哈爾濱工業大學

视听觉与信号处理实验报告

实验 3

学　　院　计算机科学与技术学院

专　　业　视听觉信息处理

学　　号　1170300511

学　　生　易 亚 玲

任 课 教 师　姚 鸿 勋

哈尔滨工业大学计算机科学与技术学院

2019 年秋季

一、 实验内容（contents）

对给定的几张车的图片进行车牌定位，定位结果以边框、马赛克或者遮盖形式展示。

二、 实验目的（purposes）

1. 综合运用图像处理的知识解决实际问题。
2. 掌握常见滤波在图像处理中的应用。
3. 掌握常见的图像复原手段。

三、 实验设计、算法和流程(Design, algorithm and procedure)

3.1 将图片映射到 HSV 空间，提取蓝色通道

由于 RGB 空间不能直接识别某一个区域的颜色是什么，所以将图像由 RGB 空间变换到 HSV 空间，根据蓝色的范围为 H(100-124)，S(43-255)，V(46-255)，利用 cv2.inRange 设置阈值，再阈值和 HSV 图像做与运算(cv2.bitwise_and)提取出蓝色部分并转化为灰度图。由于背景中可能也有蓝色部分，所以设置一个最小的值，低于这个值的部分置 0，防止背景部分的干扰。(注：第五张图片整体比较暗，所以需要对灰度图做直方图均衡化处理)。代码实现如下：

```
hsv_img = cv2.cvtColor(img_array, cv2.COLOR_BGR2HSV)  # 转化为hsv空间
blue_low = np.array([100, 43, 46])  # 蓝色较低的值，由统计数据得出
blue_high = np.array([124, 255, 255])  # 蓝色较高的值，由统计数据得出
mask = cv2.inRange(hsv_img, blue_low, blue_high)  # 设阈值，阈值外的为0
blue_img = cv2.bitwise_and(hsv_img, hsv_img, mask=mask)  # 执行阈值的与运算，提取出蓝色部分的图
blue_gray = cv2.cvtColor(blue_img, cv2.COLOR_BGR2GRAY)  # 将蓝色部分的图转化为灰度图
blue_gray = cv2.equalizeHist(blue_gray)  # 直方图均衡化
for i in range(len(blue_gray[:, 0])):
    for j in range(len(blue_gray[0, :])):
        if blue_gray[i][j] < lest_blue:
            blue_gray[i][j] = 0
```

3.2 对图像进行去噪处理

由于图像的噪声未知，使用比较普适的高斯算子去噪，去除噪声对图片边缘检测的影响。代码实现如下：

```
gauss_smooth_img = cv2.GaussianBlur(img_array, (9, 9), 1)  # 高斯平滑
```

3.3 检测图片的边缘

设置上限和下限，用 canny 算子检测图片的边缘。第二幅图片中，车的挡风玻璃的轮廓也很清晰，容易对实验结果形成干扰，所以下限要设置大一些，否则后面经过闭合操作后，整个图像会连成一大片，这样旧没办法检测车牌的位置了。代码实现如下：

```
canny_img = cv2.Canny(img_array, low, high)
```

3.4 开闭操作

执行闭合操作，将车牌中间填满，其中膨胀和腐蚀的次数大约在 5-10 次。如果多余的边缘太多，还可以使用开启适当地分离某些区域。(本实验中涉

及的这几幅图没有这个需求，仅需通过阈值就能很好地去除杂线）

```
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (step, step))  # 闭合操作的参数设置
close_img = cv2.morphologyEx(img_array, cv2.MORPH_CLOSE, kernel)  # 执行闭合操作
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (1, 1))
open_img = cv2.morphologyEx(close_img, cv2.MORPH_OPEN, kernel)
```

3.5　识别处理后的图中的轮廓，并判断哪个轮廓是车牌

使用 cv2.findContours 检测图片中所有的轮廓，计算每个轮廓的面积，找出面积最大的轮廓，根据轮廓生成一个最接近的 box，这个 box 就能将车牌成功定位。由于之前的处理已经很大程度去掉了干扰因子，所以干扰区域很难形成比车牌区域更大的轮廓。代码实现如下：

```
# 寻找边框
max_area = 0
max_edge_num = 0
contours, hierarchy = cv2.findContours(img_array, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
for i in range(len(contours)):
    area = cv2.contourArea(contours[i])
    if area > max_area:
        max_area = area
        max_edge_num = i
rect = cv2.minAreaRect(contours[max_edge_num])
box = np.int32(cv2.boxPoints(rect))
```

四、 实验结果(results)

4.1 蓝色部分的灰度图

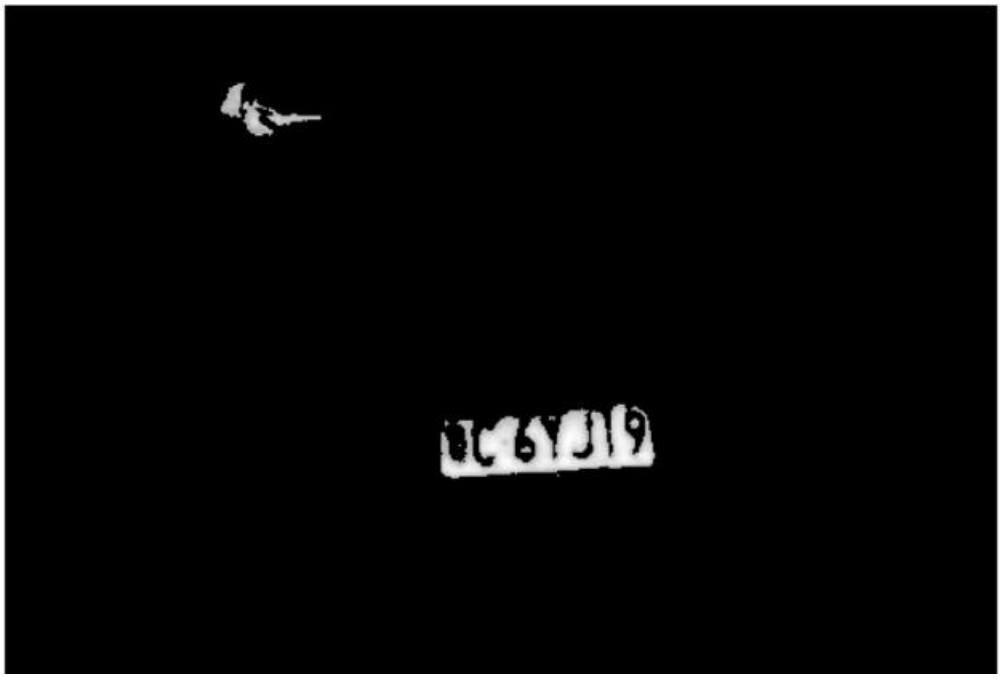

图 – 1.jpg

图 – 2.jpg

图 – 3.jpg

图 – 4.jpg

图 – 5.jpg

4.2 蓝色部分平滑后的图像

图 – 1.jpg

图 – 2.jpg

图 – 3.jpg

图 – 4.jpg

图 – 5.jpg

4.3 Canny 边缘检测

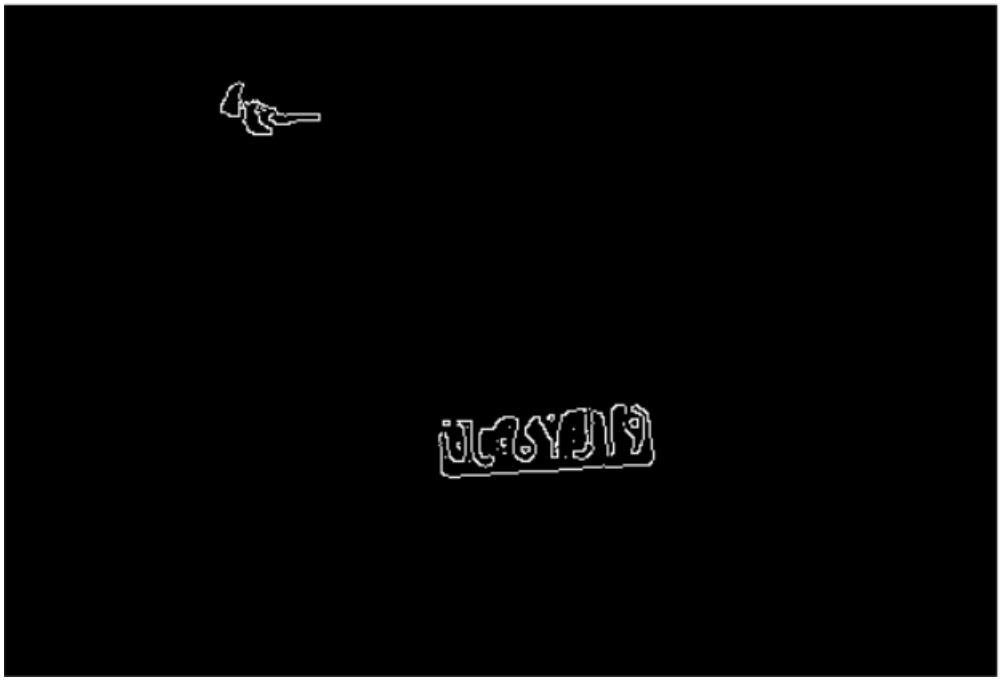

图 – 1.jpg

图 – 2.jpg

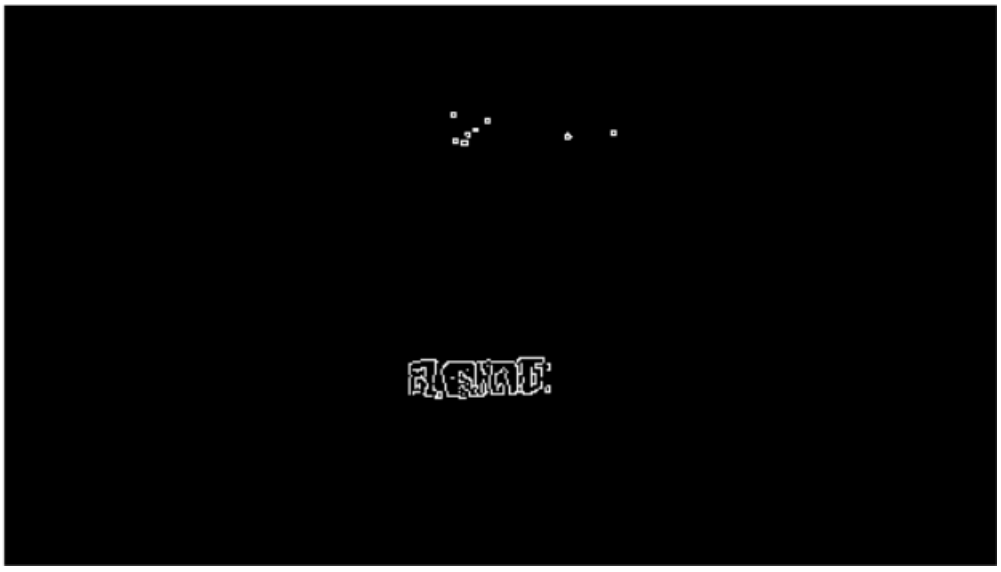

图 – 3.jpg

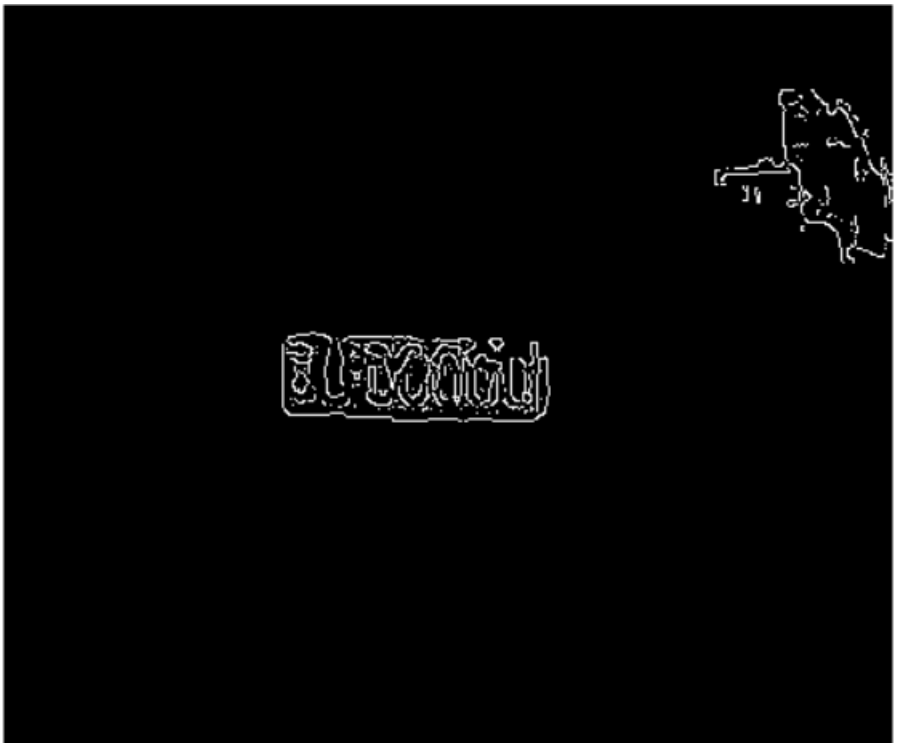

图 – 4.jpg

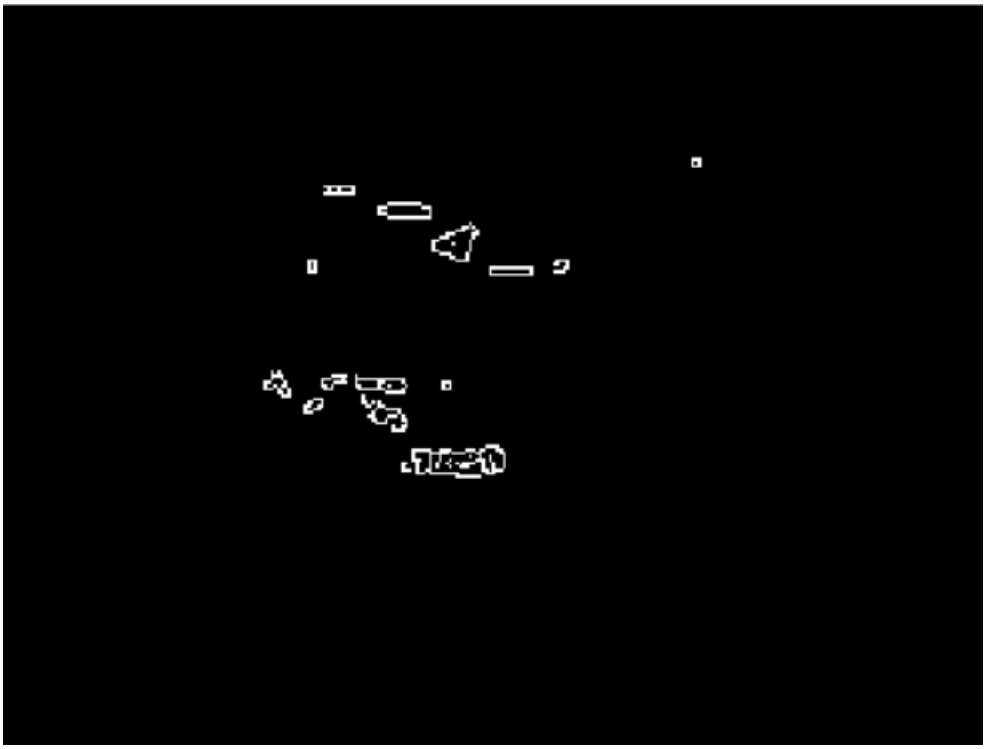

图 – 5.jpg

4.4 闭合操作

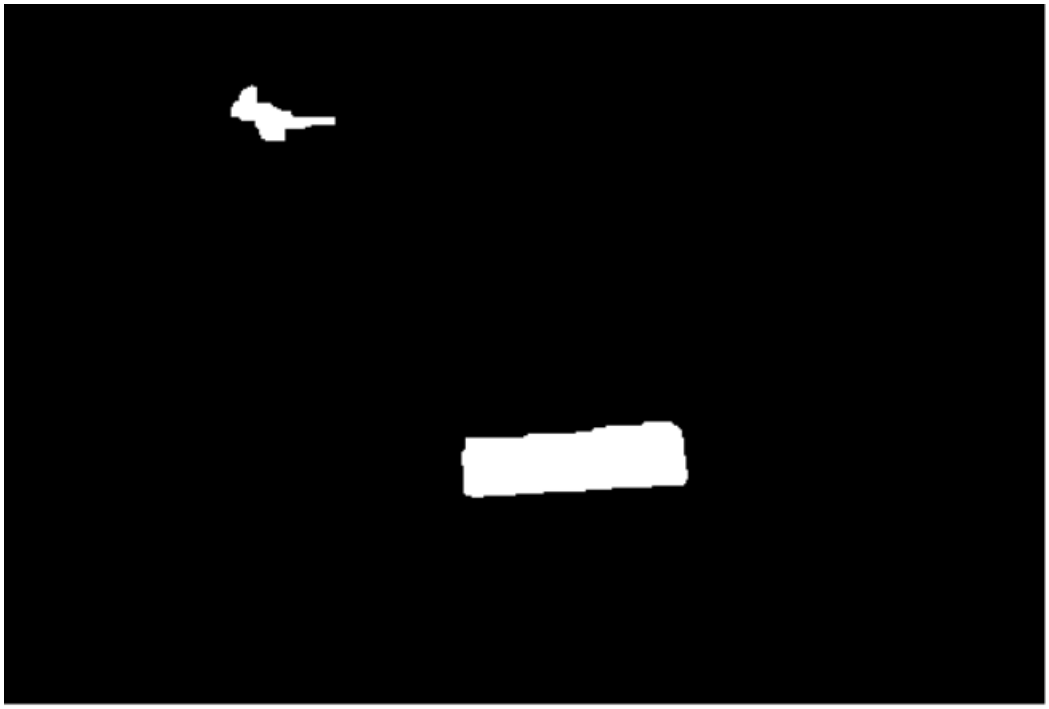

图 – 1.jpg

图 – 2.jpg

图 – 3.jpg

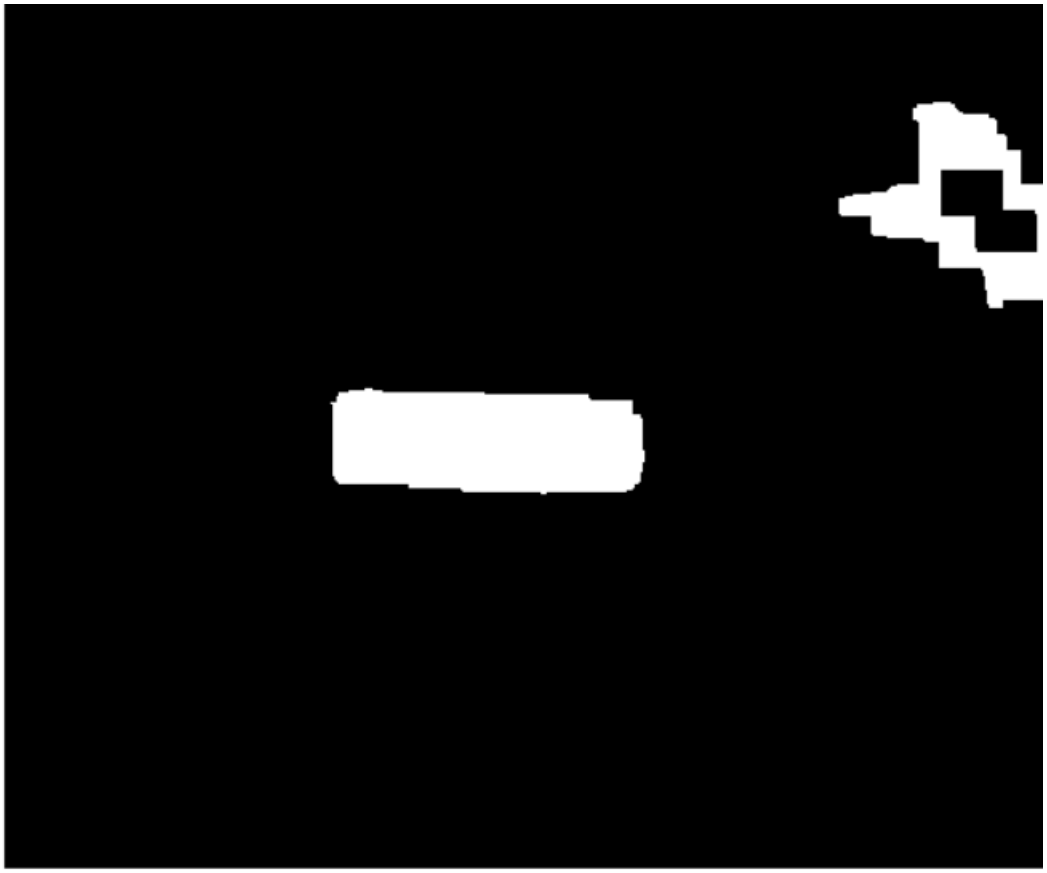

图 – 4.jpg

图 – 5.jpg

4.5 定位车牌区域结果展示

图 – 1.jpg

图 – 2.jpg

图 – 3.jpg

图 – 4.jpg

图 – 5.jpg

五、 结论(conclusion)

1. 当我们需要检测某个物体的时候，需要从待检测物体本身的特点出发，取分析物体本身的一些特性。例如本次实验中的车牌检测，车牌即使在非常恶劣的情况下，一般也是呈现蓝色，所以我们可以将图片中的蓝色部分单独提取出来研究。
2. 在实际检测的过程中，干扰因素很多，比如本次实验中，第五幅图清晰度很低，边缘检测会有点困难，而且会引入很多干扰因素。在以后的检测人物中，噪声只会更多，我们需要仔细对比噪声和目标的微小差异，利用一些操作将这种差异放大，好将噪声和目标分离开。
3. 开启和闭合在目标检测过程中，如果好好运用能够发挥比较好的效果。闭合操作能够将图像中的一些孔补上，开启操作能够去除一些噪声或者将细小部分断开，只留下图片主体。
4. 本次实验中，车牌检测完成后，还可以进一步提取出车牌区域，利用 OCR 识别，能够提取出车牌信息，整个车牌识别系统就完成了。