黃瑞得
0860078
huangjuite@gmail.com

project5

source code

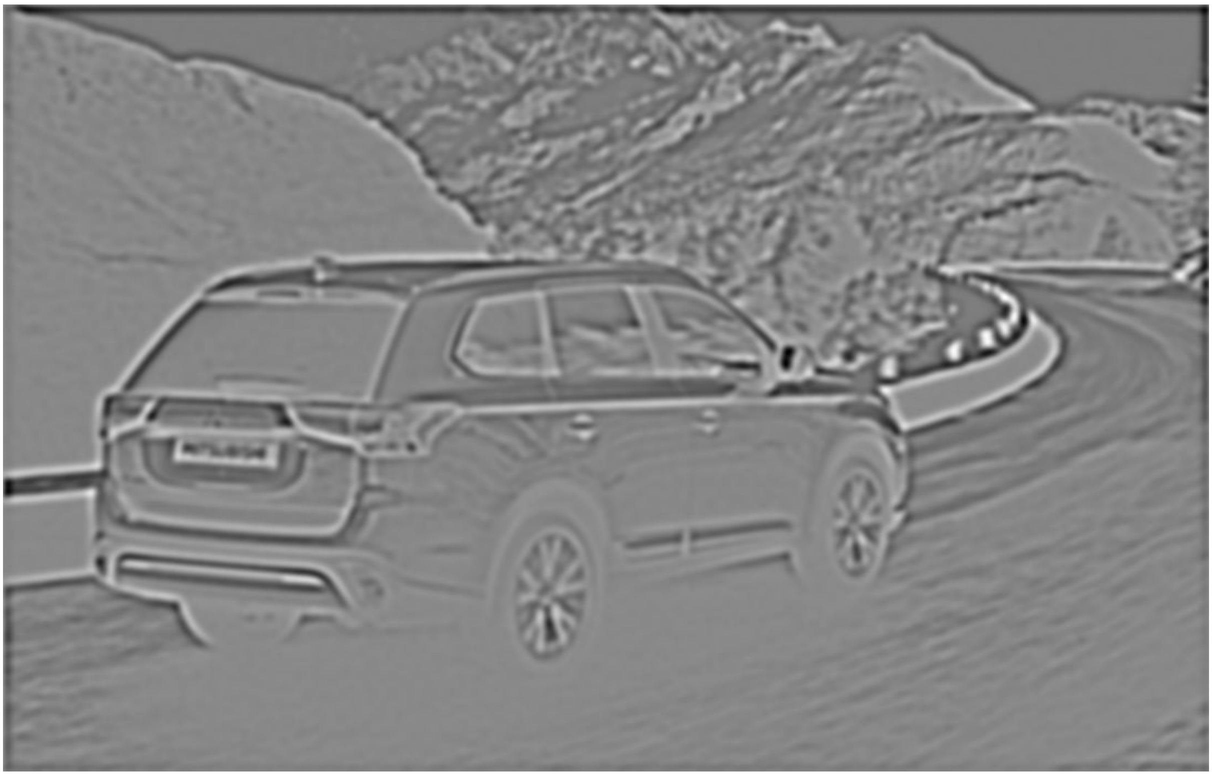
```
import numpy as np
import matplotlib.pyplot as plt
import cv2
import math
import pandas as pd
import copy

def get_kernel(sigma):
    size = int(2*(np.ceil(3*sigma))+1)
    x, y = np.meshgrid(np.arange(-size/2+1, size/2+1),
                       np.arange(-size/2+1, size/2+1))
    normal = 1 / (2.0 * np.pi * sigma**2)
    kernel = ((x**2 + y**2 - (2.0*sigma**2)) / sigma**4) * \
        np.exp(-(x**2+y**2) / (2.0*sigma**2)) / normal

    return kernel

def convolution(img, ker, padding=2):
    m, n = ker.shape
    y, x = img.shape
    # padding
    pad_img = np.zeros((y+padding*2, x+padding*2))
    pad_img[padding:-padding, padding:-padding] = img

    new_img = np.zeros((y, x))
    for i in range(y):
        for j in range(x):
            new_img[i][j] = np.sum(pad_img[i:i+m, j:j+n] * ker)
    return new_img

def zero_crossing(img, padding=1, thres_hold=0):
    y, x = img.shape
    # padding
    pad_img = np.zeros((y+padding*2, x+padding*2))
    pad_img[padding:-padding, padding:-padding] = img
    thres = thres_hold*(np.max(pad_img)-np.min(pad_img))

    new_img = np.zeros((y, x))
    for i in range(y):
        for j in range(x):

            # left/right
            if pad_img[i-1][j]*pad_img[i+1][j] < 0 and \
                abs(pad_img[i-1][j]-pad_img[i+1][j]) > thres:
                new_img[i][j] = 1
                continue

            # up/down
            if pad_img[i][j-1]*pad_img[i][j+1] < 0 and \
                abs(pad_img[i][j-1]-pad_img[i][j+1]) > thres:
                new_img[i][j] = 1
                continue

            # left diagonals
            if pad_img[i-1][j-1]*pad_img[i+1][j+1] < 0 and \
                abs(pad_img[i-1][j-1]-pad_img[i+1][j+1]) > thres:
                new_img[i][j] = 1
                continue

            # right diagonals
            if pad_img[i-1][j+1]*pad_img[i-1][j+1] < 0 and \
                abs(pad_img[i-1][j+1]-pad_img[i-1][j+1]) > thres:
                new_img[i][j] = 1
                continue

    return new_img

def hough_line(edge):
    theta = np.arange(0, 180, 1)
    cos = np.cos(np.deg2rad(theta))
    sin = np.sin(np.deg2rad(theta))
    rho_range = round(math.sqrt(edge.shape[0]**2 + edge.shape[1]**2))
    accumulator = np.zeros((2 * rho_range, len(theta)),  dtype=np.uint8)

    edge_pixels = np.where(edge == 1)
    coordinates = list(zip(edge_pixels[0], edge_pixels[1]))

    for p in range(len(coordinates)):
        for t in range(len(theta)):
            rho = int(round(coordinates[p][1] * cos[t] +
coordinates[p][0] * sin[t]))
            accumulator[rho, t] += 1

    return accumulator

def parameter_space_to_image(pr_space):
    param_img = (pr_space.astype(np.float)/np.max(pr_space))*255
    tmp1, tmp2 = param_img[:param_img.shape[0]//2],
param_img[param_img.shape[0]//2:]
    param_img = np.vstack((tmp2,tmp1))
    param_img = np.fliplr(param_img)
    param_img = cv2.resize(param_img,(1200,1200))
    return param_img

# read image
img = plt.imread('Car On Mountain Road.tif')
img = np.float32(img)/255.0

log_kernel = get_kernel(sigma=4)
LoG = convolution(img, log_kernel, padding=log_kernel.shape[0]//2)
LoG_norm = (LoG-np.min(LoG))/(np.max(LoG)-np.min(LoG))
cv2.imwrite('LoG.png', (LoG_norm*255))
log_0_percent = zero_crossing(LoG, thres_hold=0)
cv2.imwrite('log_0_percent.png', (log_0_percent*255))
log_4_percent = zero_crossing(LoG, thres_hold=0.04)
cv2.imwrite('log_4_percent.png', (log_4_percent*255))

parameter_space = hough_line(log_4_percent)
param_img = parameter_space_to_image(parameter_space)
cv2.imwrite('parameter_space.png', param_img)
```

```
new_img = copy.deepcopy(img)
mask = np.zeros(new_img.shape)
new_img = cv2.cvtColor(new_img, cv2.COLOR_GRAY2RGB)
color_param_img = cv2.cvtColor(param_img.astype(np.float32),
cv2.COLOR_GRAY2RGB)
blank_img = np.ones(new_img.shape)
# license plate cordinate
coordinates = [
    (157, 395),
    (155, 415),
    (250, 400),
    (248, 420),
]
for p in coordinates:
    mask[p[1], p[0]] = 1
mask_parameter_sapce = hough_line(mask)
mask_param_image = parameter_space_to_image(mask_parameter_sapce)
edge_pixels = np.where(mask_parameter_sapce >=2)
position = list(zip(edge_pixels[0], edge_pixels[1]))
for i in range(0, len(position)):
    a = np.cos(np.deg2rad(position[i][1]))
    b = np.sin(np.deg2rad(position[i][1]))
    x0 = a*position[i][0]
    y0 = b*position[i][0]
    x1 = int(x0 + 1000*(-b))
    y1 = int(y0 + 1000*(a))
    x2 = int(x0 - 1000*(-b))
    y2 = int(y0 - 1000*(a))

    start_point = (position[i][1]-5, position[i][0]-5)
    end_point = (position[i][1]+5, position[i][0]+5)
    cv2.rectangle(color_param_img, start_point, end_point, (1,0,0), 2)
    cv2.line(new_img,(x1,y1),(x2,y2),(1,0,0),1)
    cv2.line(blank_img,(x1,y1),(x2,y2),(1,0,0),1)

cv2.imwrite('new_img.png', (new_img*255))
cv2.imwrite('blank_img.png', (blank_img*255))
cv2.imwrite('parameter_space.png', color_param_img)
```

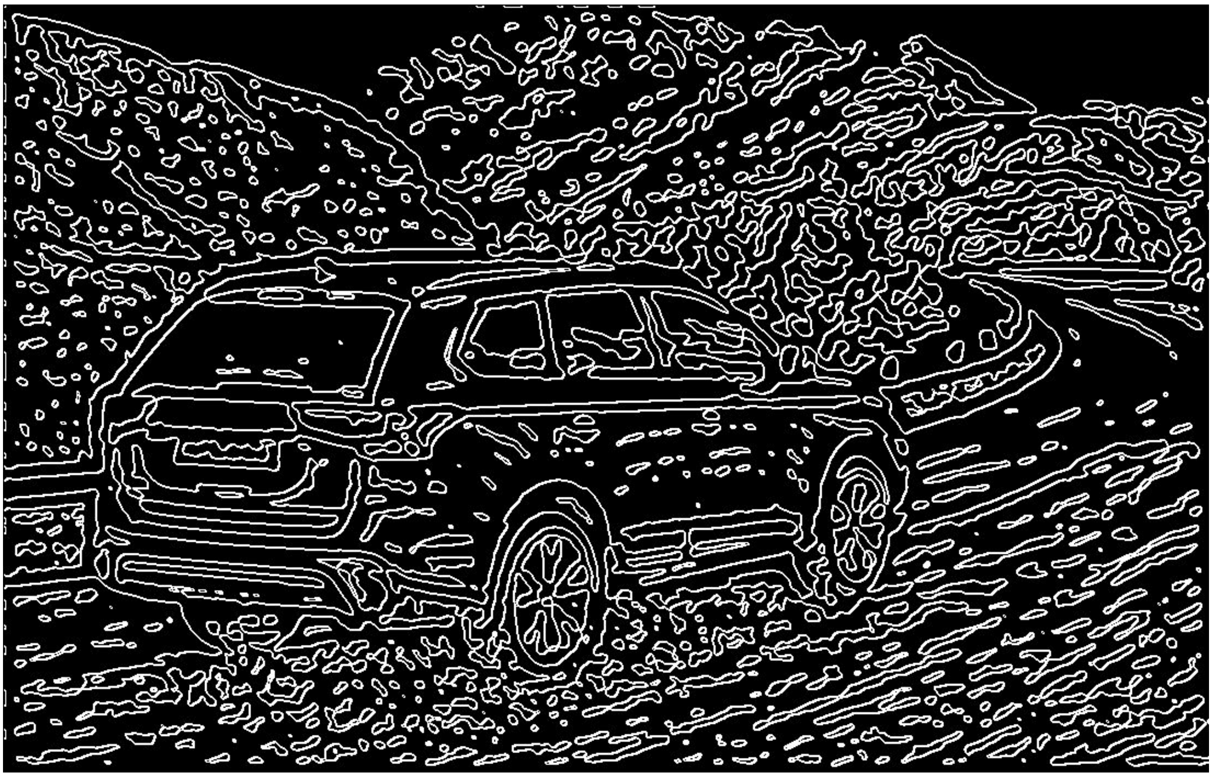
黃瑞得
0860078
huangjuite@gmail.com

Figure of LoG

original

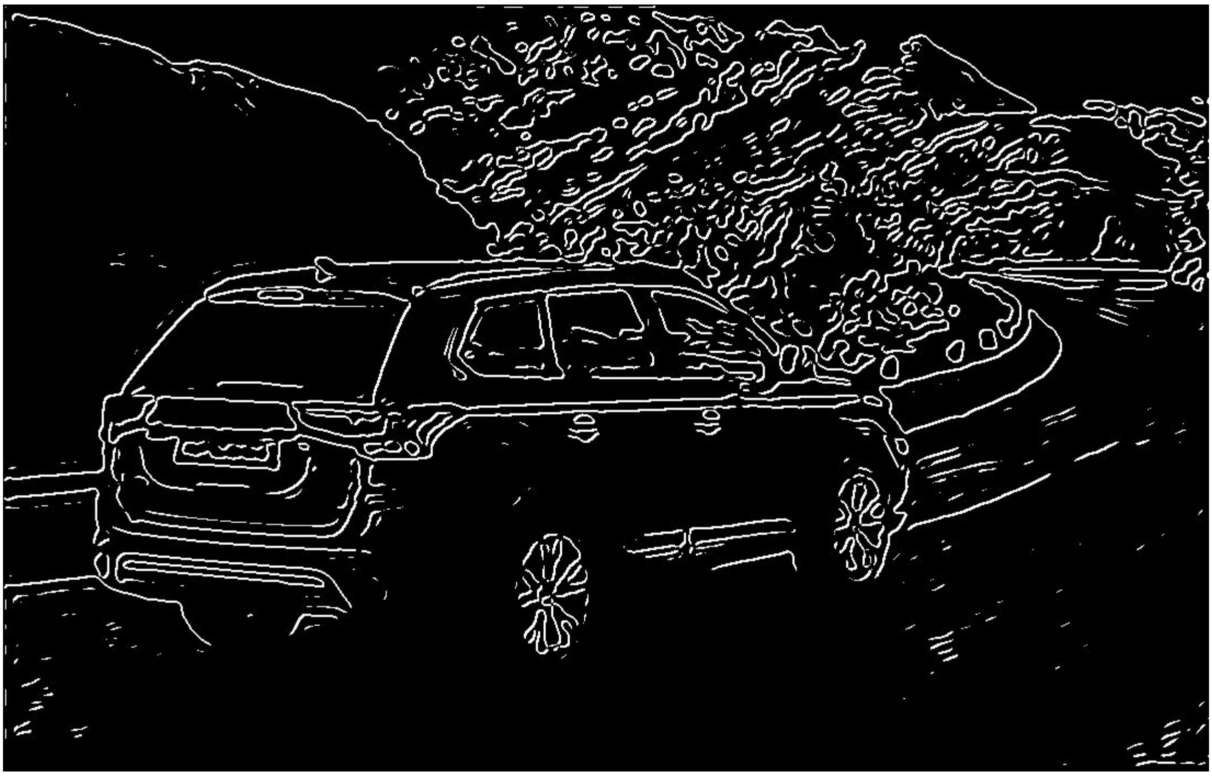


0 percent zero crossing



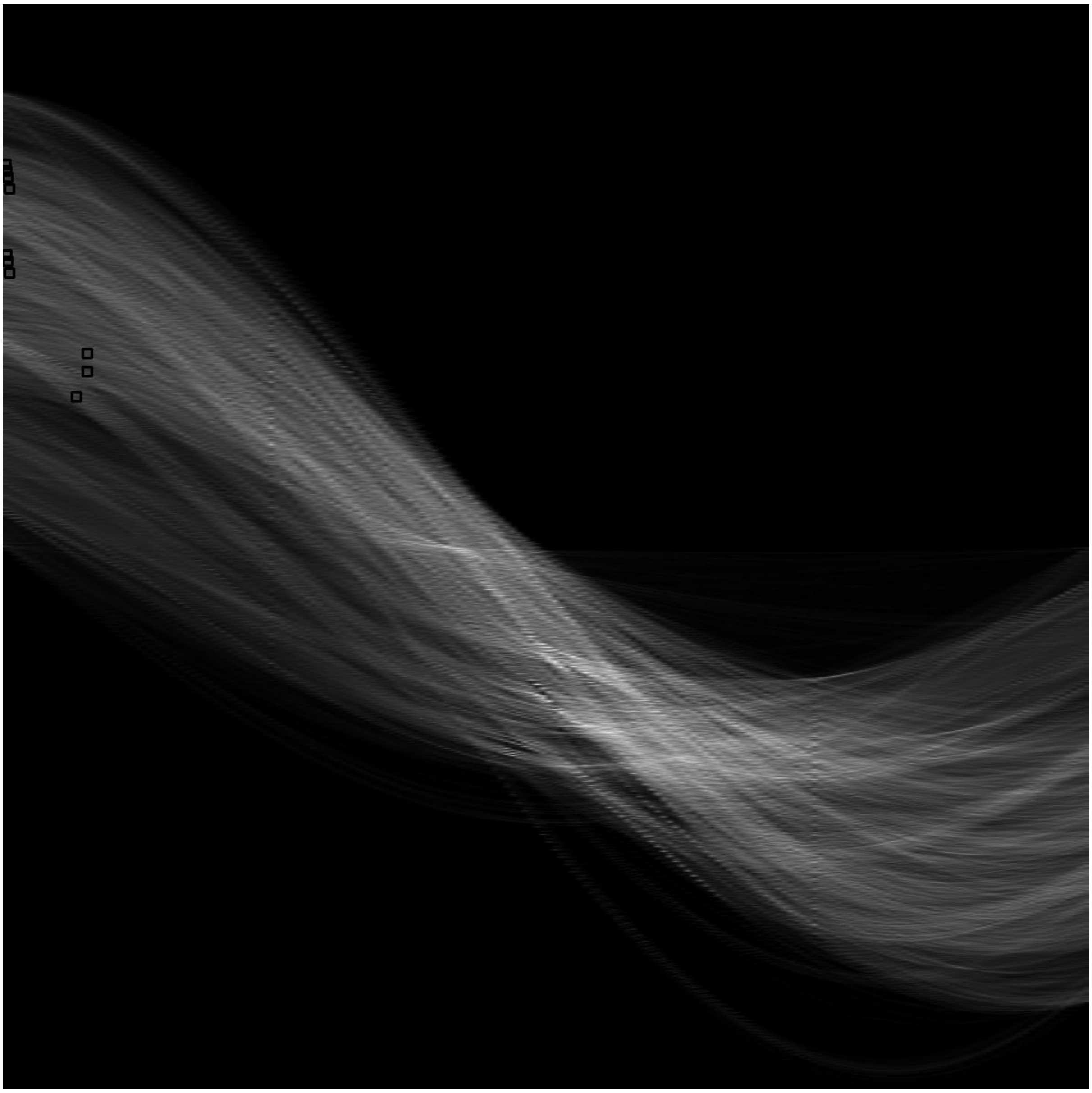
黃瑞得
0860078
huangjuite@gmail.com

4 percent zero corssing



黃瑞得
0860078
huangjuite@gmail.com

Hough parameter space



黃瑞得
0860078
huangjuite@gmail.com

license plate