《数字图像处理》实验报告

姓名： 汪雨卿 学号：19120191

实验八

一. 任务 1

将两段视频中的左右两条车道线检测出来，并用红色线进行标注，如下图
部分所需知识要 11.2（下周 2）的课上介绍，所以本次作业周期是 2 周
提交时无需提交输出视频，截图即可

a) 核心代码：

i. 主程序

```
def do_img(img):
    roi_part = np.array([[(0, img.shape[0]), (460, 325), (520, 325), (img.shape[1], img.shape[0])]])  # roi_part: 三
    gray = cv.cvtColor(img, cv.COLOR_RGB2GRAY)                                                        # 图像转换为灰度图
    blur_gray = cv.GaussianBlur(gray, (gk_size, gk_size), 0, 0)                                       # 使用高斯模糊去噪声
    edges = cv.Canny(blur_gray, canny_low, canny_high)                                                # 使用Canny进行边缘检测
    roi_edges = roi_mask(edges, roi_part)                                                             # 对边缘检测的图像生成图像蒙板，去
    img_line = hough_lines(roi_edges, rho_step, theta, threshold, min_len, max_gap)                   # 使用霍夫直线检测，并且绘制直线
    img_out = cv.addWeighted(img, 0.8, img_line, 1, 0)                                                # 将处理后的图像与原图做融合
    return img_out
```

ii. 进行边缘检测，保留感兴趣区域的图像

```
18
19  def roi_mask(img, intere_vec):                  # img是输入的图像，intere_vec是兴趣区的四个点的坐标（三维的数组）
20      mask = np.zeros_like(img)                   # 创建副本：生成与输入图像相同大小的图像，并使用0填充,图像为黑色
21      if len(img.shape) > 2:
22          img_channel = img.shape[2]              # 获取图片宽和高
23          mask_color = (255,) * img_channel       # 如果 img_channel=3, 则为(255,255,255)
24      else:
25          mask_color = 255
26      cv.fillPoly(mask, intere_vec, mask_color)   # 使用白色填充多边形，形成蒙板
27      img_mask = cv.bitwise_and(img, mask)        # img&mask，经过此操作后，兴趣区域以外的部分被蒙住了，只留下兴趣区域的图像
28      return img_mask
```

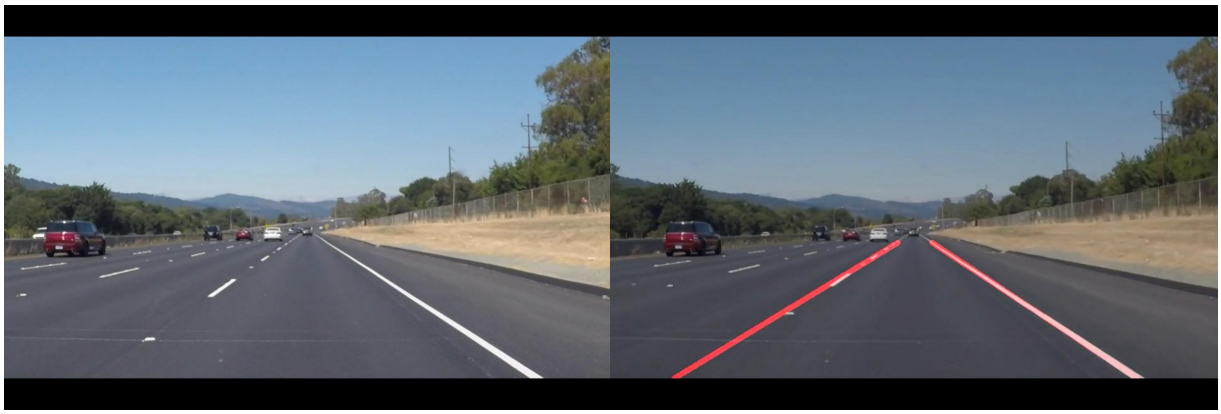
iii. 霍夫直线检测

```
41  def hough_lines(img, step, theta_h, thresh, min_line_len, max_gap):
42      lines = cv.HoughLinesP(img, step, theta_h, thresh, np.array([]), minLineLength=min_line_len,maxLineGap=max_gap)
43      img_line = np.zeros((img.shape[0], img.shape[1], 3), dtype=np.uint8)  # 生成绘制直线的绘图板，黑底
44      add_line(img_line, lines)
45      return img_line
46
```

```python

def add_line(img, lines, color=[255, 0, 0], thickness=8):
    left_lines, right_lines = [], []                # 用于存储左边和右边的直线
    for line in lines:                              # 对直线进行分类
        for x1, y1, x2, y2 in line:
            k = (y2 - y1) / (x2 - x1)
            if k < 0:
                left_lines.append(line)
            else:
                right_lines.append(line)

    if len(left_lines) <= 0 or len(right_lines) <= 0:
        return img

    clean_lines(left_lines, 0.1)    # 弹出左侧不满足斜率要求的直线
    clean_lines(right_lines, 0.1)   # 弹出右侧不满足斜率要求的直线
    left_points = [(x1, y1) for line in left_lines for x1, y1, x2, y2 in line]                    # 提取左侧直线族中的所有的第一个点
    left_points = left_points + [(x2, y2) for line in left_lines for x1, y1, x2, y2 in line]      # 提取左侧直线族中的所有的第二个点
    right_points = [(x1, y1) for line in right_lines for x1, y1, x2, y2 in line]                  # 提取右侧直线族中的所有的第一个点
    right_points = right_points + [(x2, y2) for line in right_lines for x1, y1, x2, y2 in line]   # 提取右侧侧直线族中的所有的第二个点

    left_vtx = calc_lane_intere_vec(left_points, 325, img.shape[0])          # 拟合点集，生成直线表达式，并计算左侧直线在图像中的两个端点的坐标
    right_vtx = calc_lane_intere_vec(right_points, 325, img.shape[0])        # 拟合点集，生成直线表达式，并计算右侧直线在图像中的两个端点的坐标

    cv.line(img, left_vtx[0], left_vtx[1], color, thickness)    # 画出直线
    cv.line(img, right_vtx[0], right_vtx[1], color, thickness)  # 画出直线


```

b) 实验结果截图

视频 1



视频 2

四、实验小结

本实验和以往实验的最大区别在于本次实验是对于视频的操作。通过学习，可以利用 opencv 库的视频处理方法，对视频的内容切片。进而将复杂的问题，转换为与前两个实验相关的处理方式，即先对图片帧进行处理，再对处理之后的图片重新拟合成视频。对于图片的处理，主要运用图像变化的功能，利用霍夫变换去检测图像中的直线部分，然后对检测得到的区域进行颜色填充。虽然本实验只是一个简单的车道识别，也让我初窥到数字图像处理在生活中的实际应用。