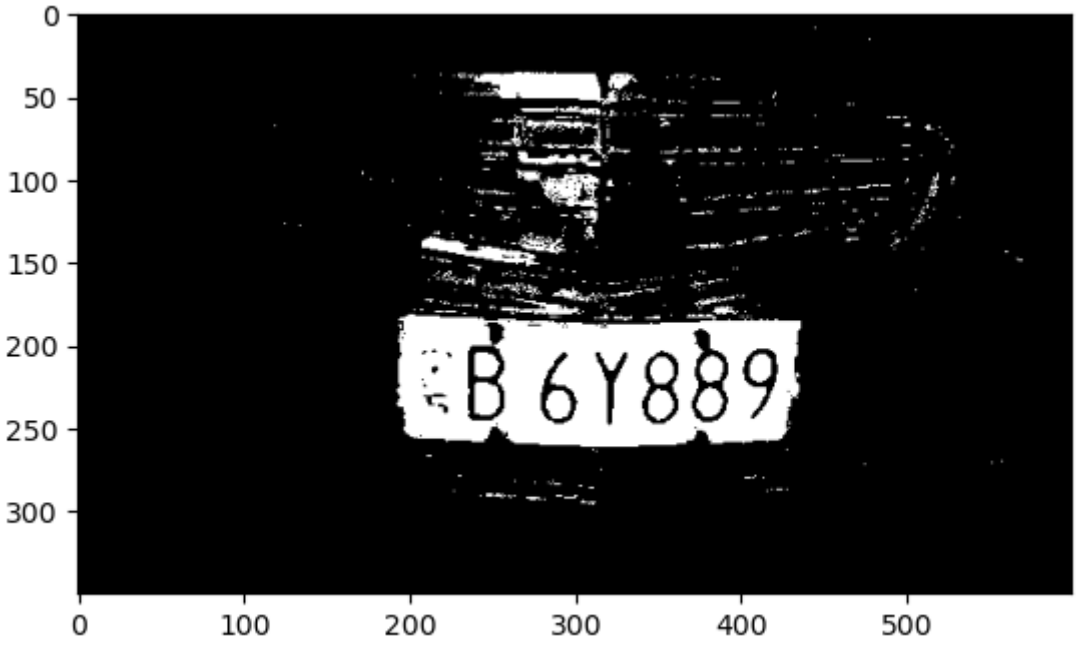
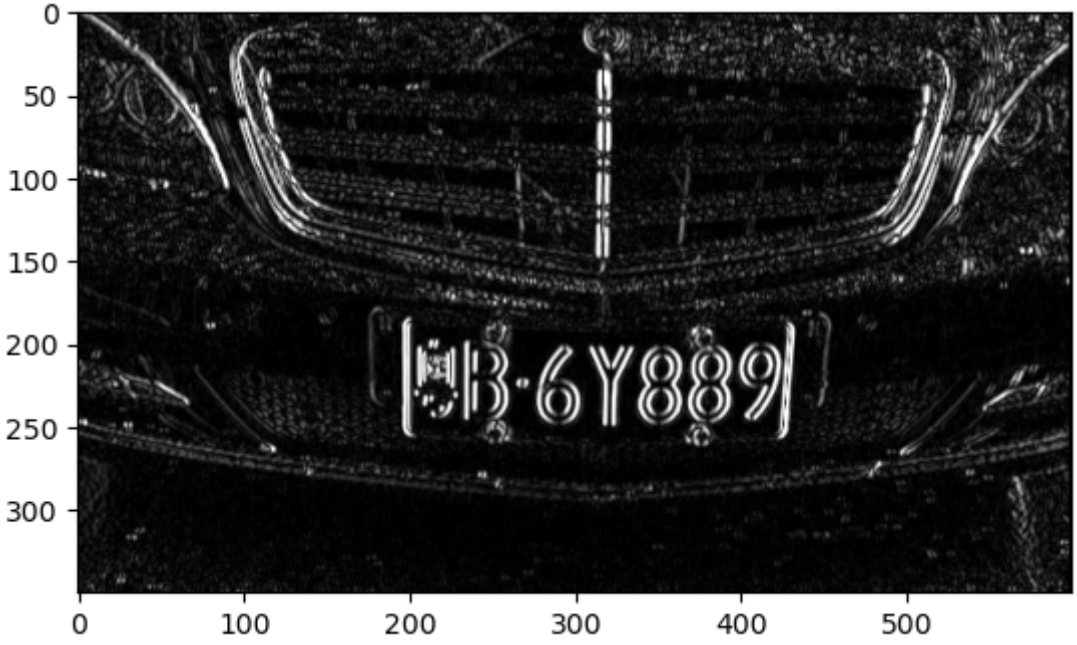
In [48]:
```
import cv2
from matplotlib import pyplot as plt
import os
import numpy as np

# 显示图片
def cv_show(name,img):
    cv2.imshow(name,img)
    cv2.waitKey()
    cv2.destroyAllWindows()

# plt显示彩色图片
def plt_show0(img):
    b,g,r = cv2.split(img)
    img = cv2.merge([r, g, b])
    plt.imshow(img)
    plt.show()

# plt显示灰度图片
def plt_show(img):
    plt.imshow(img,cmap='gray')
    plt.show()

# 图像去噪灰度处理
def gray_guss(image):
    image = cv2.GaussianBlur(image, (3, 3), 0)
    gray_image = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
    return gray_image
```

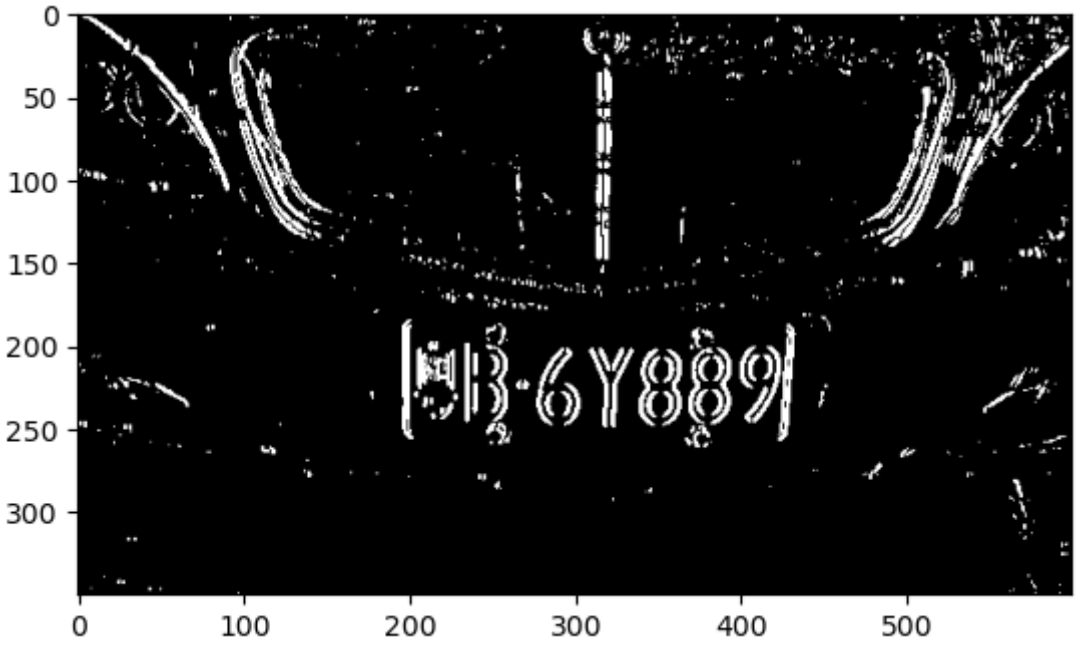
In [104…
```
# 读取待检测图片
origin_image = cv2.imread('License Plate Recognition/car4.jpg')
plt_show0(origin_image)

# 转换为HSV颜色空间
hsv = cv2.cvtColor(origin_image.copy(), cv2.COLOR_BGR2HSV)

# 定义蓝色的HSV范围
lower_blue = np.array([100, 43, 46])
upper_blue = np.array([124, 255, 255])

# 创建一个二值化图像，只包含蓝色区域
blue_mask = cv2.inRange(hsv, lower_blue, upper_blue)

# 显示蓝色掩膜
plt_show(blue_mask)

# 提取车牌部分图片
image = origin_image.copy()
# 图像去噪灰度处理
gray_image = gray_guss(image)
# 显示灰度图像
plt_show(gray_image)

# x方向上的边缘检测
Sobel_x = cv2.Sobel(gray_image,cv2.CV_16S,1,0)
absX = cv2.convertScaleAbs(Sobel_x)
image = absX
# 显示灰度图像
plt_show(image)

# 图像阈值化操作，获得二值化图
ret,image = cv2.threshold(image,0,255,cv2.THRESH_OTSU)
```

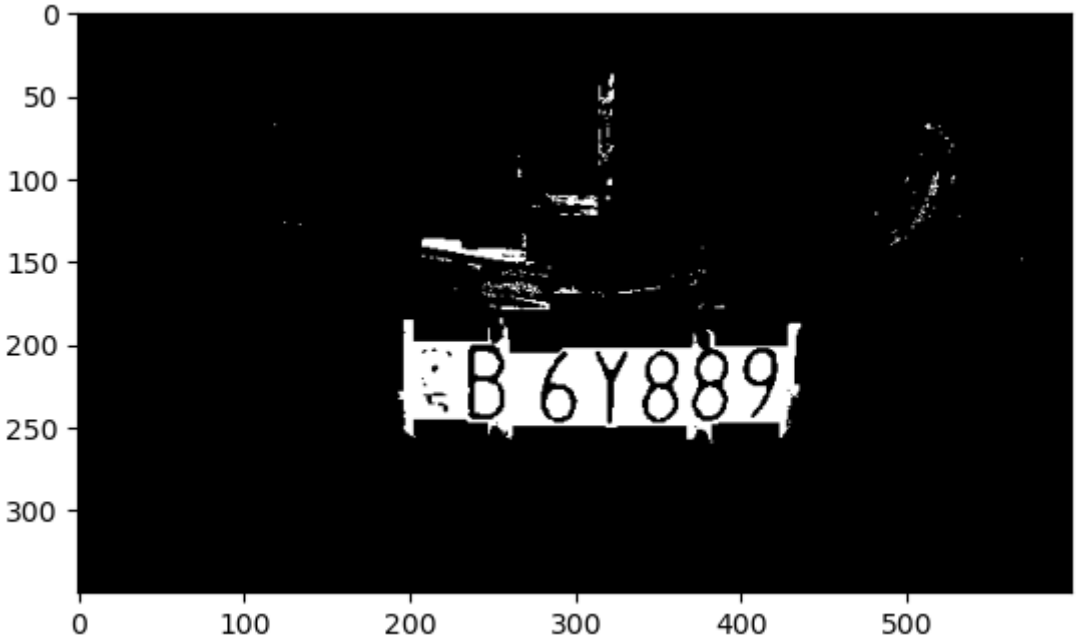
```
# 显示灰度图像
plt_show(image)
# 形态学，闭操作
kernelX = cv2.getStructuringElement(cv2.MORPH_RECT,(30,10))
image = cv2.morphologyEx(image,cv2.MORPH_CLOSE,kernelX,iterations = 1)
# 显示灰度图像
image = cv2.bitwise_and(image, image, mask=blue_mask)
plt_show(image)
```




0
50
100
150
200
250
300
0
100
200
300
400
500

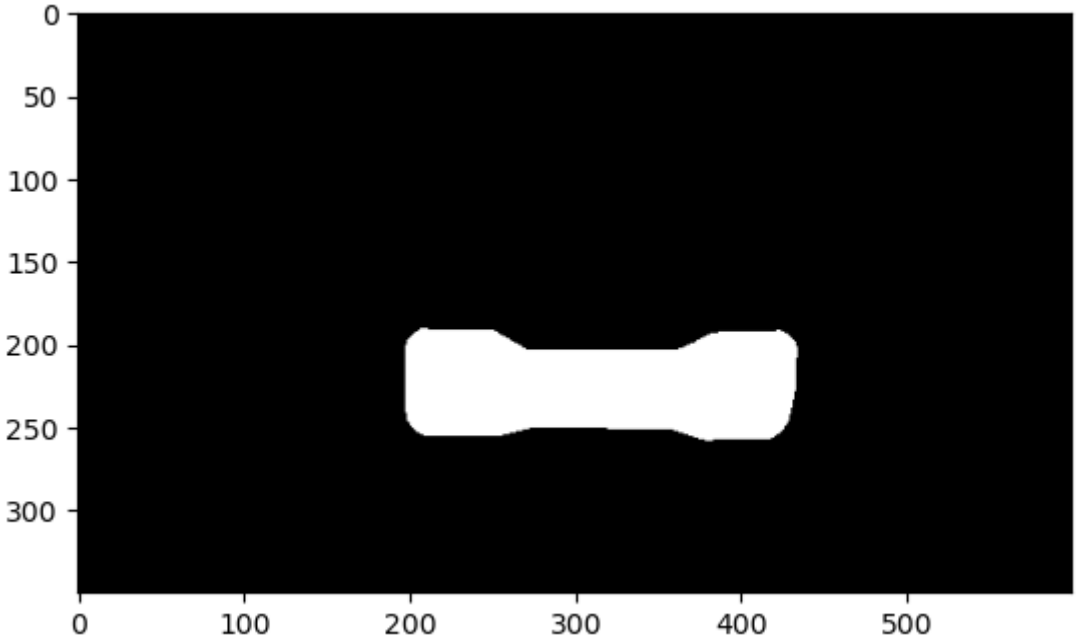

0
50
100
150
200
250
300
0
100
200
300
400
500


0
50
100
150
200
250
300
0
100
200
300
400
500



In [105…

```
# 腐蚀（erode）和膨胀（dilate）
kernelX = cv2.getStructuringElement(cv2.MORPH_RECT, (50, 1))
kernelY = cv2.getStructuringElement(cv2.MORPH_RECT, (1, 20))
#x方向进行闭操作（抑制暗细节）
image = cv2.dilate(image, kernelX)
image = cv2.erode(image, kernelX)
#y方向的开操作
image = cv2.erode(image, kernelY)
image = cv2.dilate(image, kernelY)
# 中值滤波（去噪）
image = cv2.medianBlur(image, 21)
# 显示灰度图像
plt_show(image)
# 获得轮廓
contours, hierarchy = cv2.findContours(image, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SI

plate_contours = []

for item in contours:
    rect = cv2.boundingRect(item)
    x = rect[0]
    y = rect[1]
    weight = rect[2]
    height = rect[3]
    # 根据轮廓的形状特点，确定车牌的轮廓位置并截取图像
    if (weight > (height * 2)) and (weight < (height * 5)):
        plate_contours.append((x, y, x + weight, y + height))
        image = origin_image[y:y + height, x:x + weight]
        plt_show0(image)
```

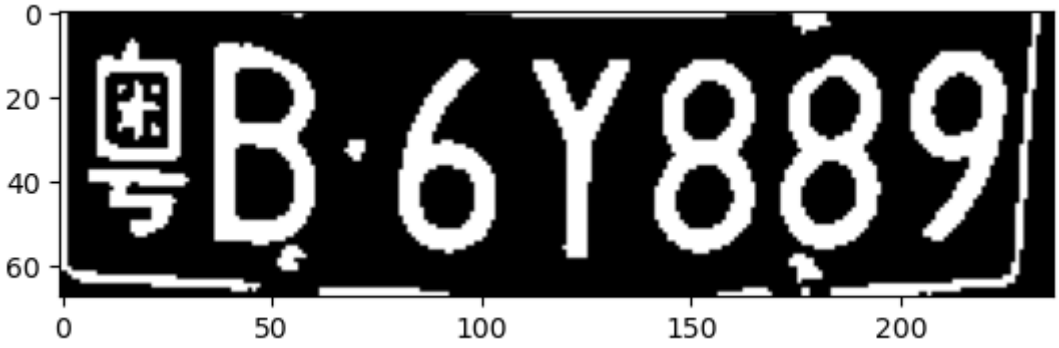


In [106…

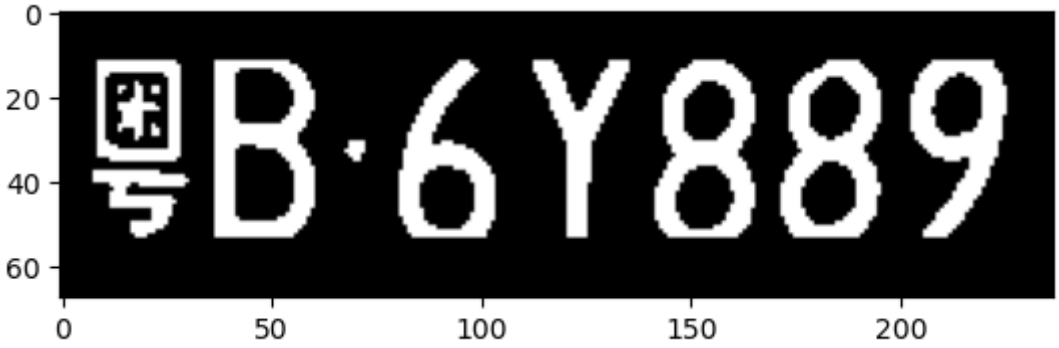
```
# 图像去噪灰度处理
gray_image = gray_guss(image)
# 图像阈值化操作——获得二值化图
ret, image = cv2.threshold(gray_image, 0, 255, cv2.THRESH_OTSU)
plt_show(image)

height, width = image.shape
LICENSE_WIDTH = width
LICENSE_HIGH = height
# 跳变次数去掉铆钉和边框
times_row = []  #存储哪些行符合跳变次数的阈值
for row in range(LICENSE_HIGH): # 按行检测 白字黑底
    pc = 0
    for col in range(LICENSE_WIDTH):
        if col != LICENSE_WIDTH-1:
            if image[row][col+1] != image[row][col]:
                pc = pc + 1
    times_row.append(pc)
#print("每行的跳变次数:",times_row)

#找车牌的下边缘-从下往上扫描
row_end = 0
row_start = 0
for row in range(LICENSE_HIGH-2):
    if times_row[row] < 16:
        continue
    elif times_row[row+1] < 16:
        continue
    elif times_row[row+2] < 16:
        continue
```

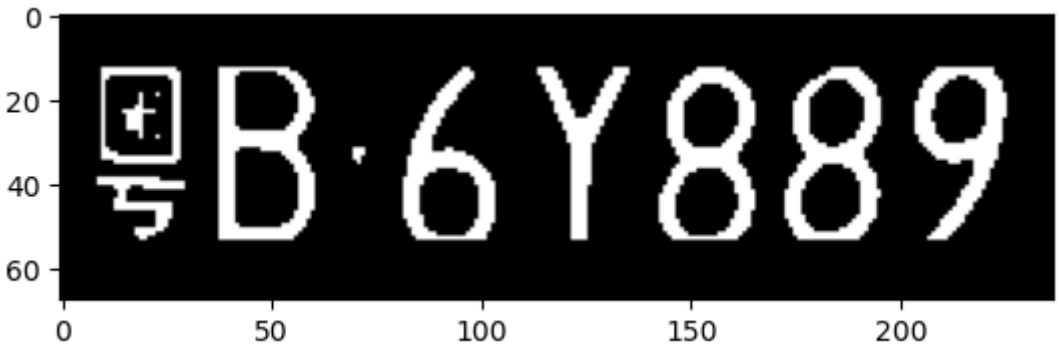
```
    else:
        row_end = row + 2
#print("row_end",row_end)

#找车牌的上边缘-从上往下扫描
i = LICENSE_HIGH-1
row_num = [] #记录row_start可能的位置
while i > 1:
    if times_row[i] < 16:
        i = i - 1
        continue
    elif times_row[i-1] < 16:
        i = i - 1
        continue
    elif times_row[i-2] < 16:
        i = i - 1
        continue
    else:
        row_start = i - 2
        row_num.append(row_start)
        i = i - 1
#print("row_num",row_num)

#确定row_start最终位置
for i in range(len(row_num)):
    if i != len(row_num)-1:
        if abs(row_num[i] - row_num[i+1])>3:
            row_start = row_num[i]
#print("row_start",row_start)

times_col = [0]
for col in range(LICENSE_WIDTH):
    pc = 0
    for row in range(LICENSE_HIGH):
        if row != LICENSE_HIGH-1:
            if image[row,col] != image[row+1,col]:
                pc = pc + 1
    times_col.append(pc)
#print("每列的跳变次数",times_col)
# 找车牌的左右边缘-从左到右扫描
col_start = 0
col_end = 0
for col in range(len(times_col)):
    if times_col[col] > 2:
        col_end = col
print('col_end',col_end)

j = LICENSE_WIDTH-1
while j >= 0:
    if times_col[j] > 2:
        col_start = j
    j = j-1
#print('col_start',col_start)

# 将车牌非字符区域变成纯黑色
for i in range(LICENSE_HIGH):
    if i > row_end or i < row_start:
        image[i] = 0
for j in range(LICENSE_WIDTH):
    if j < col_start or j > col_end:
        image[:,j] = 0
plt_show(image)
# plate_binary = image.copy()
for i in range(LICENSE_WIDTH-1,LICENSE_WIDTH):
```

```
    image[:,i] = 0

# 字符细化操作
specify = cv2.erode(image,kernel,iterations=1)
plt_show(specify)
plate_specify = specify.copy()
```




```
col_end 226
```

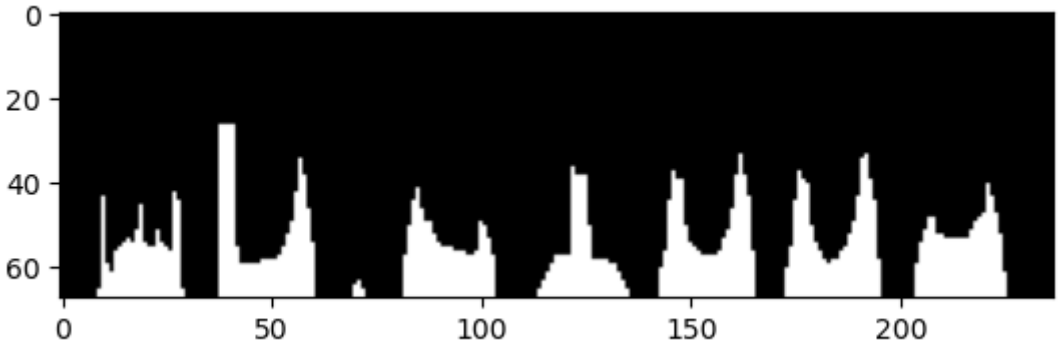




In [107…

```
lst_heise = []  #记录每一列中的白色像素点数量
for i in range(LICENSE_WIDTH):
    pc = 0
    for j in range(LICENSE_HIGH):
        if specify[j][i] == 255:
            pc = pc + 1
    lst_heise.append(pc)
# print("lst_heise",lst_heise)
a = [0 for i in range(0,LICENSE_WIDTH)]
for j in range(0, LICENSE_WIDTH):  # 遍历一列
    for i in range(0, LICENSE_HIGH):  # 遍历一行
        if specify[i, j] == 255:  # 如果该点为白点
            a[j] += 1  # 该列的计数器加一计数
            specify[i, j] = 0  # 记录完后将其变为黑色
    # print (j)
for j in range(0, LICENSE_WIDTH):  # 遍历每一列
    for i in range((LICENSE_HIGH - a[j]), LICENSE_HIGH):  # 从该列应该变白的最顶部的
        specify[i, j] = 255
plt_show(specify)
```

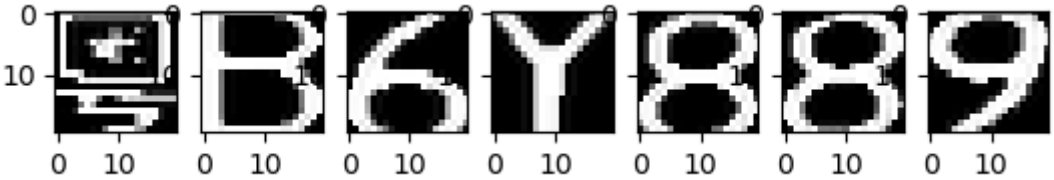
```
#开始找字符的边界
in_block = False #用来指示是否遍历到字符区
startIndex = 0
endIndex = 0
threshold = 10
index = 0
word_images = [] # 存放一个个分割后的字符
for i in range(LICENSE_WIDTH):
    if lst_heise[i] != 0 and in_block == False: # 表示进入有白色像素点的区域
        in_block = True
        startIndex = i
        print("start", startIndex)
    elif lst_heise[i] == 0 and in_block == True: # 表示进入纯黑区域,且纯黑区域前面
        endIndex = i
        in_block = False
        print("end", endIndex)
        if endIndex < startIndex:
            endIndex = LICENSE_WIDTH
        if endIndex - startIndex > 10:
            res = plate_specify[row_start:row_end,startIndex:endIndex]
            index = index + 1
            res = cv2.resize(res, (20, 20), interpolation=cv2.INTER_LANCZOS4)  # 分
            word_images.append(res)
```




```
start 9
end 30
start 38
end 61
start 70
end 73
start 82
end 104
start 114
end 136
start 143
end 166
start 173
end 196
start 204
end 226
```

In [108…

```
for i,j in enumerate(word_images):
    plt.subplot(1,7,i+1)
    plt.imshow(word_images[i],cmap='gray')
plt.show()
```



In [109…

```
#模版匹配
# 准备模板(template[0-9]为数字模板；)
template = ['0','1','2','3','4','5','6','7','8','9',
            'A','B','C','D','E','F','G','H','J','K','L','M','N','P','Q','R','S','T',
            '藏','川','鄂','甘','赣','贵','桂','黑','沪','吉','冀','津','晋','京','辽
            '青','琼','陕','苏','皖','湘','新','渝','豫','粤','云','浙']

# 读取一个文件夹下的所有图片，输入参数是文件名，返回模板文件地址列表
def read_directory(directory_name):
    referImg_list = []
    for filename in os.listdir(directory_name):
        referImg_list.append(directory_name + "/" + filename)
    return referImg_list

# 获得中文模板列表（只匹配车牌的第一个字符）
def get_chinese_words_list():
    chinese_words_list = []
    for i in range(34,64):
        #将模板存放在字典中
        c_word = read_directory('./refer1/'+ template[i])
        chinese_words_list.append(c_word)
    return chinese_words_list
chinese_words_list = get_chinese_words_list()


# 获得英文模板列表（只匹配车牌的第二个字符）
def get_eng_words_list():
    eng_words_list = []
    for i in range(10,34):
        e_word = read_directory('./refer1/'+ template[i])
        eng_words_list.append(e_word)
    return eng_words_list
eng_words_list = get_eng_words_list()


# 获得英文和数字模板列表（匹配车牌后面的字符）
def get_eng_num_words_list():
    eng_num_words_list = []
    for i in range(0,34):
        word = read_directory('./refer1/'+ template[i])
        eng_num_words_list.append(word)
    return eng_num_words_list
eng_num_words_list = get_eng_num_words_list()


# 读取一个模板地址与图片进行匹配，返回得分
def template_score(template,image):
    #将模板进行格式转换
    template_img=cv2.imdecode(np.fromfile(template,dtype=np.uint8),1)
    template_img = cv2.cvtColor(template_img, cv2.COLOR_RGB2GRAY)
    #模板图像阈值化处理——获得黑白图
    ret, template_img = cv2.threshold(template_img, 0, 255, cv2.THRESH_OTSU)
#     height, width = template_img.shape
#     image_ = image.copy()
#     image_ = cv2.resize(image_, (width, height))
    image_ = image.copy()
```

```
    #获得待检测图片的尺寸
    height, width = image_.shape
    # 将模板resize至与图像一样大小
    template_img = cv2.resize(template_img, (width, height))
    # 模板匹配，返回匹配得分
    result = cv2.matchTemplate(image_, template_img, cv2.TM_CCOEFF)
    return result[0][0]


# 对分割得到的字符逐一匹配
def template_matching(word_images):
    results = []
    for index,word_image in enumerate(word_images):
        if index==0:
            best_score = []
            for chinese_words in chinese_words_list:
                score = []
                for chinese_word in chinese_words:
                    result = template_score(chinese_word,word_image)
                    score.append(result)
                best_score.append(max(score))
            i = best_score.index(max(best_score))
            # print(template[34+i])
            r = template[34+i]
            results.append(r)
            continue
        if index==1:
            best_score = []
            for eng_word_list in eng_words_list:
                score = []
                for eng_word in eng_word_list:
                    result = template_score(eng_word,word_image)
                    score.append(result)
                best_score.append(max(score))
            i = best_score.index(max(best_score))
            # print(template[10+i])
            r = template[10+i]
            results.append(r)
            continue
        else:
            best_score = []
            for eng_num_word_list in eng_num_words_list:
                score = []
                for eng_num_word in eng_num_word_list:
                    result = template_score(eng_num_word,word_image)
                    score.append(result)
                best_score.append(max(score))
            i = best_score.index(max(best_score))
            # print(template[i])
            r = template[i]
            results.append(r)
            continue
    return results


word_images_ = word_images.copy()
# 调用函数获得结果
result = template_matching(word_images_)
print(result)
# "".join(result)函数将列表转换为拼接好的字符串，方便结果显示
print( "".join(result))
```

```
['粤', 'B', '6', 'Y', '8', '8', '9']
粤B6Y889
```

In [110…

```
from PIL import ImageFont, ImageDraw, Image

image_1 = origin_image.copy()
for contour in plate_contours:
    x, y, x_end, y_end = contour
    cv2.rectangle(image_1, (x, y), (x_end, y_end), (0, 255, 0), 2)
    # 计算车牌的中心点
    center_x = (x + x_end) // 2
    center_y = (y + y_end) // 2

#设置需要显示的字体
fontpath = "font/simsun.ttc"
font = ImageFont.truetype(fontpath,64)
img_pil = Image.fromarray(image_1)
draw = ImageDraw.Draw(img_pil)
#绘制文字信息
draw.text((x, y-70),  "".join(result), font = font, fill = (255, 255, 0))
bk_img = np.array(img_pil)
print(result)
print( "".join(result))
plt_show0(bk_img)
```

```
['粤', 'B', '6', 'Y', '8', '8', '9']
粤B6Y889
```

In [ ]: