数字图像处理期末大作业

20122998，金怀德

任务一

1）作业内容

将所提供的公路视频中的车道线检测出并用红线进行标注，提交完成的截图。

2）实现思路

核心思路为使用边缘检测 canny 算子识别图像边缘，由于噪声和光照不匀等干扰，边缘点可能并不连续，选择使用霍夫变换检测间断点边界的形状，通过斜率正负划分为左右两侧，通过 numpy 的直线拟合函数，以最小二乘法拟合确定出两侧的唯一线段，完成检测。模拟现实，车载探测仪器功能需尽量精确且实时，因此需采取的措施来降低干扰，减少计算量：最开始以灰值化和高斯模糊预处理图像降低干扰，并且使用掩膜抠图获取 ROI 区域。用平方差剔除差异值大的离群线段。
检测出车辆两侧车道线。具体实现步骤如下：

1.图像预处理

使用灰度化减少计算量；由于噪音会轻易影响到图像的边界，不利于后续边缘检测，故采用高斯滤波平滑一些纹理较弱的非边缘区域。

2.Canny 边缘检测

选择合适的 maxVal 和 minVal 为 50 和 120，作为弱边缘和强边缘的阈值。计算图像 x 轴和 y 轴的梯度以及梯度的合方向，使用滞后跟踪边缘，抑制未连接到强边缘的边缘，完成初步边缘检测。

3.将感兴趣区域（ROI）使用掩膜截取

根据车道线分布在车体两侧设置 ROI：创建一个掩膜，规定 4 个标点将 ROI 规定为一个正立梯形，将它和 Canny 边缘检测后的图像进行布尔运算，以此保留感兴趣的位置，获取确切的车道线边缘。

4.通过霍夫变换检测线段集合

对边缘二值图像使用 cv2.HoughLines()完成霍夫直线变换，将所有像素点映射到极坐标系，同一直线的所有点将会交汇于同一（r,theta），以此获取所有线段的坐标集合。

5.获取左右车道线段集合

使用列表生成式，结合线段斜率计算函数，划分斜率大于 0 和小于 0 分别为左右车道线线段集合。而后计算斜率均值，并求得所有线段斜率平方差，并设置边界值 0.1，当线段和均值的平方差大于 0.1，就判定为差异值较大的利群线段，删除之。

6.用最小二乘法拟合车道线

使用 np.polyfit 函数，通过最小化误差的平方和来寻找线段的最佳匹配多项式系数。再获取线段集合中横坐标值最小以及最大的值，依靠 polyval 计算出拟合多项式对应的纵坐标值，拟合出车道线段。

7.标注车道线

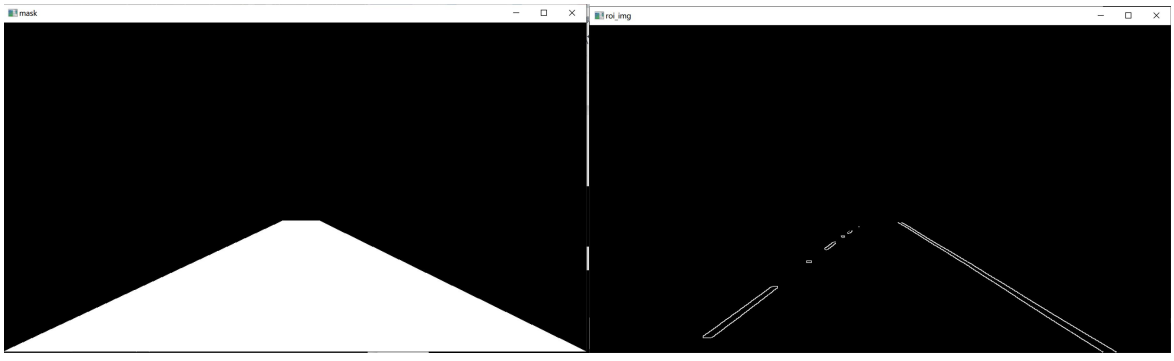
使用 cv2.line 将拟合出的车道线，设置红色和粗细 3，标注在原始帧图像上。

3）核心代码分析

```
def frame_process(color_img):# 对每一帧的图像的处理如下
    # 1. 灰度化、滤波
    kernal = (5, 5)  # 高斯滤波核形状
    stride = 1  # 高斯滤波横向滑动步距默认为 1
    gray_img = cv2.cvtColor(color_img, cv2.COLOR_BGR2GRAY)  # 灰度化
    gaus_img = cv2.GaussianBlur(gray_img, kernal, stride)  # 高斯滤波
    # 2. Canny 获取边缘图像
    canny_maxth = 120  # Canny 边缘检测最高阈值
    canny_minth = 50  # Canny 边缘检测最低阈值
    edges_img = cv2.Canny(gaus_img, canny_minth, canny_maxth)
    # 3.截取感兴趣区域
    roi_img = roi_mask(edges_img)
    # 4. 霍夫变换直线提取所有线段集合，列表嵌套列表元素:端点坐标 x1,x2,y1,y2
    lines = cv2.HoughLinesP(roi_img, rho=1, theta=np.pi / 180, threshold=15, minLineLength=40,
maxLineGap=20)
    # rho 和 theta 为极坐标系下系数的精度，一个范围内获取线段交点。最后两个参数分别规定了
线段的最短长度阈值，和同一直线间的最大距离
    # 5. 获取左右车道线段集合
    # a.按照斜率划分左右车道
    left_lines = [line for line in lines if calculate_slope(line) > 0]
    right_lines = [line for line in lines if calculate_slope(line) < 0]
    # b.剔除离群线段
    left_lines = clean_lines(left_lines)
    right_lines = clean_lines(right_lines)
    # 6. 最小二乘法获取拟合后的直线
    fit_lines = least_squares_fit(left_lines), least_squares_fit(right_lines)
    # 7.标记车道线
    left_fit_line, right_fit_line = fit_lines
    cv2.line(color_img, left_fit_line[0], left_fit_line[1], color=(0, 0, 255), thickness=3)
    cv2.line(color_img, right_fit_line[0], right_fit_line[1], color=(0, 0, 255), thickness=3)
return color_img
```

4）结果及分析

下图左侧白色区域为掩膜形状，右侧为将二值化边缘检测图像和掩膜进行布尔运算后的 ROI 图像。



下方左图和右图分别为对视频 1 和 2 的车道线检测结果，结果清晰直观。

任务二

1）英语答题卷的选项和条形码识别

A. 答题卡图像的旋转校正

检测试卷四周的正方形定位符，配合 opencv 的几何变换方法，对存在倾斜的试卷图像进行旋转校正，完成答题区的定位。

B. 条形码识别

截取试卷中的条形码，对其进行形态学操作，识别其码字后，标注于试卷上。

C. 选项识别

截取指定答题区域，选择出正确填涂的目标选项进行标注，将识别出的对应选项标注于试卷上。

2）实现思路

A. 检测试卷图像的倾斜度后校正

1. 检测图像边缘

为减少噪声干扰以及计算量，使用 3 步预处理：图像灰度化，非局部平均去噪，高斯滤波，之后设定 Canny 算子阈值，检测图像边缘。

2. 寻找试卷定位正方形

以边缘检测的长宽和边缘形状为过滤条件，从所有边缘轮廓中得到试卷四周正方形定位坐标。其中过滤的阈值类似于调参过程，需要反复尝试获取相对较好的限定阈值。

3. 试卷位置矫正

通过 4 个坐标进行仿射变换，将试卷位置矫正，输出矫正图像为新图像。

B. 条形码识别

1. 图像分割

由于试卷方向已经矫正，可以通过创建掩膜图像进行布尔运算截取条形码大致所处位置。

2. 识别条形码轮廓

条形码在灰度化和高斯滤波后，可利用 sobel 算子计算梯度差获取边缘，取代 canny 算子加快计算，并且相对与 Canny 算子，使用 sobel 算子提取边缘得到的灰度值更大，利于后续识别码字信息。

3. 识别条形码的码字信息

识别条形码，将对应的值写入原图中。

C. 选项识别

1. 图像分割

使用 ROI 指定答题区域位置。

2. 强化正确填涂的目标选项

由于填涂选项框周围的像素变化较大，需要识别并突出正确的填涂，故使用闭运算处理填涂和选项框的小空隙，使用 Canny 算子尽可能识别填涂边缘，以轮廓的边缘面积及每个轮廓间的距离绝对值为筛选条件，过滤细小的干扰填涂，识别正确填涂的目标，使用黑线在周围标注，强化填涂。

3. 目标选项判定

第二次使用 Canny 算子，选中正确的填涂轮廓，根据轮廓坐标，获取其重心位置，并将其与预先录入的选项和坐标的映射进行比较，识别其选项信息在右侧，写入原始图像。

3）核心代码分析

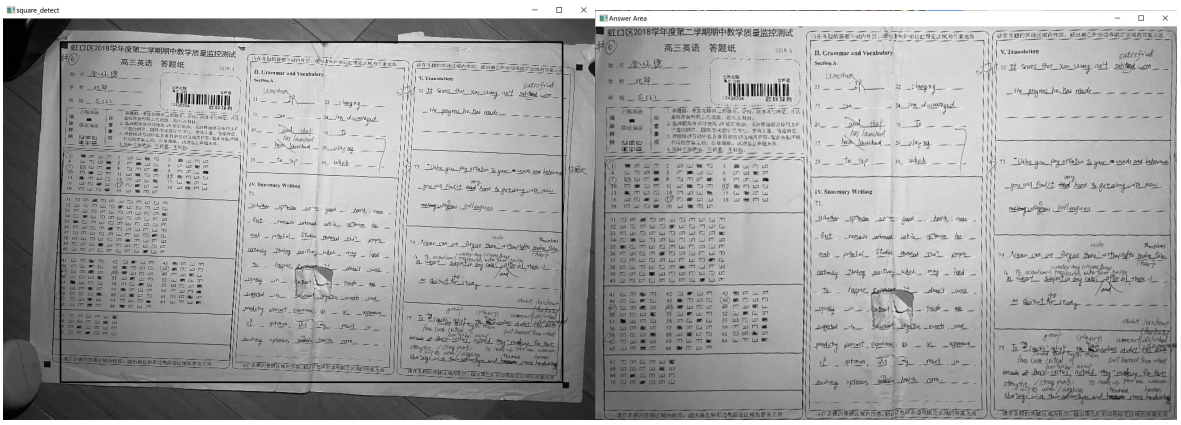
```
def entire_process(img): #完整阅卷过程
    four_point = detect_square(img)  # 预处理检测试卷边缘后识别试卷四周的定位正方形
    img_dst = Homography(img, four_point)  # 根据定位正方形坐标进行仿射变换，矫正试卷图像倾斜
    roi_img1, roi_img2 = roi_mask(img_dst)  # 获得选项和条形码两个 ROI
    barcode_img = barcode_detect(roi_img2)  # 检测条形码位置，并识别其具体值
    answer_img = sure_if_fill(roi_img1)  # 边缘检测配合闭运算填充空隙，强化填涂
    check_img = check_answer(answer_img)  # 识别填涂结果将批阅结果绘制到图像上
    final_img = integrate_final(barcode_img, check_img, img_dst)  # 整合图像，返回完整阅卷结果
    return final_img

def check_answer(image):# 检测填涂选框，识别标注其选项
    copy = image.copy()
    kernel = cv.getStructuringElement(cv.MORPH_RECT, (3, 3)) # 闭运算算子
    dst = cv.morphologyEx(copy, cv.MORPH_CLOSE, kernel, iterations=1)  # 闭运算
    canny_img = cv.Canny(dst, 250, 256) # 设定 Canny 算子阈值
    contours, hierarchy = cv.findContours(canny_img,
cv.RETR_EXTERNAL,cv.CHAIN_APPROX_NONE) # 在二值图像（binary image）中寻找轮廓，只检测最外围轮廓 且保留该方向的终点坐标
    m = 30
    contours = tuple(reversed(list(contours)))# 将识别的边缘信息倒序，方便从上到下阅卷
    mem_x = 0 # 记录上次选项横坐标位置
    answer_list = []
    for i, c in enumerate(contours):
```

```
            area = cv.contourArea(c) #  计算选项轮廓面积
            x, y, w, h = cv.boundingRect(c) #  找到角点坐标  高和宽
            if area > 2.5 and abs(mem_x - x) > 5:   #  筛选正确选项
                m += 1
                mem_x = x
                mm = cv.moments(c)   #  几何重心的获取
                cx = mm['m10'] / mm['m00']   #  获取几何中心的 x
                cy = mm['m01'] / mm['m00']   #  获取几何重心的 y
                cv.putText(copy, str(m), (np.int(cx) + 15, np.int(cy)), cv.FONT_HERSHEY_COMPLEX,
0.4, (255, 0, 0), 1) #标注选项题号
                answer_r, color = option(cx, m - 31)   #  识别选项详细
                cv.drawContours(copy, c, -1, (255, 0, 0), 4)   #  绘制轮廓线
                cv.putText(copy, "    " + answer_r, (np.int(cx) + 20, np.int(cy)),
cv.FONT_HERSHEY_SIMPLEX, 0.4, color, 1) #  标注选项
                answer_list.append([np.int(cx), np.int(cy)])   #  集合点作为集合
    return copy
```

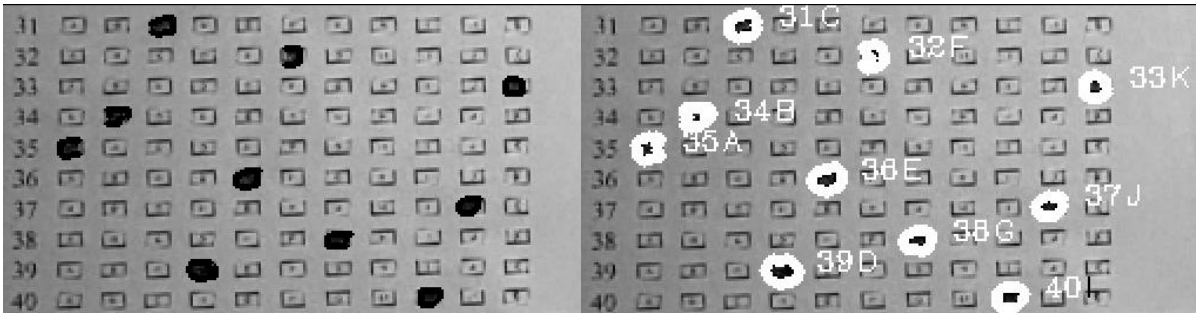
4）结果及分析

通过左上右下的定位矩形定位倾斜试卷的答题区域，仿射变换后得到矫正图像：



识别条形码区域后识别码字并标注：

使用闭运算后识别边缘，绘制轮廓强调填涂，随后再次边缘识别后选中选项并标注选项：



原图和完整阅卷结果对比，可见选项和条形码码字被正确识别并标注。