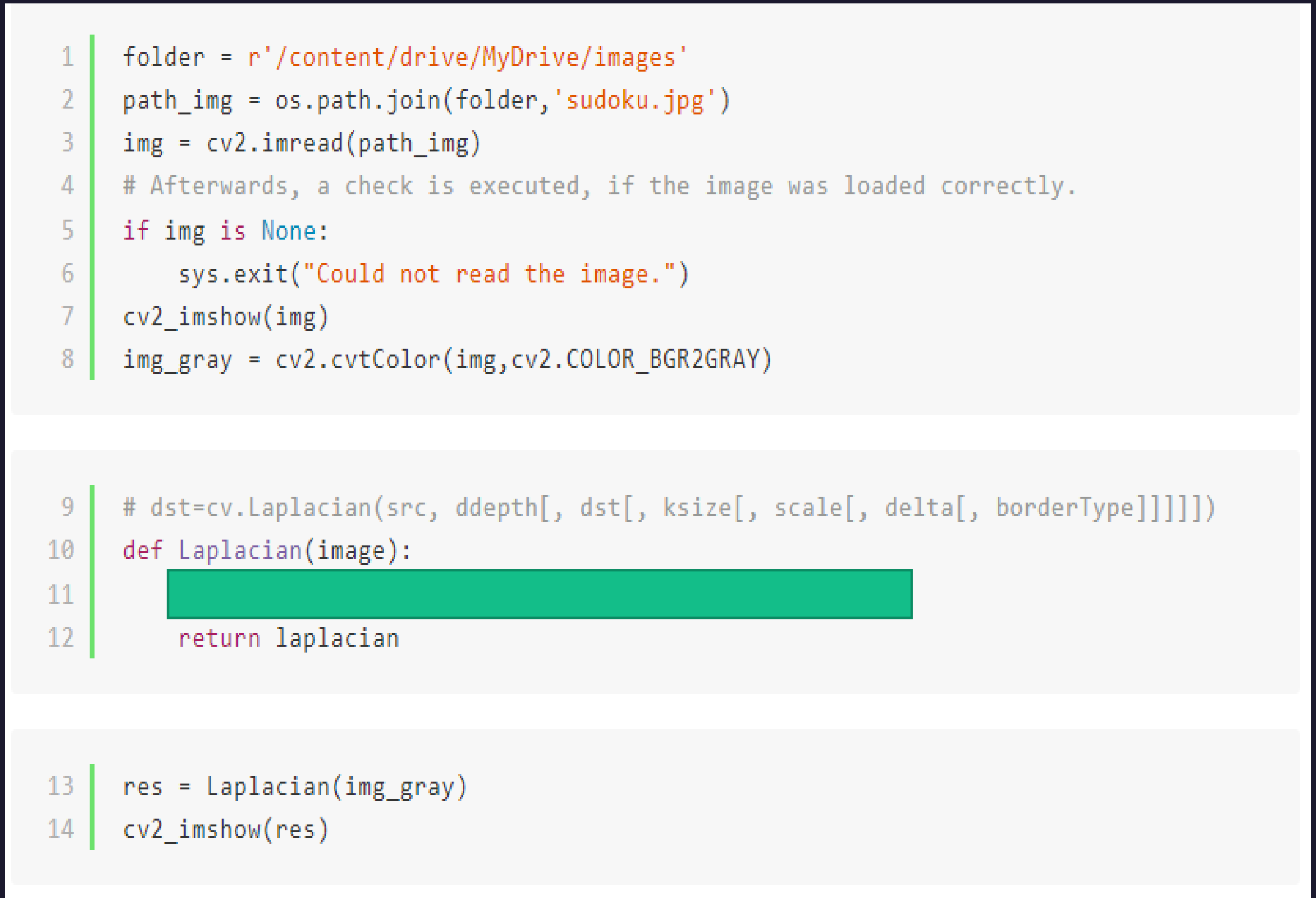


grad_x



grad_y

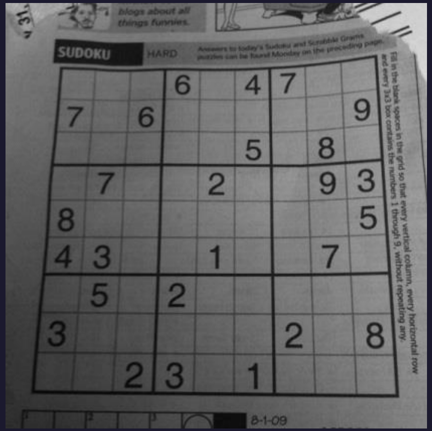
提示: 程式碼與結果 (Laplacian)



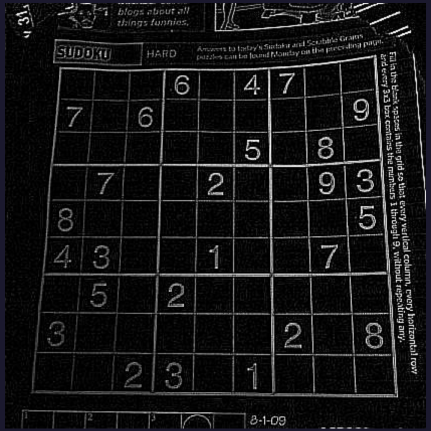
```python
folder = r'/content/drive/MyDrive/images'
path_img = os.path.join(folder,'sudoku.jpg')
img = cv2.imread(path_img)
# Afterwards, a check is executed, if the image was loaded correctly.
if img is None:
    sys.exit("Could not read the image.")
cv2_imshow(img)
img_gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
```

```python
# dst=cv.Laplacian(src, ddepth[, dst[, ksize[, scale[, delta[, borderType]]]]])
def Laplacian(image):
    [redacted]
    return laplacian
```

```python
res = Laplacian(img_gray)
cv2_imshow(res)
```



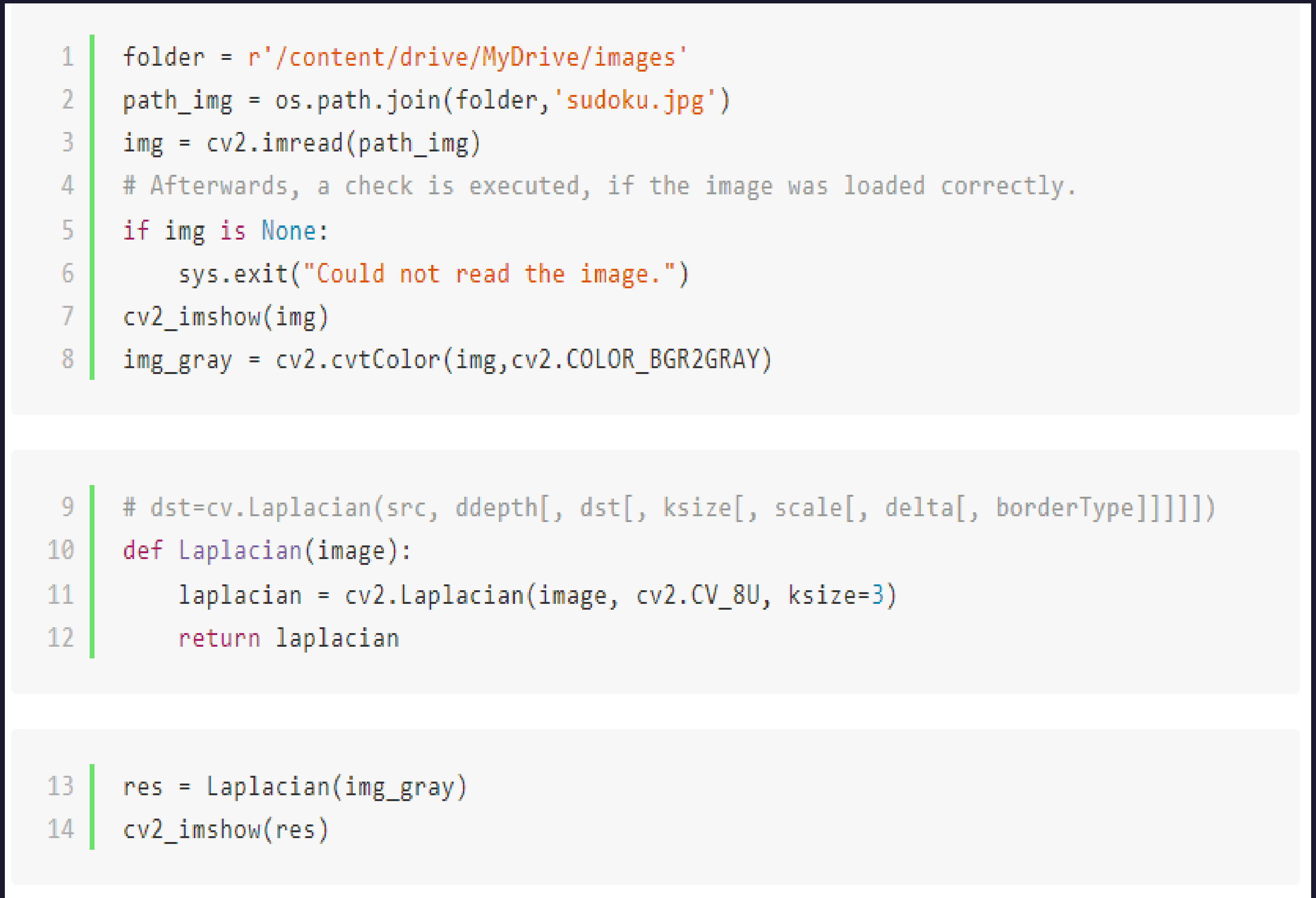
original



Laplacian

程式碼與結果 (Laplacian)

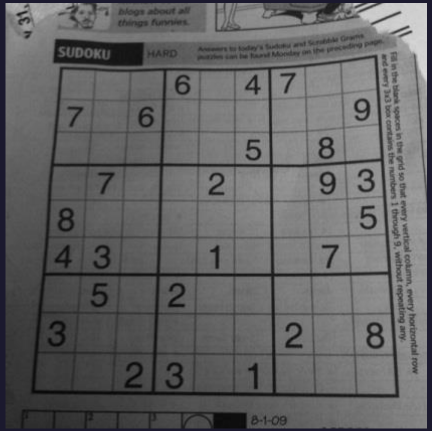
Colab : https://colab.research.google.com/drive/1Cm6ko2exZp91uFZfNj9wMj2i93SwLQWS?usp=sharing
HackMD : https://hackmd.io/@chanhts/rky7Hfwfj



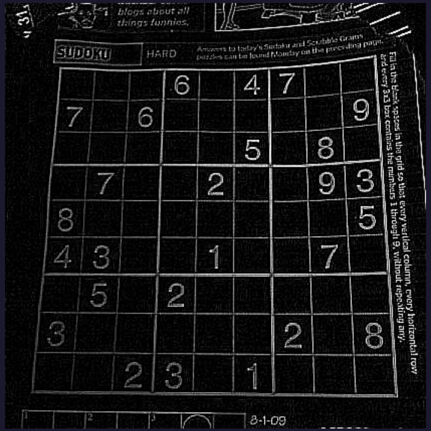
```python
folder = r'/content/drive/MyDrive/images'
path_img = os.path.join(folder,'sudoku.jpg')
img = cv2.imread(path_img)
# Afterwards, a check is executed, if the image was loaded correctly.
if img is None:
    sys.exit("Could not read the image.")
cv2_imshow(img)
img_gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
```

```python
# dst=cv.Laplacian(src, ddepth[, dst[, ksize[, scale[, delta[, borderType]]]]])
def Laplacian(image):
    laplacian = cv2.Laplacian(image, cv2.CV_8U, ksize=3)
    return laplacian
```

```python
res = Laplacian(img_gray)
cv2_imshow(res)
```

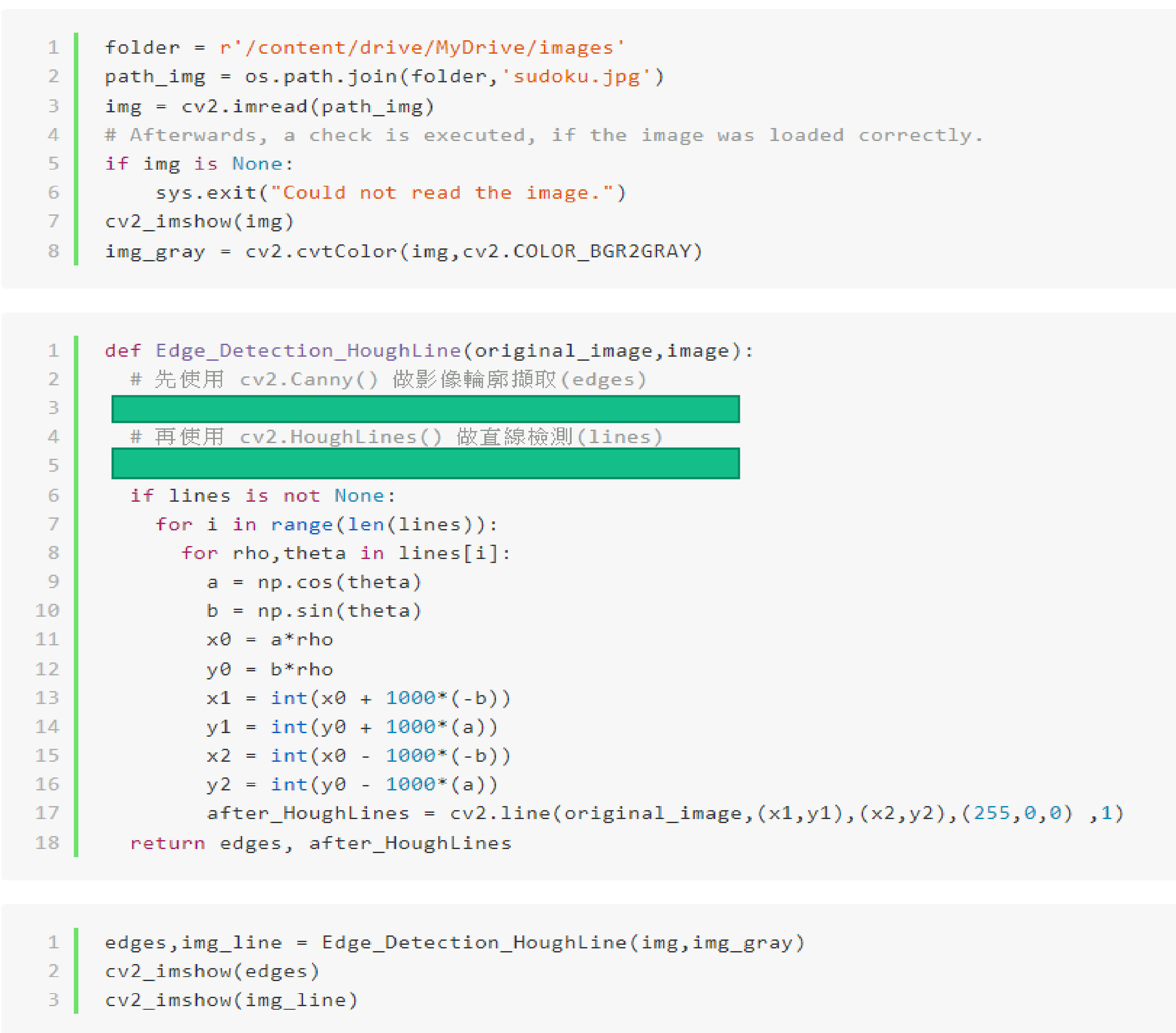


original



Laplacian

提示: 程式碼與結果 (Hough: line)

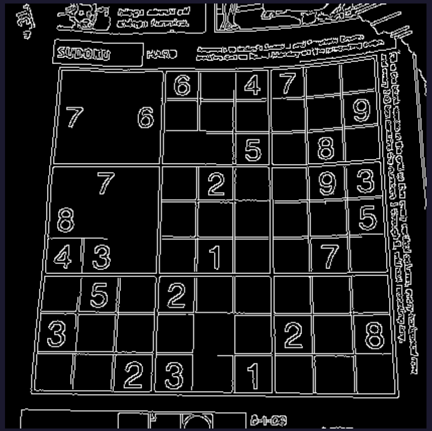


```
folder = r'/content/drive/MyDrive/images'
path_img = os.path.join(folder,'sudoku.jpg')
img = cv2.imread(path_img)
# Afterwards, a check is executed, if the image was loaded correctly.
if img is None:
    sys.exit("Could not read the image.")
cv2_imshow(img)
img_gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
```

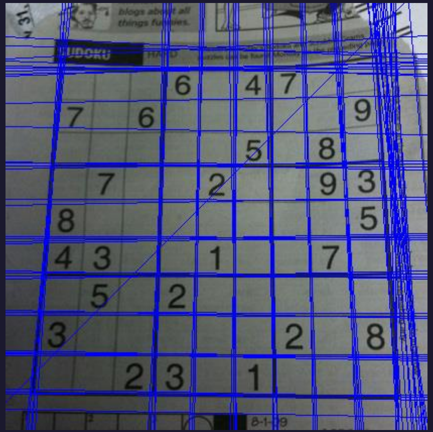
```
def Edge_Detection_HoughLine(original_image,image):
  # 先使用 cv2.Canny() 做影像輪廓擷取(edges)

  # 再使用 cv2.HoughLines() 做直線檢測(lines)

  if lines is not None:
    for i in range(len(lines)):
      for rho,theta in lines[i]:
        a = np.cos(theta)
        b = np.sin(theta)
        x0 = a*rho
        y0 = b*rho
        x1 = int(x0 + 1000*(-b))
        y1 = int(y0 + 1000*(a))
        x2 = int(x0 - 1000*(-b))
        y2 = int(y0 - 1000*(a))
        after_HoughLines = cv2.line(original_image,(x1,y1),(x2,y2),(255,0,0) ,1)
  return edges, after_HoughLines
```

```
edges,img_line = Edge_Detection_HoughLine(img,img_gray)
cv2_imshow(edges)
cv2_imshow(img_line)
```

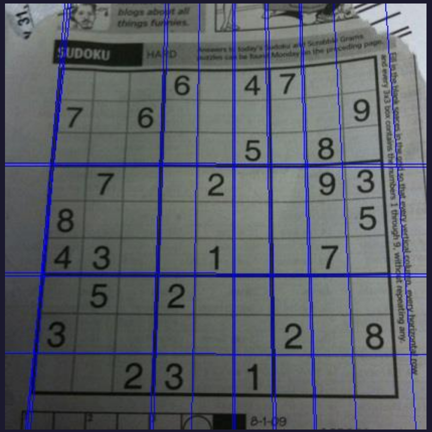
original



canny



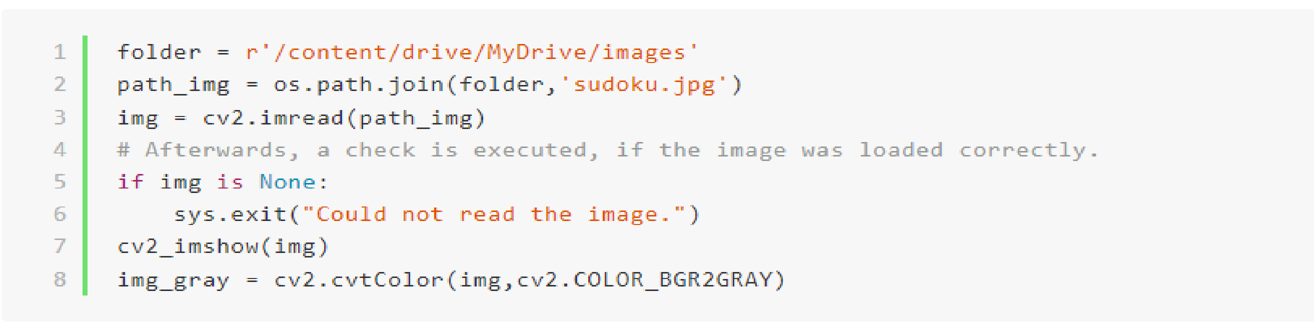
Hough_line (threshold=120)



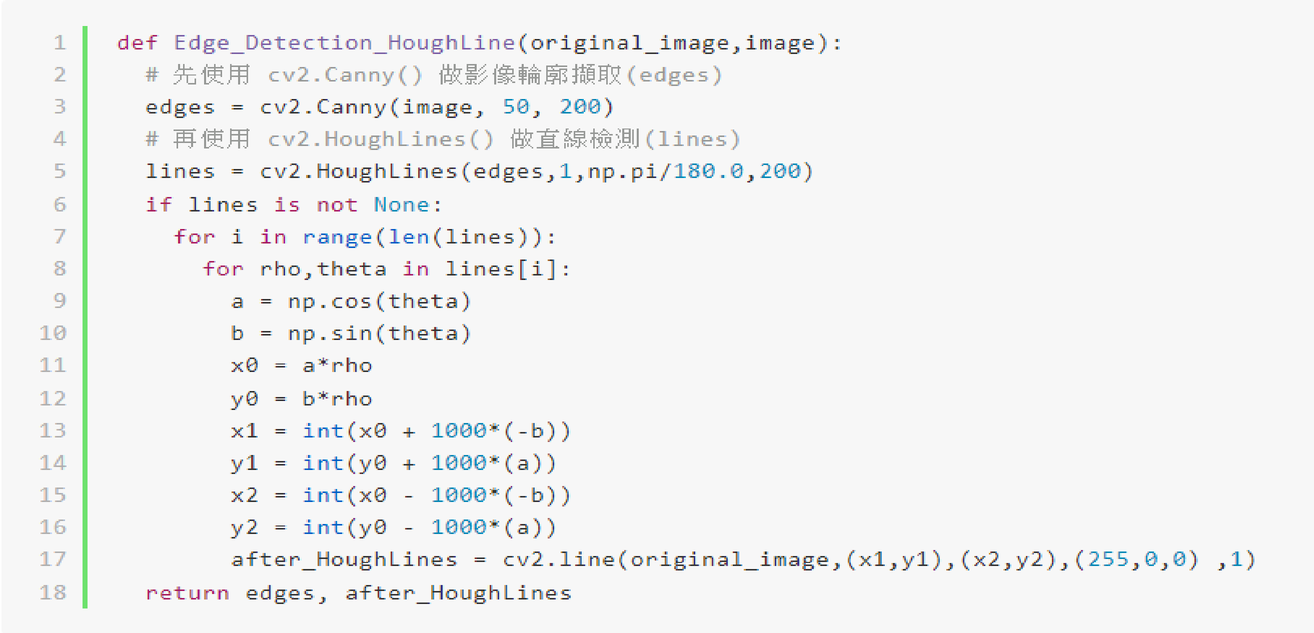
Hough_line (threshold=200)

程式碼與結果 (Hough: line)

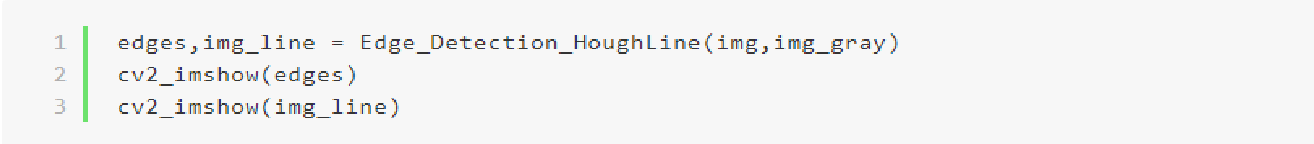
Colab : https://drive.google.com/file/d/1iVYykWNzwo3D0Xvx1f-8mILNNt1mJCeI/view?usp=sharing
HackMD : https://hackmd.io/@chanhts/rky7Hfwfj



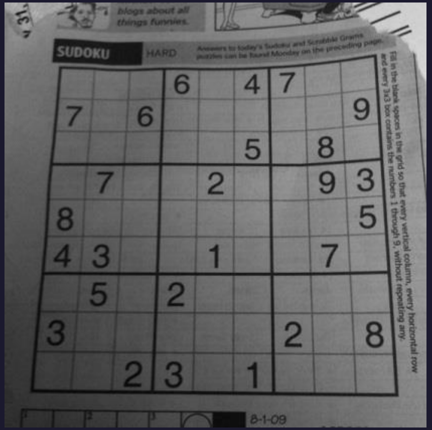
```python
folder = r'/content/drive/MyDrive/images'
path_img = os.path.join(folder,'sudoku.jpg')
img = cv2.imread(path_img)
# Afterwards, a check is executed, if the image was loaded correctly.
if img is None:
    sys.exit("Could not read the image.")
cv2_imshow(img)
img_gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
```



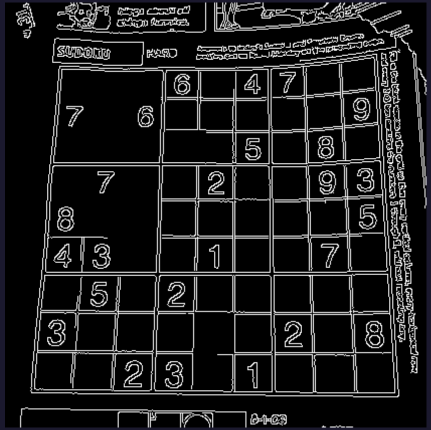
```python
def Edge_Detection_HoughLine(original_image,image):
  # 先使用 cv2.Canny() 做影像輪廓擷取(edges)
  edges = cv2.Canny(image, 50, 200)
  # 再使用 cv2.HoughLines() 做直線檢測(lines)
  lines = cv2.HoughLines(edges,1,np.pi/180.0,200)
  if lines is not None:
    for i in range(len(lines)):
      for rho,theta in lines[i]:
        a = np.cos(theta)
        b = np.sin(theta)
        x0 = a*rho
        y0 = b*rho
        x1 = int(x0 + 1000*(-b))
        y1 = int(y0 + 1000*(a))
        x2 = int(x0 - 1000*(-b))
        y2 = int(y0 - 1000*(a))
        after_HoughLines = cv2.line(original_image,(x1,y1),(x2,y2),(255,0,0) ,1)
  return edges, after_HoughLines
```



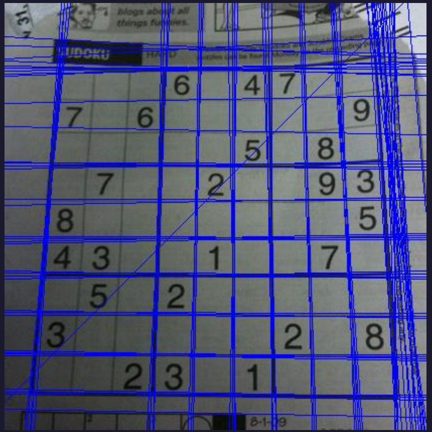
```python
edges,img_line = Edge_Detection_HoughLine(img,img_gray)
cv2_imshow(edges)
cv2_imshow(img_line)
```



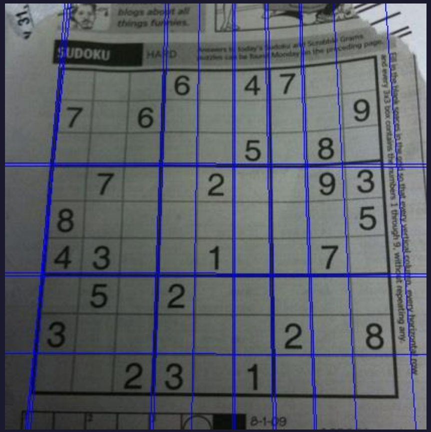
original



canny



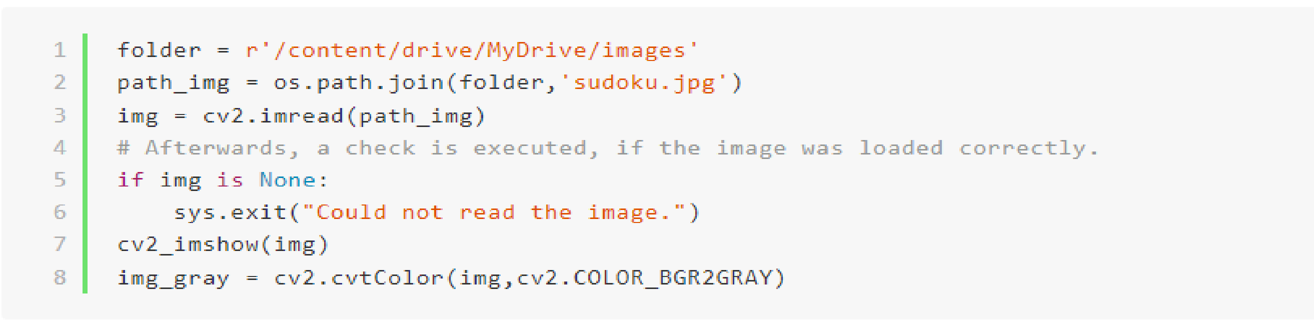
Hough_line (threshold=120)



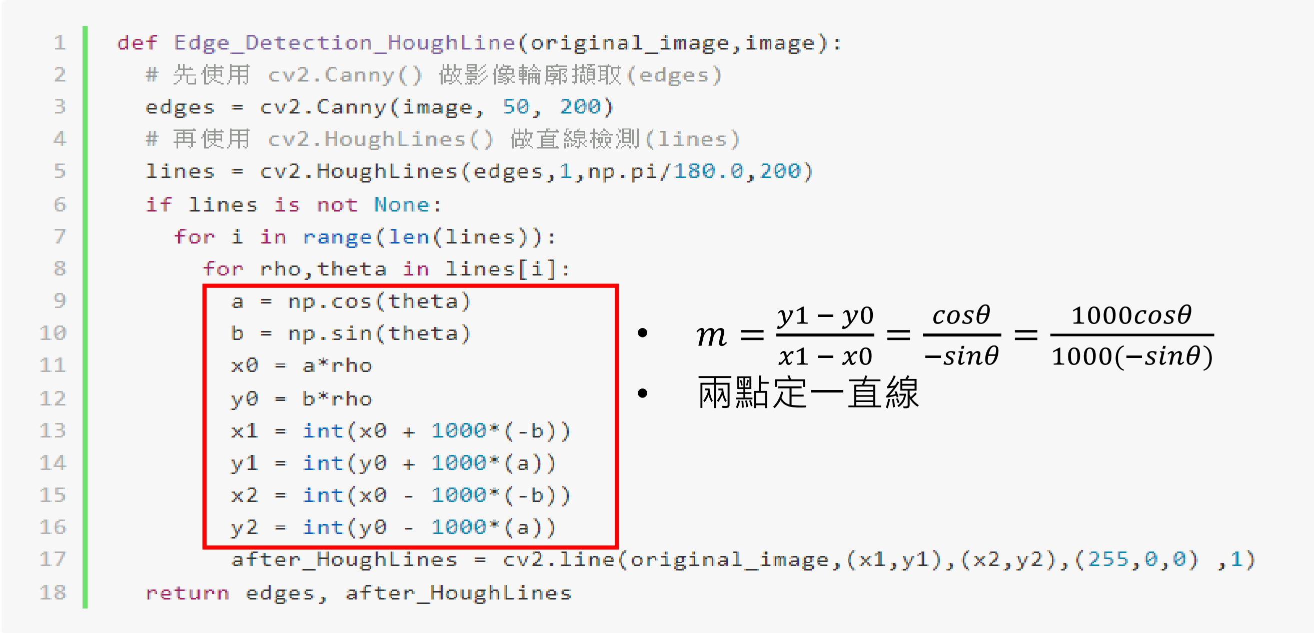
Hough_line (threshold=200)

程式碼與結果 (Hough: line)

Colab : https://drive.google.com/file/d/1iVYykWNzwo3D0Xvx1f-8mILNNt1mJCeI/view?usp=sharing
HackMD : https://hackmd.io/@chanhts/rky7Hfwfj

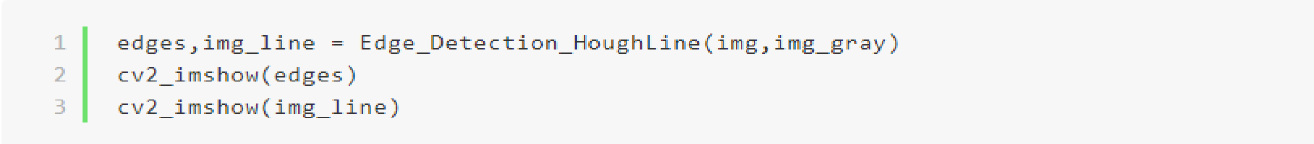


```python
folder = r'/content/drive/MyDrive/images'
path_img = os.path.join(folder,'sudoku.jpg')
img = cv2.imread(path_img)
# Afterwards, a check is executed, if the image was loaded correctly.
if img is None:
    sys.exit("Could not read the image.")
cv2_imshow(img)
img_gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
```

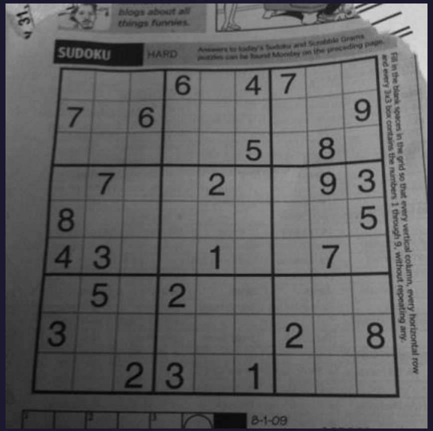


```python
def Edge_Detection_HoughLine(original_image,image):
  # 先使用 cv2.Canny() 做影像輪廓擷取(edges)
  edges = cv2.Canny(image, 50, 200)
  # 再使用 cv2.HoughLines() 做直線檢測(lines)
  lines = cv2.HoughLines(edges,1,np.pi/180.0,200)
  if lines is not None:
    for i in range(len(lines)):
      for rho,theta in lines[i]:
        a = np.cos(theta)
        b = np.sin(theta)
        x0 = a*rho
        y0 = b*rho
        x1 = int(x0 + 1000*(-b))
        y1 = int(y0 + 1000*(a))
        x2 = int(x0 - 1000*(-b))
        y2 = int(y0 - 1000*(a))
        after_HoughLines = cv2.line(original_image,(x1,y1),(x2,y2),(255,0,0) ,1)
  return edges, after_HoughLines
```

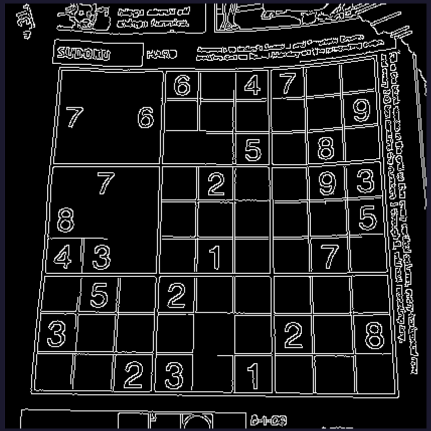
- $m = \frac{y1-y0}{x1-x0} = \frac{cos\theta}{-sin\theta} = \frac{1000cos\theta}{1000(-sin\theta)}$
- 兩點定一直線



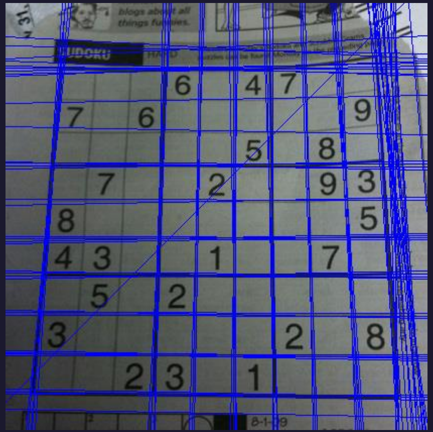
```python
edges,img_line = Edge_Detection_HoughLine(img,img_gray)
cv2_imshow(edges)
cv2_imshow(img_line)
```



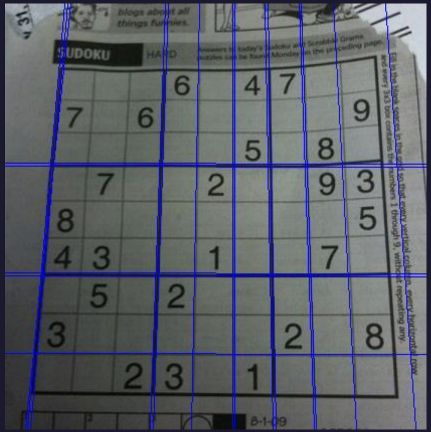
original



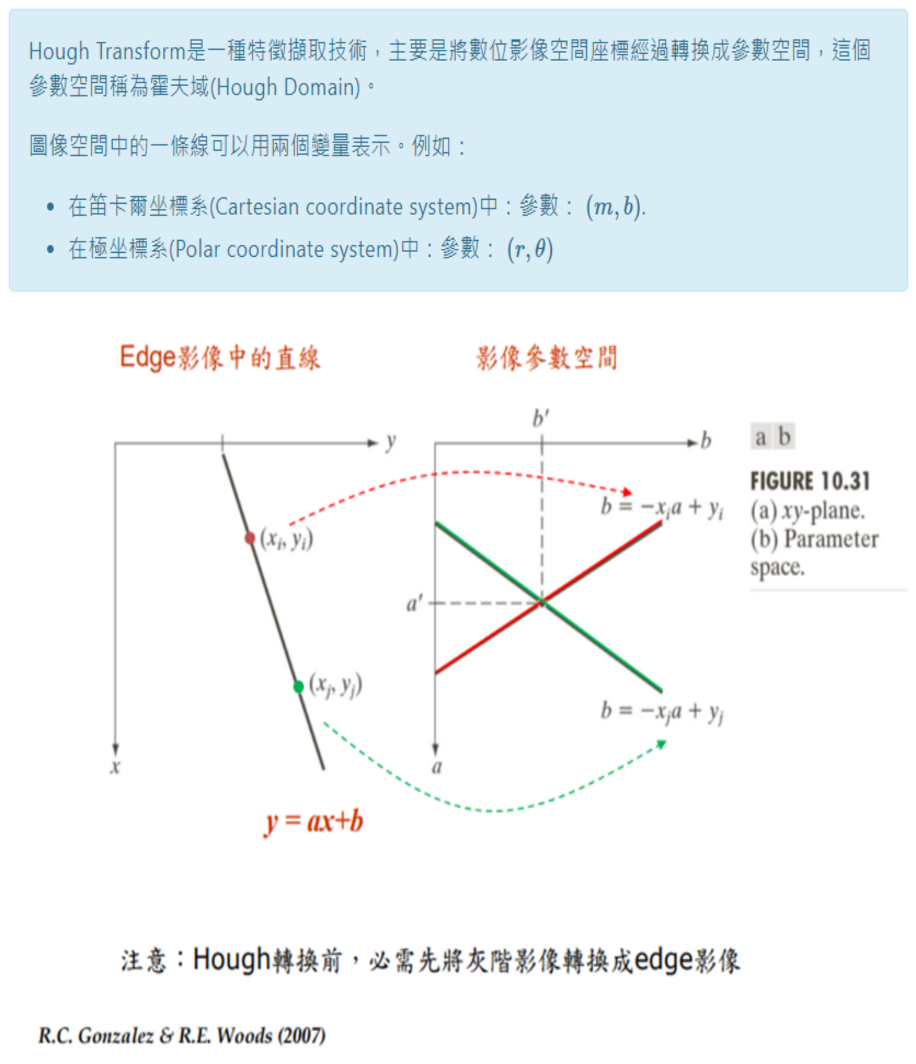
canny



Hough_line
(threshold=120
)



Hough_line
(threshold=200)



Hough Transform是一種特徵擷取技術，主要是將數位影像空間座標經過轉換成參數空間，這個參數空間稱為霍夫域(Hough Domain)。

圖像空間中的一條線可以用兩個變量表示。例如：

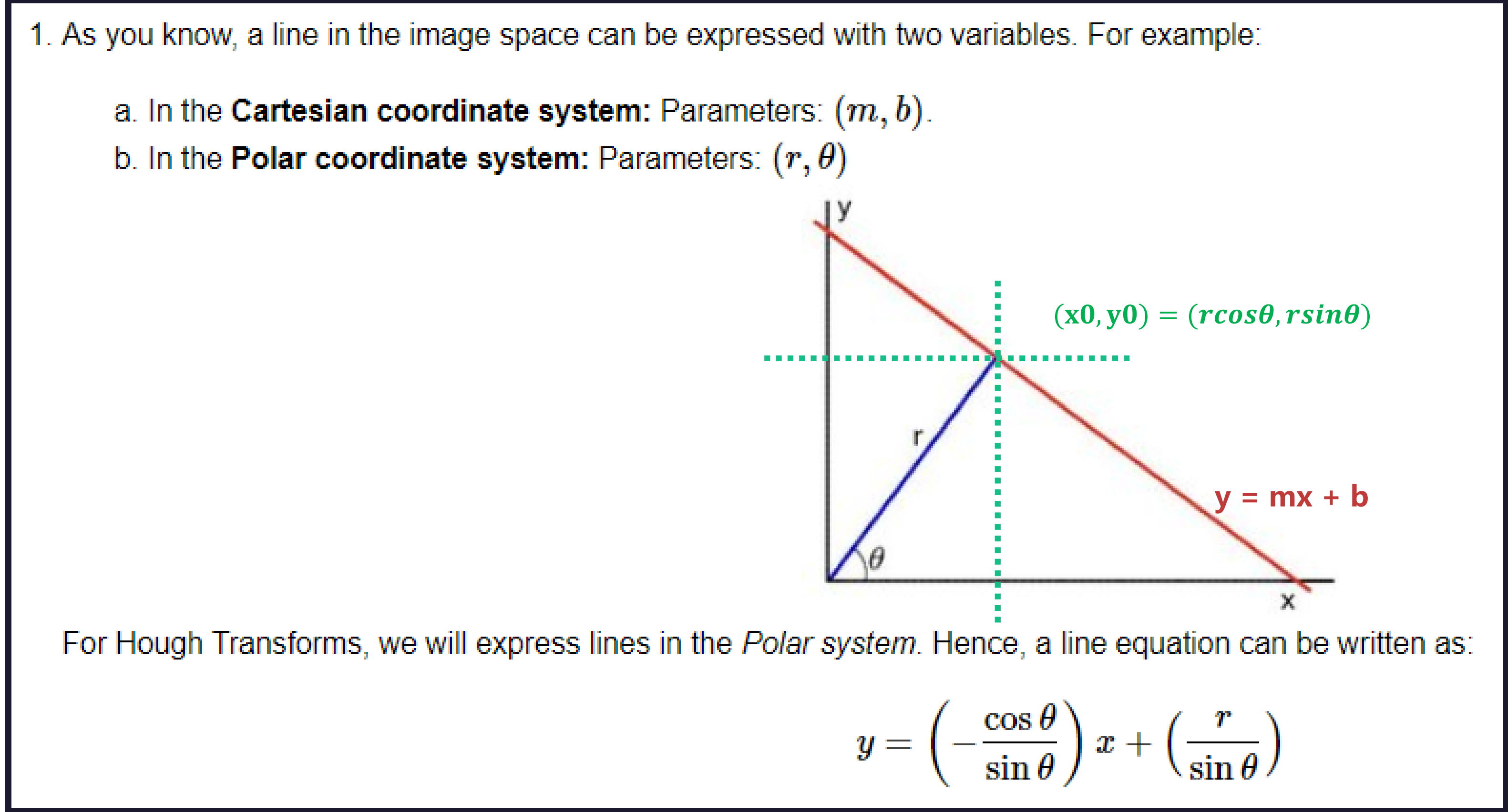
- 在笛卡爾坐標系(Cartesian coordinate system)中：參數：(m,b).
- 在極坐標系(Polar coordinate system)中：參數：(r,θ)

a b

FIGURE 10.31 (a) xy-plane. (b) Parameter space.

注意：Hough轉換前，必需先將灰階影像轉換成edge影像

R.C. Gonzalez & R.E. Woods (2007)



1. As you know, a line in the image space can be expressed with two variables. For example:

    a. In the **Cartesian coordinate system:** Parameters: (m,b).

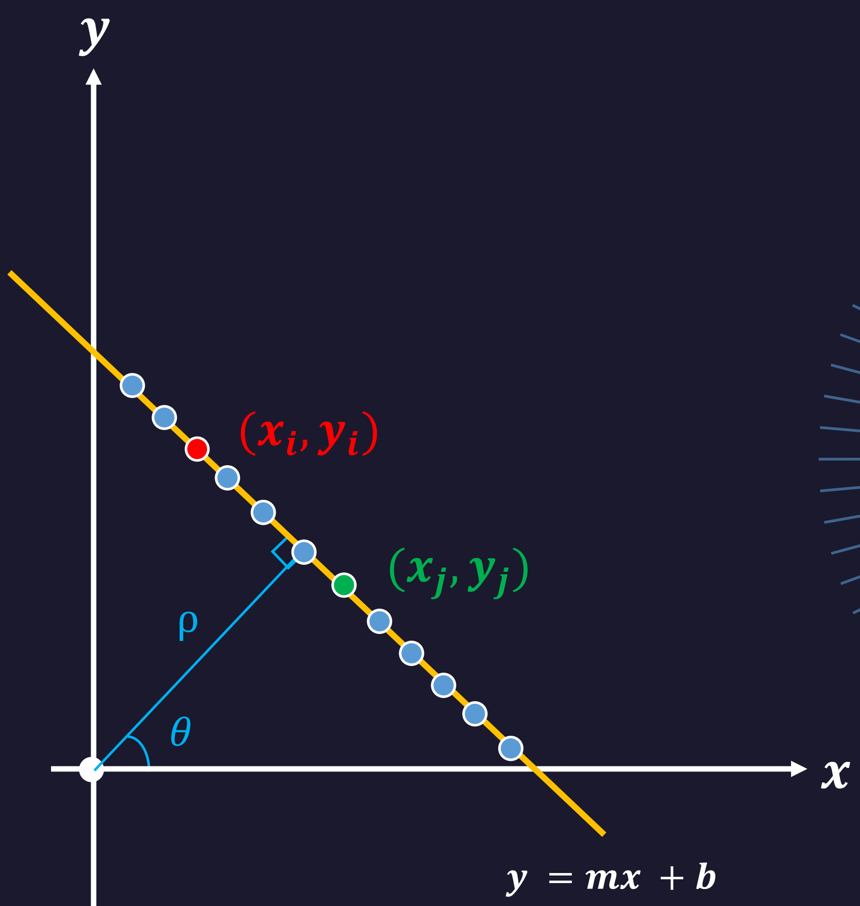
    b. In the **Polar coordinate system:** Parameters: (r,θ)

For Hough Transforms, we will express lines in the *Polar system*. Hence, a line equation can be written as:

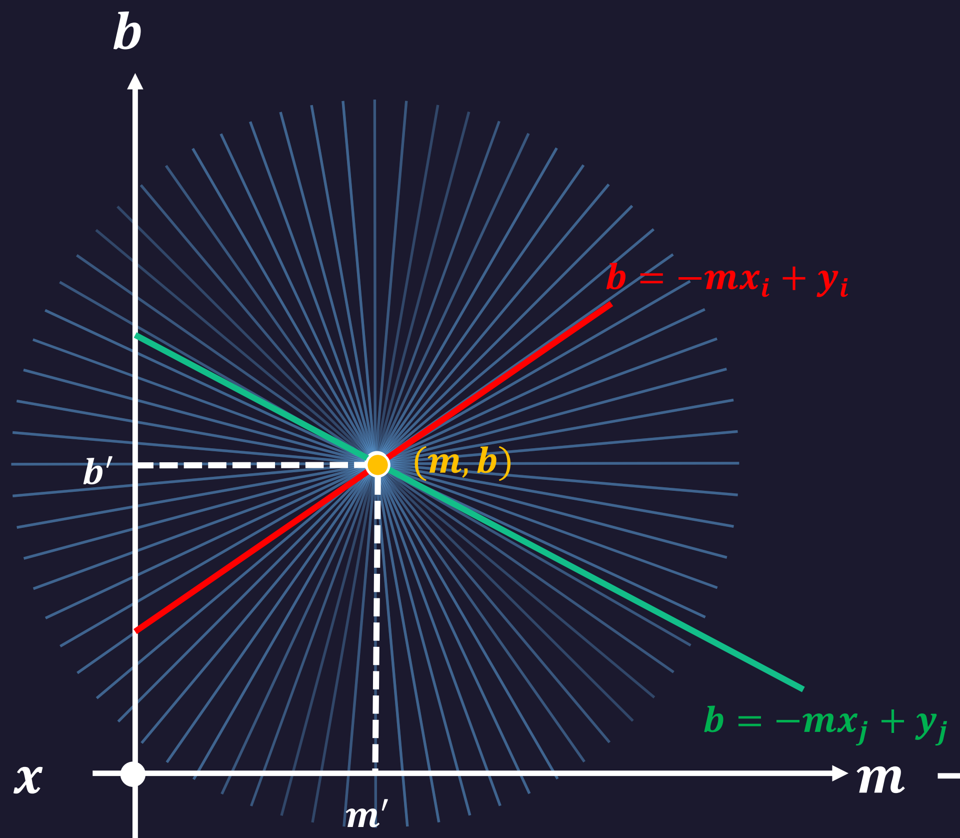
$$y = \left(-\frac{\cos\theta}{\sin\theta}\right)x + \left(\frac{r}{\sin\theta}\right)$$

斜率m　截距b

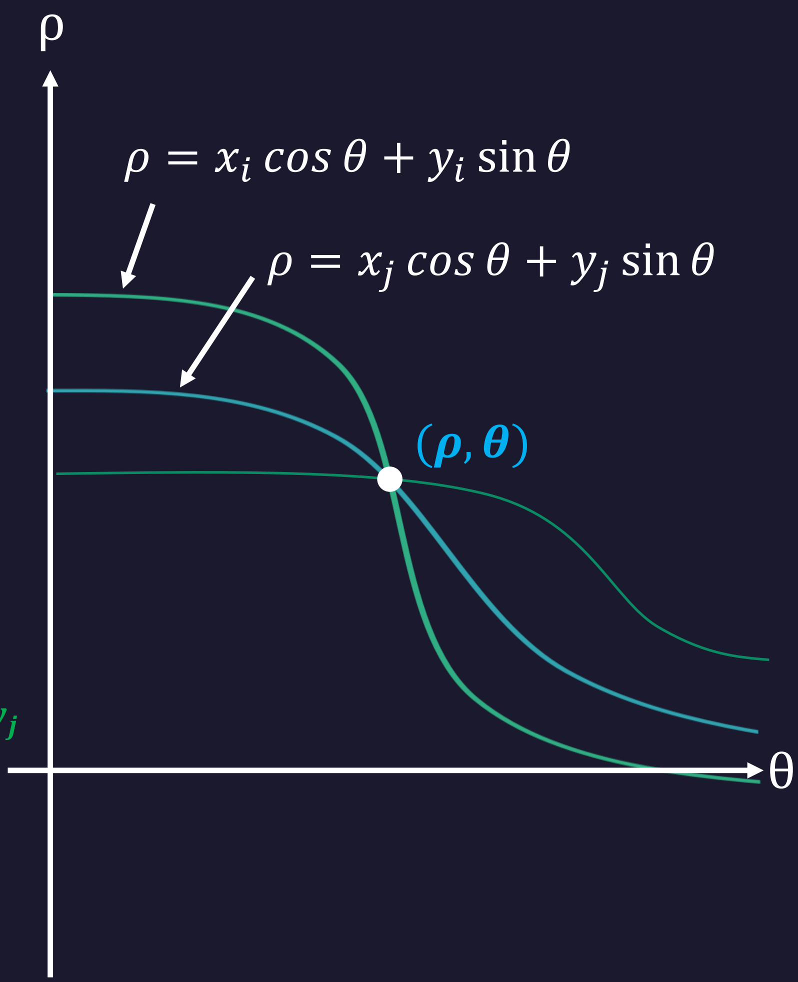
- 缺點: 當 xy 空間中直線趨近垂直時，參數空間中的斜率(m)會趨近無限大
- 改進方式: 將直角坐標改為極座標(ρ, θ)表示，則弦波曲線表示為: $x\cos\theta + y\sin\theta = \rho$



影像空間
(Image Space)



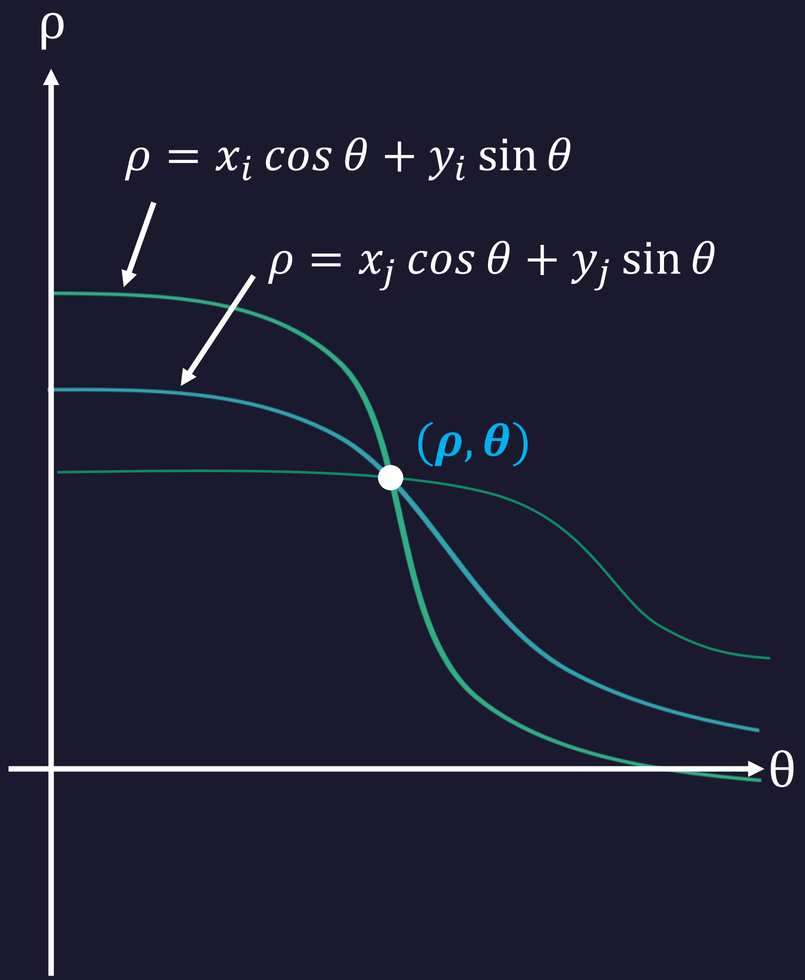
參數空間
(Parameter Space)



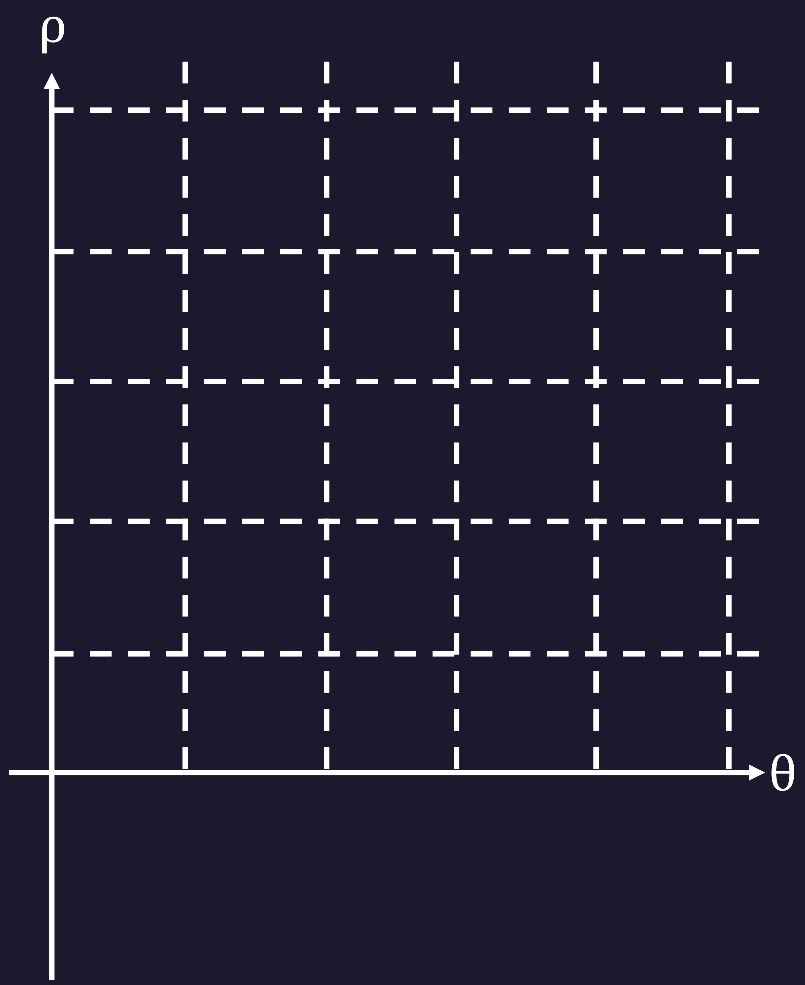
ρ
$\rho = x_i \cos\theta + y_i \sin\theta$
$\rho = x_j \cos\theta + y_j \sin\theta$
(ρ, θ)
θ

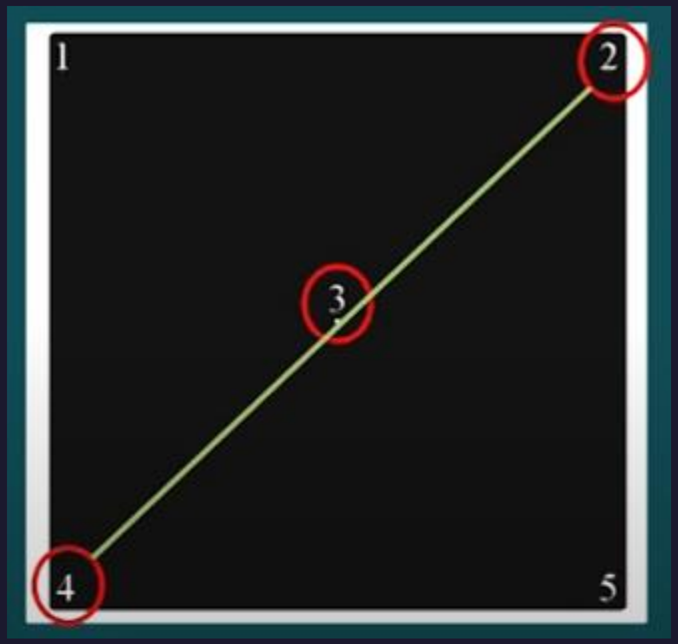
霍夫空間
(Hough Space)

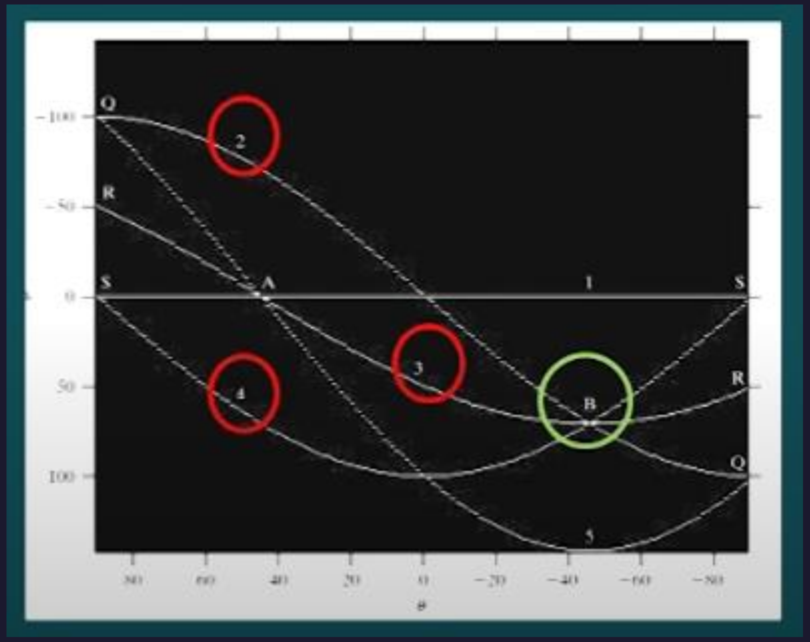
- 將(ρ, θ)平面分割成累計單元(accumulator cell)，用於紀錄非背景點


ρ
$\rho = x_i \, cos \, \theta + y_i \sin \theta$
$\rho = x_j \, cos \, \theta + y_j \sin \theta$
$(\boldsymbol{\rho}, \boldsymbol{\theta})$
θ

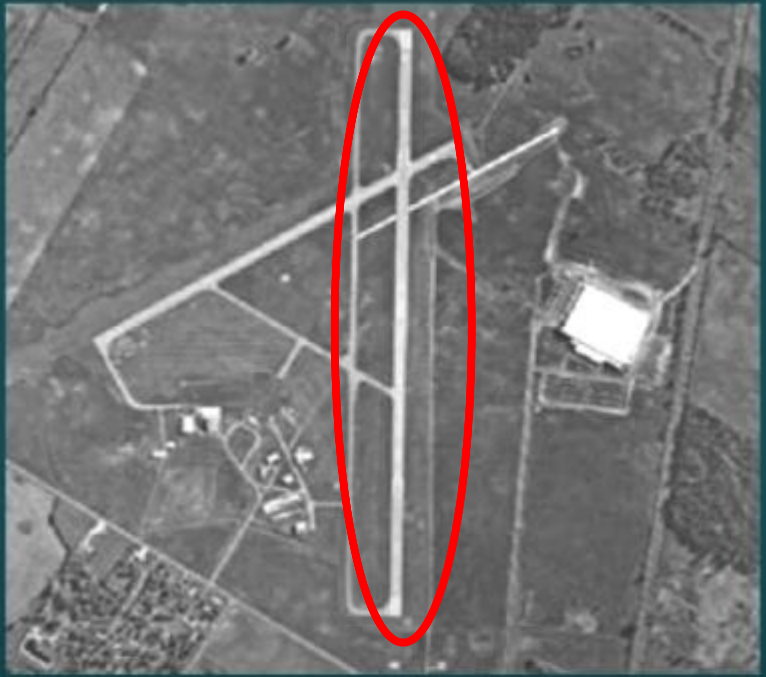
霍夫空間
(Hough Space)



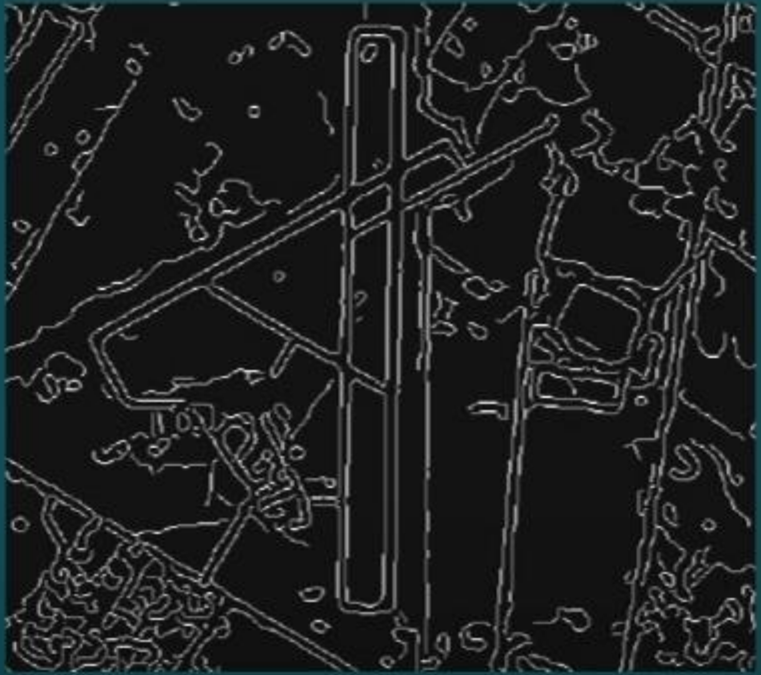




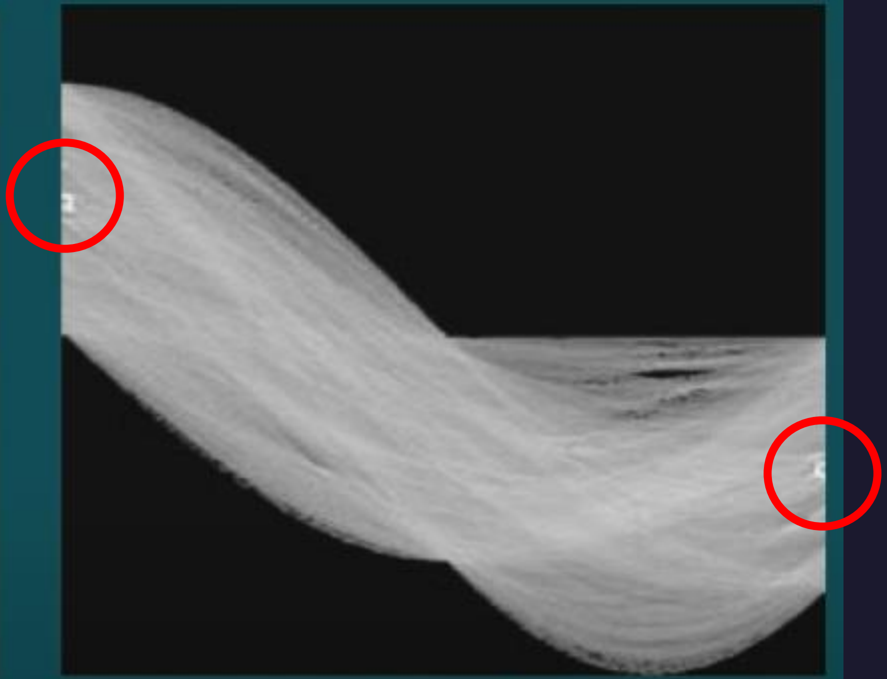
由此密集處兩點定義出兩條直線



機場空照影像



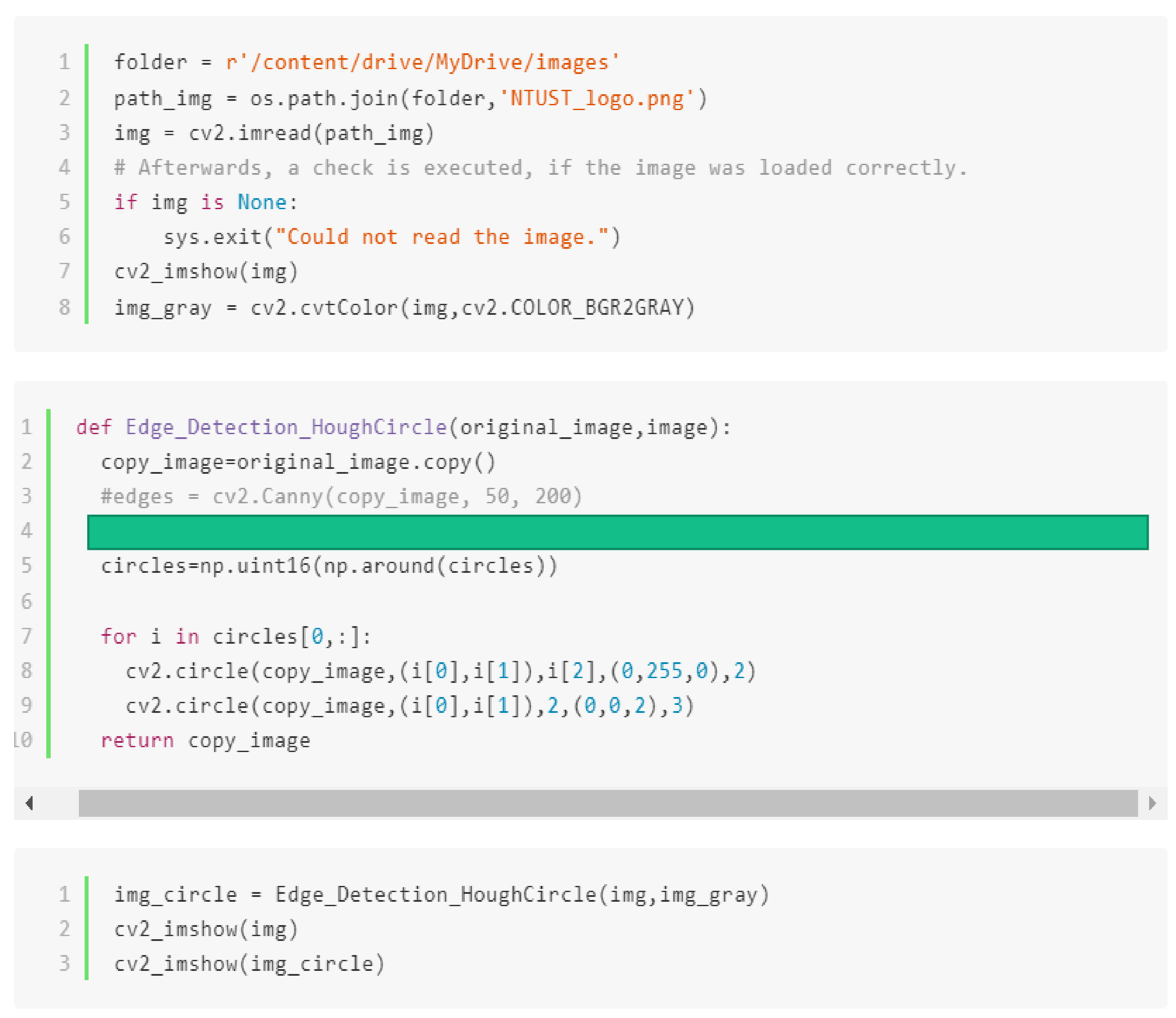
Canny邊緣



Hough轉換

Thanks for listening

提示: 程式碼與結果 (Hough: circle)

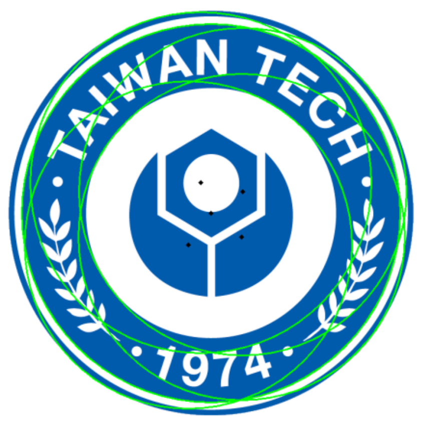


```
folder = r'/content/drive/MyDrive/images'
path_img = os.path.join(folder,'NTUST_logo.png')
img = cv2.imread(path_img)
# Afterwards, a check is executed, if the image was loaded correctly.
if img is None:
    sys.exit("Could not read the image.")
cv2_imshow(img)
img_gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
```

```
def Edge_Detection_HoughCircle(original_image,image):
  copy_image=original_image.copy()
  #edges = cv2.Canny(copy_image, 50, 200)

  circles=np.uint16(np.around(circles))

  for i in circles[0,:]:
    cv2.circle(copy_image,(i[0],i[1]),i[2],(0,255,0),2)
    cv2.circle(copy_image,(i[0],i[1]),2,(0,0,2),3)
  return copy_image
```

```
img_circle = Edge_Detection_HoughCircle(img,img_gray)
cv2_imshow(img)
cv2_imshow(img_circle)
```

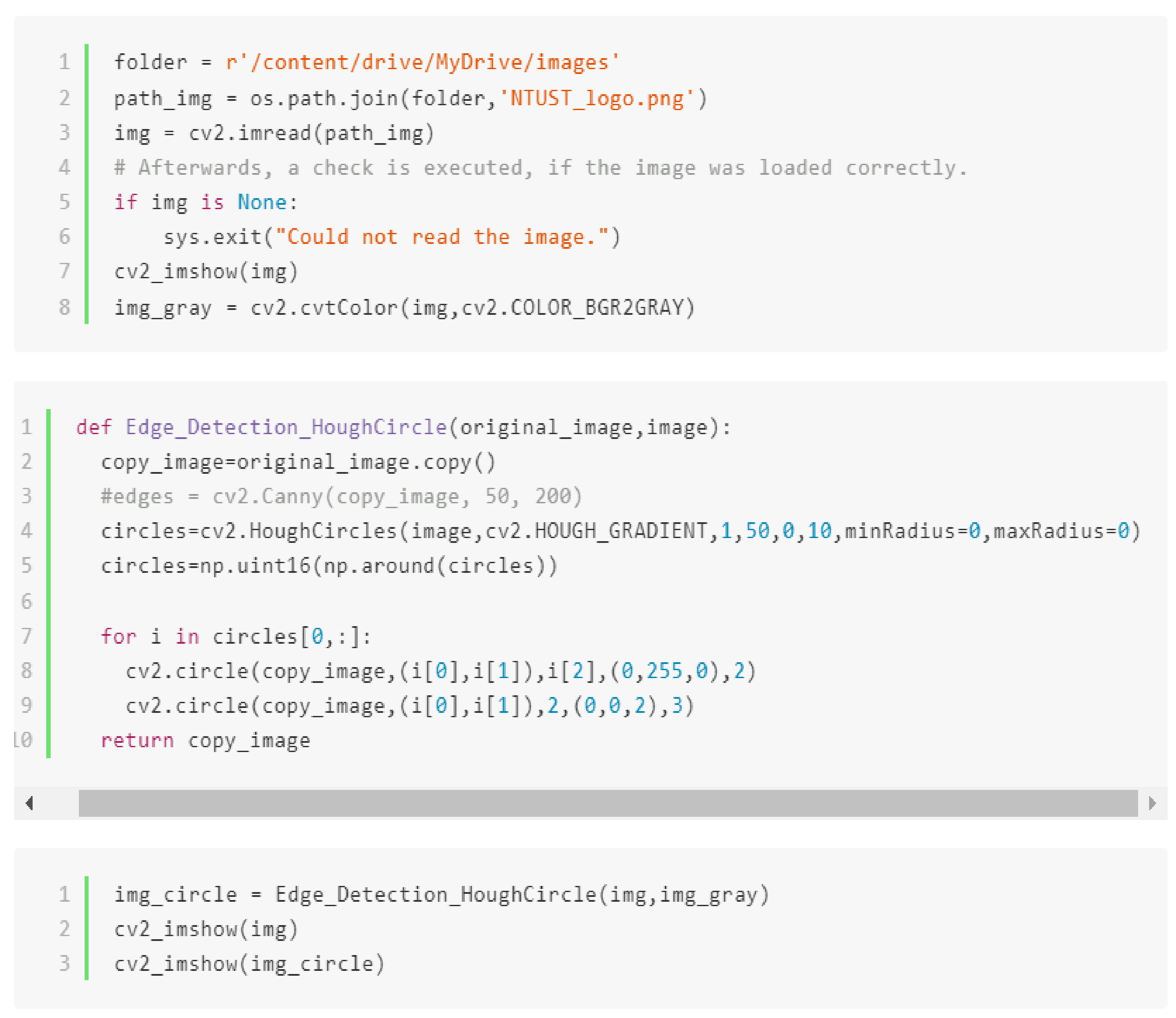
original



Hough_circle

程式碼與結果 (Hough: circle)

Colab : https://colab.research.google.com/drive/1lWsw7NSfSPWlB0MM0rGLmsn2gI0XVN3q?usp=sharing
HackMD : https://hackmd.io/@chanhts/rky7Hfwfj

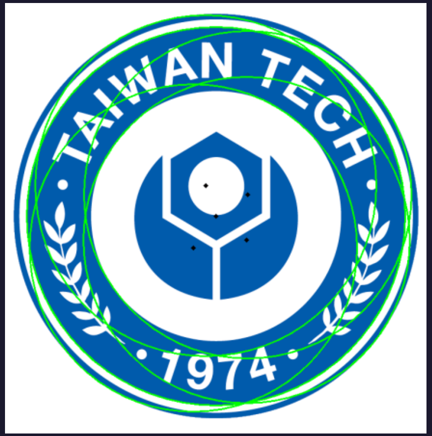


```python
folder = r'/content/drive/MyDrive/images'
path_img = os.path.join(folder,'NTUST_logo.png')
img = cv2.imread(path_img)
# Afterwards, a check is executed, if the image was loaded correctly.
if img is None:
    sys.exit("Could not read the image.")
cv2_imshow(img)
img_gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
```

```python
def Edge_Detection_HoughCircle(original_image,image):
  copy_image=original_image.copy()
  #edges = cv2.Canny(copy_image, 50, 200)
  circles=cv2.HoughCircles(image,cv2.HOUGH_GRADIENT,1,50,0,10,minRadius=0,maxRadius=0)
  circles=np.uint16(np.around(circles))

  for i in circles[0,:]:
    cv2.circle(copy_image,(i[0],i[1]),i[2],(0,255,0),2)
    cv2.circle(copy_image,(i[0],i[1]),2,(0,0,2),3)
  return copy_image
```

```python
img_circle = Edge_Detection_HoughCircle(img,img_gray)
cv2_imshow(img)
cv2_imshow(img_circle)
```

original



Hough_circle