哈爾濱工業大學

视听觉信号处理

实验报告

题　　目	视听觉信号处理实验三
学　　院	计算学部
专　　业	视听觉处理
学　　号	1180300204
学　　生	陶冶
任课教师	姚鸿勋

哈尔滨工业大学计算机科学与技术学院

2020 秋季

一、 实验目标

综合运用图像处理的知识解决实际问题，以及可能出现的多种多样的情况。

二、 实验内容

1. 对给定的静止状态下的一辆汽车图像进行车牌定位与检测(7 points)，框出车牌保存结果图像(1 points)，描述清楚整个算法流程(5 points)。
2. （选做）在现实情况下，我们可能存在多种多样的情况。高速移动下的车牌；晚上夜景下的车牌；车牌的某些字符部分遮挡；图片中含有多张车牌；车牌倾斜情况。可以从中挑出一个感兴趣的去尝试进行定位（不仅限于上述情况）。（2 points）

三、 实验结果

车牌 1 原图：

提取 1：

车牌 2 原图：

提取 2：

车牌 3 原图：

提取 3：

车牌 4 原图：

提取 4：

车牌 5 原图：

提取 5：

四、 实验分析

选做内容我实现的是车牌倾斜情况。

主要分成两个部分：车牌定位及畸变校正

车牌定位

主要通过颜色特征对车牌进行定位，以 HSV 空间的 H 分量为主，以 RGB 空间的 R 分量和 B 分量为辅，后续再用车牌的长宽比例排除干扰。

畸变校正

在车牌的图像采集过程中，相机镜头通常都不是垂直于车牌的，所以待识别图像中车牌或多或少都会有一定程度的畸变，这给后续的车牌内容识别

带来了一定的困难。因此需要对车牌进行畸变校正，消除畸变带来的不利影响。

4.1 车牌定位

4.1.1 通过颜色特征选定可疑区域

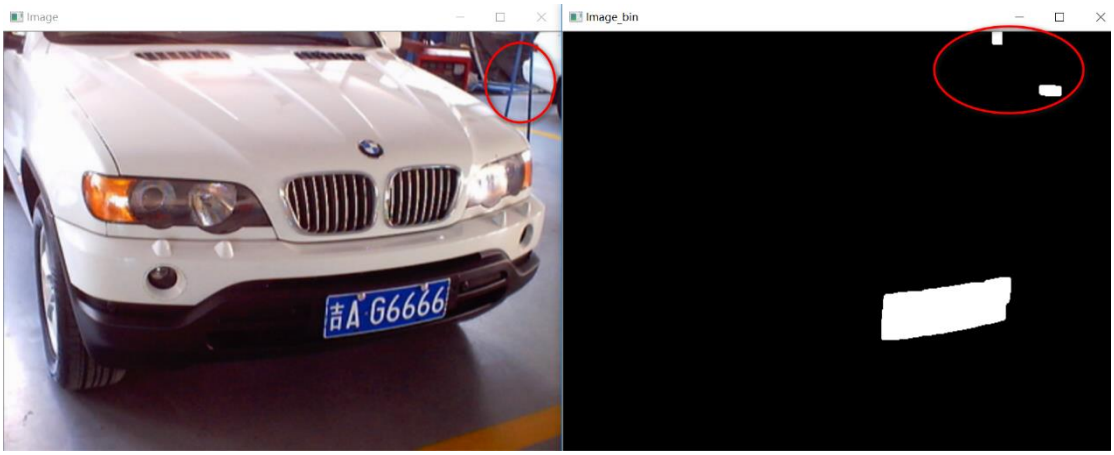
取了不同光照环境下车牌的图像，截取其背景颜色，利用 opencv 进行通道分离和颜色空间转换，经试验后，总结出车牌背景色的以下特征：
(1)在 HSV 空间下，H 分量的值通常都在 115 附近徘徊，S 分量和 V 分量因光照不同而差异较大(opencv 中 H 分量的取值范围是 0 到 179，而不是图像学中的 0 到 360；S 分量和 V 分量的取值范围是到 255)；
(2)在 RGB 空间下，R 分量通常较小，一般在 30 以下，B 分量通常较大，一般在 80 以上，G 分量波动较大；
(3)在 HSV 空间下对图像进行补光和加饱和度处理，即将图像的 S 分量和 V 分量均置为 255，再进行色彩空间转换，由 HSV 空间转换为 RGB 空间，发现 R 分量全部变为 0，B 分量全部变为 255(此操作会引入较大的干扰，后续没有使用）。
根据以上特征可初步筛选出可疑的车牌区域。随后对灰度图进行操作，将可疑位置的像素值置为 255，其他位置的像素值置为 0，即根据特征对图像进行了二值化。二值化图像中，可疑区域用白色表示，其他区域均为黑色。随后可通过膨胀腐蚀等操作对图像进一步处理。

```
for i in range(img_h):
    for j in range(img_w):
        # 普通蓝色车牌，同时排除透明反光物质的干扰
        if ((img_HSV[:, :, 0][i, j]-
115)**2 < 15**2) and (img_B[i, j] > 70) and (img_R[i, j] < 40):
            img_gray[i, j] = 255
        else:
            img_gray[i, j] = 0
```

4.1.2 寻找车牌外围轮廓

选定可疑区域并将图像二值化后，一般情况下，图像中就只有车牌位置的像素颜色为白，但在一些特殊情况下还会存在一些噪声。如上图所示，由于图像右上角存在蓝色支架，与车牌颜色特征相符，因此也被当做车牌识别了出来，由此引入了噪声。
经过观察可以发现，车牌区域与噪声之间存在较大的差异，且车牌区域特征比较明显：
(1)根据我国常规车牌的形状可知，车牌的形状为扁平矩形，长宽比约为 3：1；
(2)车牌区域面积远大于噪声区域，一般为图像中最大的白色区域。

可以通过 cv2.findContours()函数寻找二值化后图像中白色区域的轮廓。



在本例中，因为二值化图像中共有三块白色区域(车牌及两处噪声)，因此返回值 contours 为长度为 3 的 list。list 内装有 3 个 array，每个 array 内各存放着一块白色区域的轮廓信息。每个 array 的 shape 均为(n, 1, 2)，即每个 array 存放着对应白色区域轮廓上 n 个点的坐标。

目前得到了 3 个 array，即 3 组轮廓信息，但我们并不清楚其中哪个是车牌区域对应的那一组轮廓信息。此时可以根据车牌的上述特征筛选出车牌区域的轮廓。

```
#形状及大小筛选校验
det_x_max = 0
det_y_max = 0
num = 0
for i in range(len(contours)):
    x_min = np.min(contours[i][ :, :, 0])
    x_max = np.max(contours[i][ :, :, 0])
    y_min = np.min(contours[i][ :, :, 1])
    y_max = np.max(contours[i][ :, :, 1])
    det_x = x_max - x_min
    det_y = y_max - y_min
    if (det_x / det_y > 1.8) and (det_x > det_x_max ) and (det_y > det_y_max ):
        det_y_max = det_y
        det_x_max = det_x
        num = i
# 获取最可疑区域轮廓点集
points = np.array(contours[num][:, 0])
```

最终得到的 points 的 shape 为(n, 2)，即存放了 n 个点的坐标，这 n 个点均分布在车牌的边缘上。

4.1.3 车牌区域定位

获取车牌轮廓上的点集后，可用 cv2.minAreaRect()获取点集的最小外接矩形。返回值 rect 内包含该矩形的中心点坐标、高度宽度及倾斜角度等信息，使用 cv2.boxPoints()可获取该矩形的四个顶点坐标。

```
# 获取最小外接矩阵，中心点坐标，宽高，旋转角度
rect = cv2.minAreaRect(points)
# 获取矩形四个顶点，浮点型
box = cv2.boxPoints(rect)
# 取整
box = np.int0(box)
```

但我们并不清楚这四个坐标点各对应着矩形的哪一个顶点，因此无法充分地利用这些坐标信息。

可以从坐标值的大小特征入手，将四个坐标与矩形的四个顶点匹配起来：在 opencv 的坐标体系下，纵坐标最小的是 top_point，纵坐标最大的是 bottom_point，横坐标最小的是 left_point，横坐标最大的是 right_point。

```
# 获取四个顶点坐标
left_point_x = np.min(box[:, 0])
right_point_x = np.max(box[:, 0])
top_point_y = np.min(box[:, 1])
bottom_point_y = np.max(box[:, 1])

left_point_y = box[:, 1][np.where(box[:, 0] == left_point_x)][0]
right_point_y = box[:, 1][np.where(box[:, 0] == right_point_x)][0]
top_point_x = box[:, 0][np.where(box[:, 1] == top_point_y)][0]
bottom_point_x = box[:, 0][np.where(box[:, 1] == bottom_point_y)][0]
# 上下左右四个点坐标
vertices = np.array([[top_point_x, top_point_y], [bottom_point_x, bottom_poi
nt_y], [left_point_x, left_point_y], [right_point_x, right_point_y]])
```

4.2 畸变校正

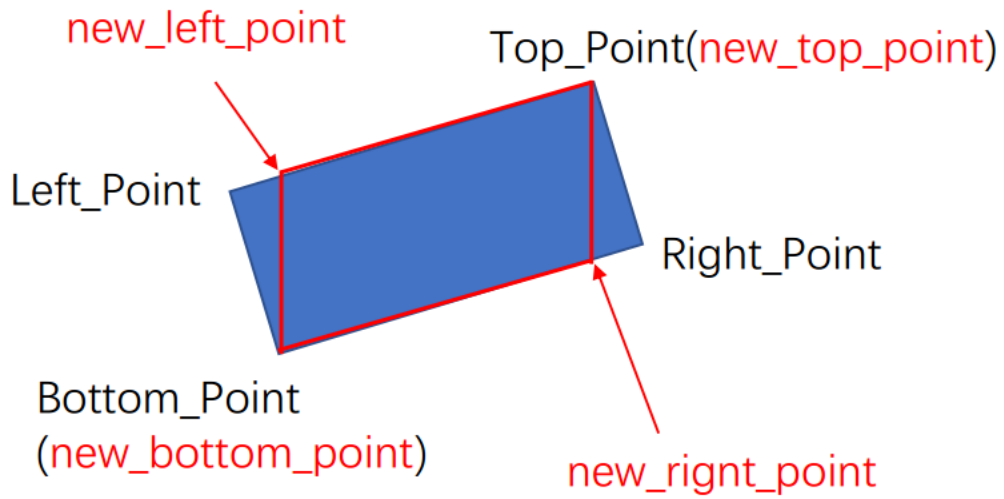
4.2.1 畸变后车牌顶点定位

要想实现车牌的畸变矫正，必须找到畸变前后对应点的位置关系。

可以看出，本是矩形的车牌畸变后变成了平行四边形，因此车牌轮廓和得出来的矩形轮廓并不契合。但有了矩形的四个顶点坐标后，可以通过简单的几何相似关系求出平行四边形车牌的四个顶点坐标。

在本例中，平行四边形四个顶点与矩形四个顶点之间有如下关系：矩形顶点 Top_Point、Bottom_Point 与平行四边形顶点 new_top_point、

new_bottom_point 重合，矩形顶点 Top_Point 的横坐标与平行四边形顶点 new_right_point 的横坐标相同，矩形顶点 Bottom_Point 的横坐标与平行四边形顶点 new_left_point 的横坐标相同。



但事实上，由于拍摄的角度不同，可能出现两种不同的畸变情况。可以根据矩形倾斜角度的不同来判断具体是哪种畸变情况。

判断出具体的畸变情况后，选用对应的几何相似关系，即可轻易地求出平行四边形四个顶点坐标，即得到了畸变后车牌四个顶点的坐标。

要想实现车牌的校正，还需得到畸变前车牌四个顶点的坐标。因为我国车牌的标准尺寸为 440X140，因此可规定畸变前车牌的四个顶点坐标分别为：(0, 0)，(440, 0)，(0, 140)，(440, 140)。顺序上需与畸变后的四个顶点坐标相对应。

```
# 畸变情况 1
if rect[2] > -45:
    new_right_point_x = vertices[0, 0]
    new_right_point_y = int(vertices[1, 1] - (vertices[0, 0]- vertices[1, 0]) / (vertices[3, 0] - vertices[1, 0]) * (vertices[1, 1] - vertices[3, 1]))
    new_left_point_x = vertices[1, 0]
    new_left_point_y = int(vertices[0, 1] + (vertices[0, 0] - vertices[1, 0]) / (vertices[0, 0] - vertices[2, 0]) * (vertices[2, 1] - vertices[0, 1]))
    # 校正后的四个顶点坐标
    point_set_1 = np.float32([[440, 0],[0, 0],[0, 140],[440, 140]])
# 畸变情况 2
elif rect[2] < -45:
    new_right_point_x = vertices[1, 0]
    new_right_point_y = int(vertices[0, 1] + (vertices[1, 0] - vertices[0, 0]) / (vertices[3, 0] - vertices[0, 0]) * (vertices[3, 1] - vertices[0, 1]))

    new_left_point_x = vertices[0, 0]
```

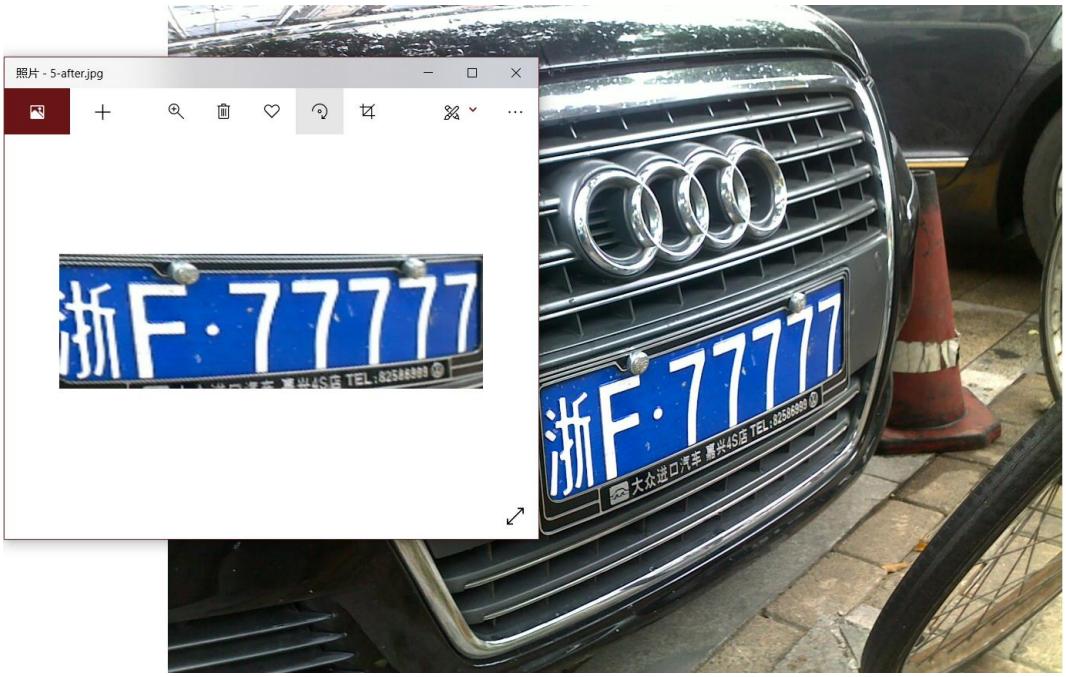
```
    new_left_point_y = int(vertices[1, 1] - (vertices[1, 0] - vertices[0, 0]) / (vertices[1, 0] - vertices[2, 0]) * (vertices[1, 1] - vertices[2, 1]))
    # 校正后的四个顶点坐标
    point_set_1 = np.float32([[0, 0],[0, 140],[440, 140],[440, 0]])

# 校正前平行四边形四个顶点坐标
new_box = np.array([(vertices[0, 0], vertices[0, 1]), (new_left_point_x, new_left_point_y), (vertices[1, 0], vertices[1, 1]), (new_right_point_x, new_right_point_y)])
point_set_0 = np.float32(new_box)
```

4.2.2 校正

该畸变是由于摄像头与车牌不垂直而引起的投影造成的，因此可用cv2.warpPerspective()来进行校正。

```
# 变换矩阵
mat = cv2.getPerspectiveTransform(point_set_0, point_set_1)
# 投影变换
lic = cv2.warpPerspective(img, mat, (440, 140))
```



五、 实验总结

通过本次实验，我运用所学的知识，初步解决了车牌定位的问题，激发了学习视觉处理的兴趣，同时也了解了更多的库函数，掌握了更多的知识。