차량 번호판 실습 과제

전진희

1. 차량 이미지 읽기

```
# 필요한 라이브러리 임포트
import cv2
import matplotlib.pyplot as plt
import numpy as np
import pytesseract # 이미지에서 글자 찾아내는 라이브러리


# 이미지 불러오기
image_path = "./data/car_plate.png"

image = cv2.imread(image_path)
"""
[] : 경로 오류 !!
"""
print(image)

# 색상의 문제 발생 BGR -> RGB 컨버터 필요
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

h, w, c = image.shape
print('이미지 크기 >> ', h, w, c)

plt.imshow(image, 'gray')
plt.show()
```

-pytesseract 라이브러리 없으면 설치할 것 : !pip install pytesseract

pytesseract은 Python에서 Tesseract OCR 엔진에 액세스하기 위한 인터페이스입니다. Tesseract OCR은 Google에서 개발한 오픈 소스 OCR(광학 문자 인식) 엔진으로, 이미지에서 텍스트를 추출하는 데 사용됩니다.

[39 20 12]]
이미지 크기 >> 720 960 3

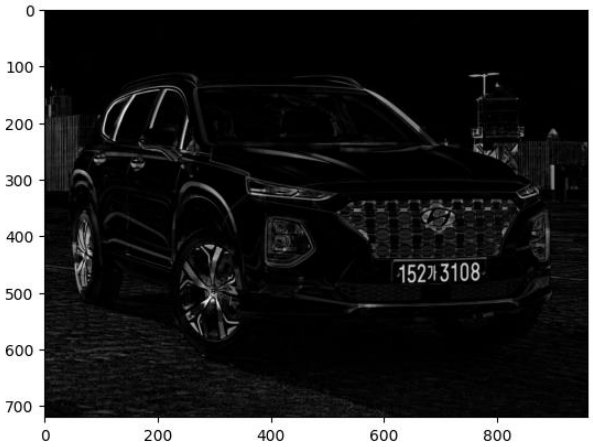
2. 차량 이미지 Grayscale 변환

```
gray_image = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)


plt.imshow(gray_image, 'gray')
 # 그레이 이미지는 그레이임을 명시해주어야 한다
plt.show()
```

3. 차량 이미지 – Maximize contrast

```
# 모폴로지 연산
structuringElement = cv2.getStructuringElement(cv2.MORPH_RECT ,(9,9)) #커널 값 , 특징점 강조시 사용하는 함수
print(structuringElement)

# TOPHAT -> 밝기 값이 크게 변화하는 영역을 강조
imgToHat = cv2.morphologyEx(gray_image, cv2.MORPH_TOPHAT, structuringElement)
# kernel 값으로 structuringElement 넣어줌

# BLACKHAT -> 어두운 부분을 강조
imgBlackHat = cv2.morphologyEx(gray_image, cv2.MORPH_BLACKHAT, structuringElement)

plt.imshow(imgToHat, 'gray')
plt.show()

# 합치기 – 밝은 부분
imgGrayscalePlusToHat = cv2.add(gray_image, imgToHat)

plt.imshow(imgGrayscalePlusToHat, 'gray')
plt.show()

# 뺄셈 연산
gray = cv2.subtract(imgGrayscalePlusToHat, imgBlackHat)
# 0보다 작은 값에 대해선 모두 0이 된다
plt.imshow(gray, 'gray')
plt.show()
```

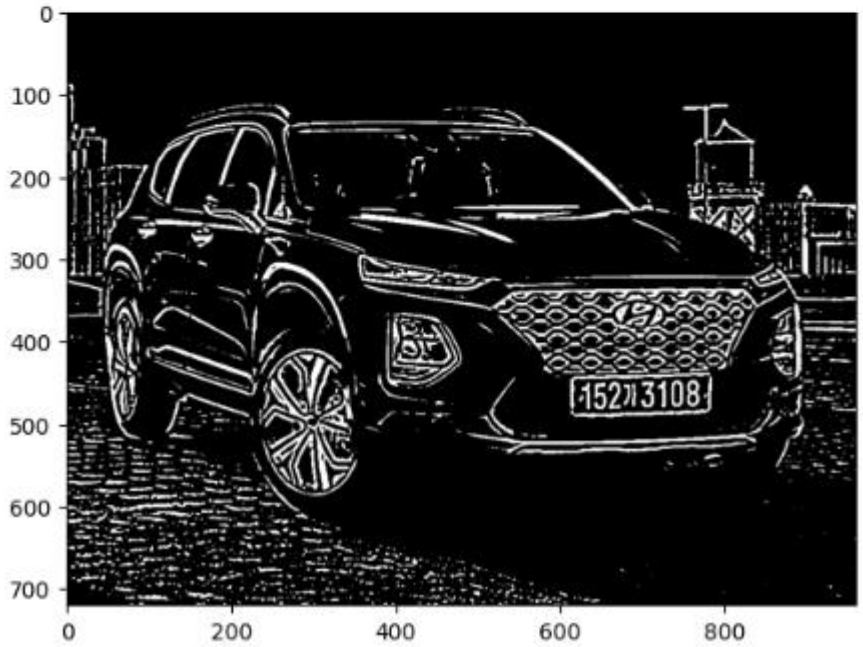

0
100
200
300
400
500
600
700
0
200
400
600
800


0
100
200
300
400
500
600
700
0
200
400
600
800
152 3108


0
100
200
300
400
500
600
700
0
200
400
600
800
152 3108

4. 차량 이미지 – 이진화

```
img_blurred = cv2.GaussianBlur(gray, ksize=(7,7), sigmaX=0)
# 여기서 gray는 위에서 선명하게 만들어준 이미지

plt.imshow(img_blurred , 'gray')
plt.show()

ima_thresh = cv2.adaptiveThreshold(
# adaptiveThreshold는 주변 평균값 사용하는 임계값 설정
                img_blurred,
                maxValue=255,
                adaptiveMethod=cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                thresholdType=cv2.THRESH_BINARY_INV,
                blockSize=11,
                C=9
)

plt.imshow(ima_thresh, 'gray')
plt.show()
```

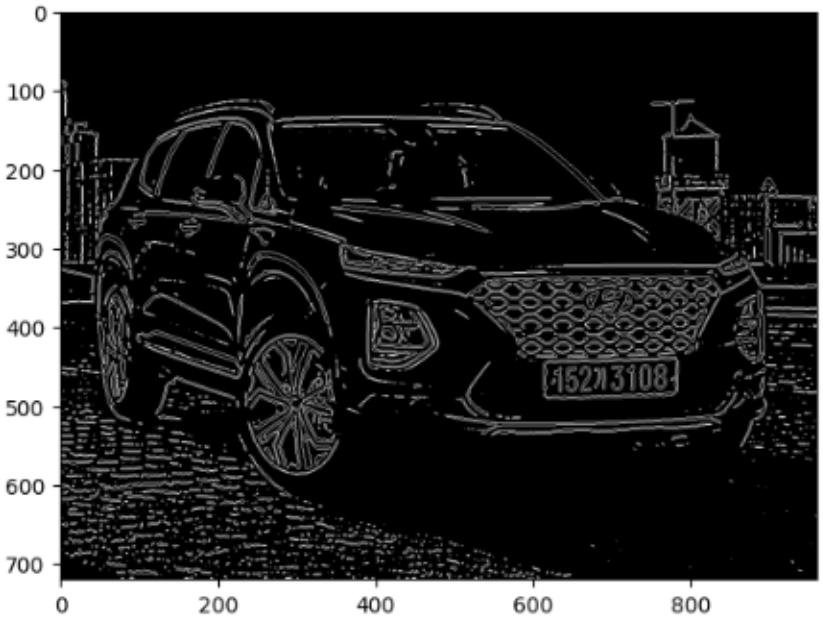


5. 차량 이미지 – Find Contour(이미지에서 객체의 외각선 검출)

```
contours, hierarchy = cv2.findContours(
        img_thresh,
        mode = cv2.RETR_LIST,
        method = cv2.CHAIN_APPROX_SIMPLE
)

# print(contours) # contour 포인트 좌표
```

```
temp_result = np.zeros((h_, w_, c), dtype = np.uint8)

cv2.drawContours(temp_result, contours=contours, contourIdx=-1,
color=(255,255,255))
plt.imshow(temp_result, 'gray')
plt.show()
```

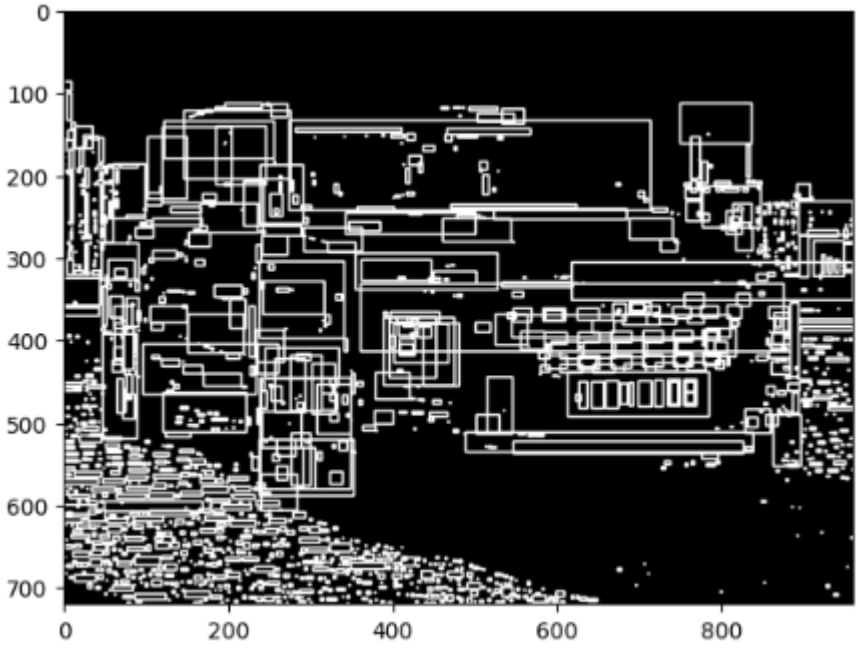
Canny은 주로 에지 검출, indContours 함수는 이미지에서 윤곽선(contour)을 검색하는 함수



6. 차량 이미지 전처리 (바운딩 박스 표시)

```
temp_result = np.zeros((h_, w_, c), dtype=np.uint8)

contours_dict = []

for contour in contours:
    x, y, w, h = cv2.boundingRect(contour)
    cv2.rectangle(temp_result, (x,y), (x+w, y+h), (255,255,255), thickness=2)

    # Insert to dict
    contours_dict.append({
        'contour' : contour,
        'x':x,
        'y':y,
```

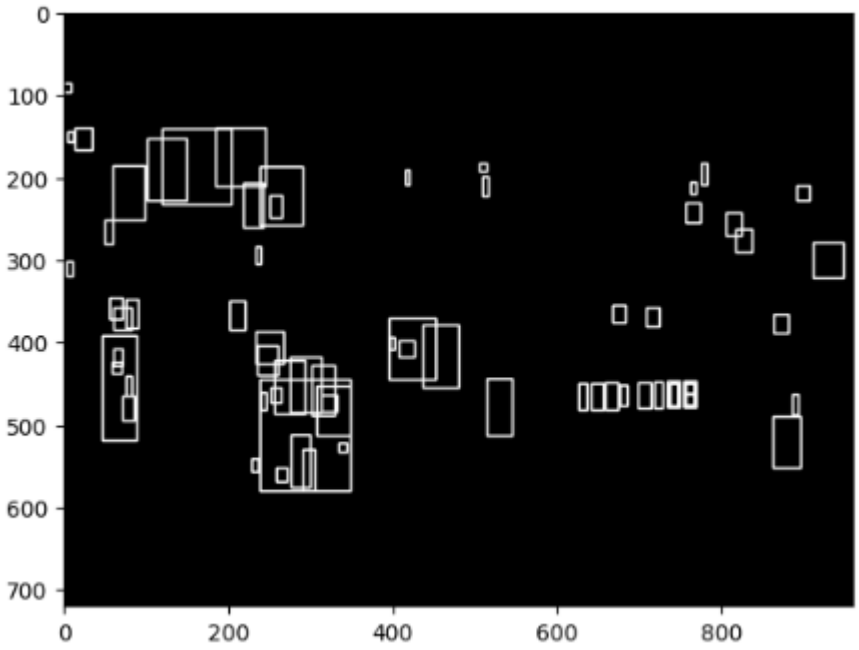
```
                    "w":w,
                    'h':h,
                    "cx": x + (w / 2),
                    "cy": y + (y / 2)
                })

        #print(contours_dict)
        plt.imshow(temp_result, 'gray')
        plt.show()

# print(x, y, w, h)
```



7. 차량 이미지 Select Candidates by Char Size 변환

```
        MIN_AREA = 80
        MIN_WIDTH, MIN_HEIGHT = 2, 8
        MIN_RATIO, MAX_RATIO = 0.25, 1.0

        possible_contours = []

        count = 0
        for d in contours_dict:
            area = d['w'] * d['h']
            ratio = d['w'] / d['h']

            if area > MIN_AREA and d['w'] > MIN_WIDTH and d['h'] > MIN_HEIGHT
```

```
            and MIN_RATIO < ratio < MAX_RATIO:
                d['idx'] = count
                count += 1
                possible_contours.append(d)


        # visualize possible contours
        temp_result = np.zeros((h_, w_, c), dtype=np.uint8)

        for d in possible_contours:
            pt1 = (int(d['x']), int(d['y']))
            pt2 = (int(d['x'] + d['w']), int(d['y'] + d['h']))
            cv2.rectangle(temp_result, pt1=pt1, pt2=pt2, color=(255, 255, 255),
        thickness=2)

        plt.imshow(temp_result.astype(np.uint8), 'gray')
        plt.show()
```



x,y가 int가 되어야 오류없이 박스가 쳐진다 (float 타입일 시 변경)

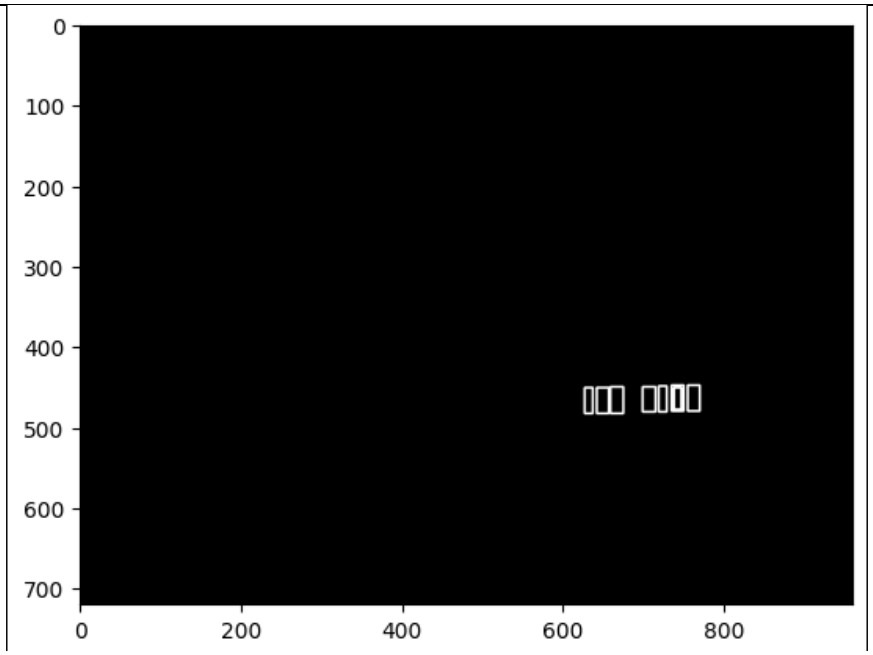
8. 차량 이미지 - Select Candidates by Arrangement of Contours 변환

```
MAX_DIAG_MULTIPLYER = 5    # 두 윤곽선 사이 최대 대각선 길이 제한
MAX_ANGLE_DIFF =12.0       # 두 윤곽선 사이 최대 각도 차이 제한
```

```
MAX_AREA_DIFF = 0.5      # 두 윤곽선 사이 최대 면적 차이 제한
MAX_HEIGHT_DIFF = 0.2    # 두 윤곽선 사이 최대 높이 차이 제한
MIN_N_MATCHED = 5.5  # 두 윤곽선 사이 매칭되는 최대 윤곽선 개수 차이 제한
MAX_WIDTH_DIFF = 0.8

def find_chars(contour_list) :
    print(contour_list)

    matched_result_idx = []
    for d1 in contour_list :
        matched_contour_idx = []
        for d2 in contour_list :
            print(d1['idx'], d2['idx'])
            if d1['idx'] == d2['idx'] :
                continue

            dx = abs(d1['cx'] - d2['cx']) # x축 거리
            dy = abs(d1['cy'] - d2['cy']) # y축 거리

            diagonal_length1 = np.sqrt(d1['w'] **2 + d2['h'] **2) # 대각선 길이

            distance = np.linalg.norm(np.array([d1['cx'], d1['cy']]) -
np.array([d2['cx'], d2['cy']])) # 거리
            # print(diagonal_length1)
            # print(distance)

            if dx == 0 :
                angle_diff = 90
            else :
                angle_diff = np.degrees(np.arctan(dy / dx)) # 각도 차이

            area_diff = abs(d1['w'] * d1['h'] - d2['w'] * d2['h']) / (d1['w'] * d1['h'])
# 면적 차이 계산
            width_diff = abs(d1['w'] - d2['w']) / d1['w']
#  d1 대비 비율로 나타내어 정규화(for 일관성)
            height_diff = abs(d1['h'] - d2['h']) / d1['h']

            if distance < diagonal_length1 * MAX_DIAG_MULTIPLYER and
angle_diff < MAX_ANGLE_DIFF ₩
```

```
                and area_diff < MAX_AREA_DIFF and width_diff < MAX_WIDTH_DIFF
and height_diff < MAX_HEIGHT_DIFF :
                    matched_contour_idx.append(d2['idx'])



        matched_contour_idx.append(d1['idx'])

        #print(len(matched_contour_idx))

        if len(matched_contour_idx) < MIN_N_MATCHED:
# array 안에 포함된 것 3개 이하인 것들 버리기
            continue

        matched_result_idx.append(matched_contour_idx)

        unmatched_contour_idx = []

        for d4 in contour_list :
            if d4['idx'] not in matched_contour_idx :
                unmatched_contour_idx.append(d4['idx'])
# 매칭되지 않는 개체의 인스턴스 추출하여  리스트에 추가

        unmatched_contour = np.take(possible_contours, unmatched_contour_idx)
        print(unmatched_contour)

        # recursive
        recursive_contour_list =   find_chars(unmatched_contour)

        for idx in recursive_contour_list :
            matched_result_idx.append(idx)

        break

    return matched_result_idx

        ##### 진행 #####


result_idx = find_chars(possible_contours)
```

```
matched_result = []
for idx_list in result_idx :
    matched_result.append(np.take(possible_contours, idx_list))

temp_result = np.zeros((h_, w_, c), dtype=np.uint8)

for r in matched_result :
    for d in r :
        x1_temp = d['x']
        y1_temp = d['y']
        w1_temp = d['w']
        h1_temp = d['h']
        print(x1_temp, y1_temp, w1_temp, h1_temp)
        cv2.rectangle(temp_result, (x1_temp, y1_temp), (x1_temp+w1_temp,
y1_temp+h1_temp), (255,255,255), thickness=2)

plt.imshow(temp_result, 'gray')
plt.show()
```

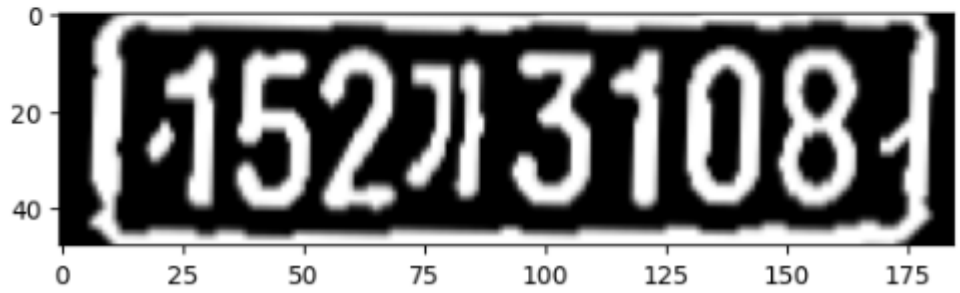


9. 차량 번호판 이미지를 회전시키는 작업

```
PLATE_WIDTH_PADDING = 1.3  # 번호판 주변 너비의 여유공간
PLATE_HEIGHT_PADDING = 1.5 # 번호판 주변 너비 여유공간
```

```
MIN_PLATE_RATIO = 3 # 가로 세로 비율 최소값
MAX_PLATE_RATIO =10 # 가로 세로 비율 최대값

plate_img = []
plate_infos = []

""" lambda 매개변수 : 표현식 """

for i, matched_chars in enumerate(matched_result) :
    sorted_chars = sorted(matched_chars, key=lambda x : x['cx'])
    print(sorted_chars)
    print(sorted_chars[-1]['cx'])
    print(sorted_chars[0]['cx'])

    plate_cx = (sorted_chars[0]['cx'] + sorted_chars[-1]['cx']) / 2    # 평균값
    plate_cy = (sorted_chars[0]['cy'] + sorted_chars[-1]['cy']) / 2

    plate_width = (sorted_chars[-1]['x'] + sorted_chars[-1]['w'] - sorted_chars[0]['x'])
* PLATE_WIDTH_PADDING

    sum_height = 0
    for d in sorted_chars :
        sum_height += d['h']

    plate_height = int(sum_height / len(sorted_chars) * PLATE_HEIGHT_PADDING) #
번호판 높이 결정

    triangle_hypotenus = np.linalg.norm(
        np.array([sorted_chars[0]['cx'], sorted_chars[0]['cy']]) -
np.array([sorted_chars[-1]['cx'], sorted_chars[-1]['cy']])
    )

plt.imshow(img_cropped, 'gray')
plt.show()
```

```
[array([[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]], dtype=uint8)]
```



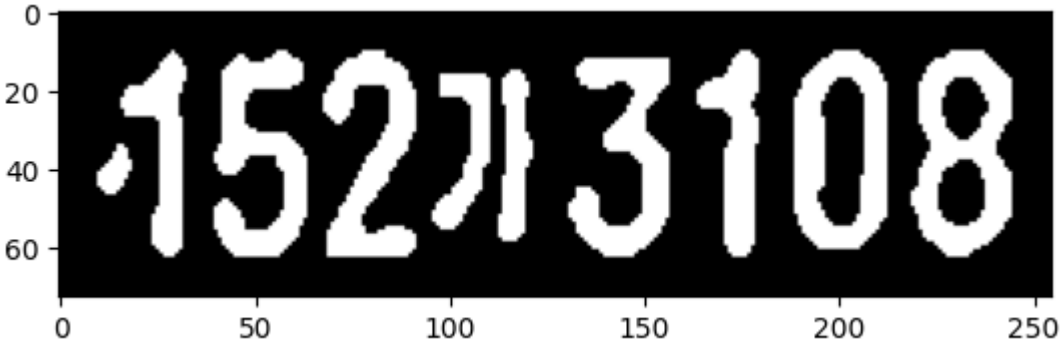
10. 차량 번호판 Another Thresholding to Find Chars

Another Thresholding to Find Chars : 문자를 찾기 위해 추가적인 이진화 작업

```
longest_idx, longest_text = -1, 0

plate_chars = []



for i, plate_imgs in enumerate(plate_img):

    plate_imgs = cv2.resize(plate_imgs, dsize=(0, 0), fx=1.6, fy=1.6)

    _, plate_imgs = cv2.threshold(plate_imgs, thresh=0.0, maxval=255.0,
type=cv2.THRESH_BINARY | cv2.THRESH_OTSU)

# (0.0), 즉 크기 조정 안하고 px, py 조정을 통해 비율 조정만하겠다

# Tresh 0 으로 주면 최적값 알아서 들어간다



    contours, hierarchy = cv2.findContours(plate_imgs, mode=cv2.RETR_LIST,
method=cv2.CHAIN_APPROX_SIMPLE)



    plate_min_x, plate_min_y = plate_imgs[1], plate_imgs[0]
```

```
    plate_max_x, plate_max_y = 0, 0



    for contour in contours:

        contour_x, contour_y, contour_w, contour_h = cv2.boundingRect(contour)



        area_temp = contour_w * contour_h

        ratio_temp = contour_w / contour_h



        if area_temp > MIN_AREA and contour_w > MIN_WIDTH and   contour_h >
MIN_HEIGHT and MIN_RATIO < ratio_temp < MAX_RATIO:




            if contour_x < plate_min_x:

                plate_min_x = contour_x

            if contour_y < plate_min_y:

                plate_min_y = contour_y

            if contour_x + contour_w > plate_max_x:

                plate_max_x = contour_x + contour_w

            if contour_y + contour_h > plate_max_y:

                plate_max_y = contour_y + contour_h



    img_result = plate_img[plate_min_y : plate_max_y, plate_min_x : plate_max_x]
```

```
    img_result = cv2.GaussianBlur(img_result, ksize=(3,3), sigmaX=0)

    _, img_result = cv2.threshold(img_result, thresh=0.0, maxval = 255.0,
type=cv2.THRESH_BINARY | cv2.THRESH_OTSU)

    img_result = cv2.copyMakeBorder(img_result, top = 10, bottom =10, left =10, right =10,

                                    borderType=cv2.BORDER_CONSTANT, value=(0,0,0))



    plt.imshow(img_result, 'gray')

    plt.show

    #



text = pytesseract.image_to_string(img_result)

print(text)
```



11. 차량 번호판 바운딩 박스 표시 결과 이미지

```
info = plate_infos[longest_idx]
img_out = image.copy()

print(info)

cv2.rectangle(img_out, (info['x'], info['y']), (info['x'] + info['w'], info['y'] + info['h']),
(255, 255, 0), thickness=2)

plt.imshow(img_out)
plt.show()
```