电子科技大学

University of Electronic Science and Technology of China

研究生期末课程设计报告

课程名称：数字图像处理

设计题目：基于特征融合的道路线提取

学生姓名：孙佳伟

学　　号：202122060713

指导教师：李玉霞

2022 年 5 月

写在前面的报告说明

1、工程项目地址：Digital_Image_Processing/code/road_lane_line_detection-master at main · sinary-sys/Digital_Image_Processing (github.com)

工程为 pycharm 工程，直接使用 pycharm 打开，配置需要的包，即可运行；

```
challenge_output = '99.mp4'
clip2 = VideoFileClip('9.mp4')
challenge_clip = clip2.fl_image(process_image)
challenge_clip.write_videofile(challenge_output, audio=False)
```

challenge_output 为输入视频，clip2 = VideoFileClip('9.mp4')为输出视频

2、最终输出视频地址：

Digital_Image_Processing/99.mp4 at main · sinary-sys/Digital_Image_Processing (github.com)

目录

基于特征融合的道路线提取

一、研究现状与分析

1 边缘检测方法的研究现状

边缘是图像各个区域相互隔离，形成了可辨识的内容，是图像的最基本特征。提取边缘是对图像轮廓的一个处理，对于边界处，灰度值变化比较剧烈的地方就定义为边缘或拐点[11]。L.GRobert et al. 1(959)最早提出了图像边缘检测的概念，从那以后，边缘提取的理论和算法不断改善，更新。边缘检测最早的思路是从图像的高低频信号出发，边缘提取的主要采取的方法是微分方法。目前，一阶主要有 Roberts 算子、Prewitt 算子、Sobel 算子以及 Kirsch 等算子。关于二阶导数的边缘检测，目前主流被使用的算子有 Laplacian 算子。在处理邻域像素灰度值的基础上，上述的算子对图像进行边缘提取。这些算子的共同的一个缺点是易受噪声干扰。由此带来的问题是检测的边缘比较粗且定位精度不够高。为此，关于克服噪声，Hildreth 和 Marretal 提出了 LoG 算子。LoG 算子经常被用于视觉神经细胞与生理中在数学模型上[12]。研究者依此建立了计算机视觉和人类视觉研究之间的对应。J.Canny（1986）提出了的 Canny 算子边缘检测算子具有三个优化标准，分别为精定位、低出错率、独特响应。

除了微分算子，数学形态学也较多的应用于边缘提取领域，它能保持图像的基本轮廓、形状特征并能大量简化图像数据，大大提高了图像分析和处理的速度。边缘检测算法在数学形态学领域，通过构造不同的结构单元以处理不同的图像目标。而目前关于灰度形态学在边缘检测中的应用也从最开始的单单的理论构想逐渐走向实际应用及成熟。随着科技高速发展，图像边缘提取的技术也日益趋于成熟，一些新兴的检测算法得到了深度研究和广泛应用。如在机器学习、小波变换、统计学方法模糊理论等领域内的边缘提取方法等。

2 道路提取方法的研究现状

人们从图像中希望了解的目标信息通常分为三类别，分别为：点类别目标（如敌人军事据点，关键设施等），线类别目标（如河流，道路等）和面类别目标（如

植被覆盖情况等）。在三个类别之中，了解线类别目标的信息对人们日常生活带来的便利最为显著[13]，而道路网络的提取是其中一个重要应用。依靠
图像的现有道路提取的方法可以分为以下两类：

1.半自动道路特征提取

半自动化道路特征的提取关键点在于计算机和人交互协同，通过依靠人为的提供道路先验信息，再由计算机设定算法进行判别，同时向人反馈它的识别结果从而能够得到实时的人工校正。半自动化道路提取可用如下图所示的 4 个步骤分解表示。

图 1- 1 半自动化道路提取流程

在先行研究中，有如下几种方法得到了较好的结果：

（l）基于边缘跟踪的方法[14]

首先根据先验信息设定初始的种子点和方向，以种子点为起点，在初始方向上寻找边缘点作为新的种子点，直至边缘信息结束。以结束点作为新的跟踪种子点，连接各边缘检测路段组合而得到较准确的道路网[15]。

（2）最小二乘模板匹配方法[16]

此方法先初始化已有的特征，再计算以最小二乘法预算模板与给定遥感图像之间的几何变形参数。基于该方法，可得到遥感图像中道路曲线相关参数的数学表达式此方法在精度有较大提升[17]。

2.自动道路特征提取

基于图像的主要轮廓和特点，利用计算机视觉、人工智能、模式识别等方法对图像中路面的特征进行处理，以达到主动识别出路面或某一种类型的道路的目的。而半自动化的方法却不可缺少人工的辅助识别。而在当前现阶段，并没有出现较成熟并完善的自动化提取算法。最常用的几种方法介绍如下：

（1）多分辨率提取算法[18]

图像的不同分辨率，可体现出道路异样的特征:

图像低分辨率时，道路表现出清晰的主体和拓扑信息，噪声影响对其干扰

小；图像在高分辨率时，道路的具体特征可以被清晰反映，但噪声影响对其干扰大（如遮挡道路或障碍物等）[19]。充分融合两方的优势将其联合用以提取道路特征成为了一种自动道路提取方法。

（2）基于平行线对的道路提取[20]

图像中，道路的两边近似平行，目前有很多道路提取的算法以此特点为基础[21]。道路自动识别的方法从下面 3 个层次分别进行了探索：低层次中的边缘检测；中层次中的特征处理；高层次中的识别处理。此思想是在 Marr 视觉理论的基础奠定下[22]。大量实验表明，此法有较高的识别效果，但通用灵活程度不高[23]。

（3）Snakes　方法[24]

高分辨率遥感图像的噪声干扰问题导致待检测道路段时常不连续,并且严重干扰道路两边的相对平行性[25]。针对这个问题,运用计算机图形学的 snakes 概念，利用 Ribbon snakes 来使道路边缘得到进一步优化,获得道路两边相对平行的道路信息[26]。此法的优势在于得到非常清晰的道路显示,但也可能带来大量的错误识别结果。

（4）利用其它数据源作为对比或匹配的对象[27]

可以将前人收集统计好的地图数据或者其他数据与遥感图像先预处理，以得到路面粗略信息将其赋值成为初始输入信息。而再依据图像提取到的特征精细识别和大致定位[28]。提取道路的特征数据反馈给用的对比的数据源，使之保持与源数据的对比来提高准确度和精确率。

（5）网状模型的方法[29]

此法充分利用道路网的拓扑特性，具体流程为先提取线性或曲线特征然后得到离散的候选路段，再预设评价函数去使道路连接，然后基于此对候选道路段连接形成网状的处理[30]。

二、算法原理

1 图像增强

图像处理领域中，图像增强（Enhancement）是一种被广为采用的基本方法，

遥感图像中也有广泛应用。由于光照和天气的影响，遥感图像中道路和周围地物或出现对比度较低的现象。所以利用图像增强的手段使得作为感兴趣区域的道路得以强调，削弱或去除周围地物的特征，对于后续的处理尤为重要。

图像增强的主要方法有基于空域或频域的参数调整，本研究中，主要被采用的方法是基于对像素进行处理的空域方法：对灰度图像进行直方图均衡化以及分段线性变换方法。

2 直方图

直方图是多种空间域处理技术的基础。直方图操作能有效地用于图像增强。除了提供有用的图像统计资料外，直方图固有的信息在其他图像处理应用中也是非常有用的，如图像压缩与分割。直方图在软件中易于计算，也适用于商用硬件设备，因此，它们成为了实时图像处理的一个流行工具。

直方图是图像的最基本的统计特征，它反映的是图像的灰度值的分布情况。直方图均衡化的目的是使图像在整个灰度值动态变化范围内的分布均匀化，改善图像的亮度分布状态，增强图像的视觉效果 。灰度直方图是图像预处理中涉及最广泛的基本概念之一。

图像的直方图事实上就是图像的亮度分布的概率密度函数，是一幅图像的所有象素集合的最基本的统计规律。直方图反映了图像的明暗分布规律，可以通过图像变换进行直方图调整，获得较好的视觉效果。

直方图均衡化是通过灰度变换将一幅图像转换为另一幅具有均衡直方图，即在每个灰度级上都具有相同的象素点数的过程。

3 形态学处理

数学形态学是以形态结构元素为基础对图像进行分析的数学工具。它的基本思想是用具有一定形态的结构元素去度量和提取图像中的对应形状以达到对图像分析和识别的目的。数学形态学的应用可以简化图像数据，保持它们基本的形状特征，并除去不相干的结构。数学形态学的基本运算有 4 个：膨胀、腐蚀、开启和闭合。它们在二值图像中和灰度图像中各有特点。基于这些基本运算还可以推导和组合成各种数学形态学实用算法。基本的形态运算是腐蚀和膨

胀。

在形态学中，结构元素是最重要最基本的概念。结构元素在形态变换中的作用相当于信号处理中的 滤波窗口 。用B(x)代表结构元素，对工作空间E中的每一点 x ，腐蚀和膨胀的定义为：

腐蚀 X=E⊙B(x);

膨胀 Y=E⊗B(y)。

用$B(x)$ 对 E进行膨胀的结果就是把结构元素 B 平移后使 B 与 E 的交集非空的点构成的集合。先腐蚀后膨胀的过程称为开运算。它具有消除细小物体 ，在纤细处分离物体和平滑较大物体边界的作用。先膨胀后腐蚀的过程称为闭运算。它具有填充物体内细小空洞，连接邻近物体和平滑边界的作用。

可见，二值形态膨胀与腐蚀可转化为集合的逻辑运算，算法简单，适于并行处理，且易于硬件实现，适于对二值图像进行图像分割、细化、抽取骨架、边缘提取、形状分析。但是，在不同的应用场合，结构元素的选择及其相应的处理算法是不一样的，对不同的目标图像需设计不同的结构元素和不同的处理算法。结构元素的大小、形状选择合适与否，将直接影响图像的形态运算结果。因此，很多学者结合自己的应用实际，提出了一 系列的改进算法。如梁勇提出的用多方位形态学结构元素进行边缘检测算法既具有较好的边缘定位能力，又具有很好的噪声平滑能力。许超提出的以最短线段结构元素构造准圆结构元素或序列结构元素生成准圆结构元素相结合的设计方法，用于骨架的提取，可大大减少形态运算的计算量，并可同时满足尺度、平移及旋转相容性，适于对形状进行分析和描述。

三、算法流程及实现

1 目标

(1)计算给定一组棋盘图像的相机校准矩阵和失真系数。

(2)对原始图像应用失真校正。

(3)使用颜色转换、渐变等创建阈值二进制图像。

(4)应用透视变换来校正二进制图像（“鸟瞰图”）。

(5)检测车道像素并拟合以查找车道边界。

(6)确定车道的曲率和车辆相对于中心的位置。

(7)将检测到的车道边界扭曲回原始图像。

(8)车道边界的输出可视化显示以及车道曲率和车辆位置的数值估计。

(9)在**老师给的视频**中表现良好。

2 相机校准

cv2.findChessboardCorners 并用于获取校准矩阵。cv2.calibrateCamera 此矩阵使失真的图像不失真。

2.1 导入有用的模块

```
import numpy as np
import cv2
import pickle
import glob
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
```

2.2 校准矩阵和失真系数

对象点是现实世界中棋盘角的 （x，y，z） 坐标。假设棋盘固定在 z 平面（z=0）上。

```
# Camera calibration
def camera_calibration():
    # prepare object points, like (0,0,0), (1,0,0), (2,0,0) ....,(6,5,0)
    objp = np.zeros((6 * 9, 3), np.float32)
    objp[:, :2] = np.mgrid[0:9, 0:6].T.reshape(-1, 2)

    # Arrays to store object points and image points from all the images.
    objpoints = []  # 3d points in real world space
    imgpoints = []  # 2d points in image plane.

    # termination criteria
    criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30,
0.001)

    # image size
    img_size = None
```

```
    # Make a list of calibration images
    chessboard_images = glob.glob('./camera_cal/calibration*.jpg')

    for image in chessboard_images:
        img = mpimg.imread(image)
        gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
        img_size = gray.shape[::-1]

        # Find the chessboard corners
        ret, corners = cv2.findChessboardCorners(gray, (9, 6), None)

        # If found, add object points, image points
        if ret == True:
            objpoints.append(objp)
            corners2 = cv2.cornerSubPix(gray, corners, (11, 11), (-1, -1),
criteria)
            imgpoints.append(corners2)

    # mtx: Camera matrix
    # dist: Distortion coefficients
    ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(objpoints,
imgpoints, img_size, None, None)
    # Save the camera calibration result for later use (we won't worry
about rvecs / tvecs)
    dist_pickle = {}
    dist_pickle['mtx'] = mtx
    dist_pickle['dist'] = dist
    print(mtx)
    print(dist)
    pickle.dump(dist_pickle, open('calibration.p', 'wb'))
```

2.3 失真校正

```
# Distortion correction
def cal_undistort(img):
    # undistortion on the original image instead of gray-scaled image
    data = pickle.load(open('calibration.p', 'rb'))
    mtx, dist = data['mtx'], data['dist']
    undist = cv2.undistort(img, mtx, dist, None, mtx)
    return undist
```

2.4 对棋盘图像应用失真校正

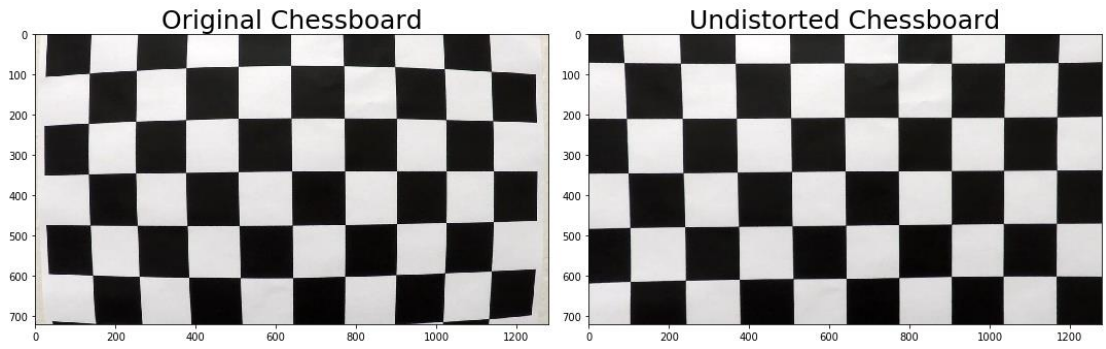

图 3- 1 棋盘失真校正

2.5 对道路图像应用失真校正

图 3- 2 图像失真校正

3 图像阈值

图像阈值包括从 HLS 色彩空间中提取 l_channel，从 LAB 色彩空间中提取 b_channel。方向阈值也会在颜色阈值后组合在一起，以突出显示水平方向线。

3.1 阈值镜像

图像阈值包括从 HLS 色彩空间中提取 l_channel，从 LAB 色彩空间中提取 b_channel。

```
# Image thresholding
def image_thresh(img, l_thresh=(180 , 255), b_thresh=(185, 255)):
    new_img = np.copy(img)

    # 1) Convert to HLS color space
    hls = cv2.cvtColor(new_img, cv2.COLOR_RGB2HLS)
    l_channel = hls[:, :, 1]
    l_channel = l_channel * (255 / np.max(l_channel))
    # 2) Apply a threshold to the L channel
```

```
    l_binary = np.zeros_like(l_channel)
    l_binary[(l_channel > l_thresh[0]) & (l_channel <= l_thresh[1])] = 1

    # 3) Convert to LAB color space
    lab = cv2.cvtColor(new_img, cv2.COLOR_RGB2LAB)
    b_channel = lab[:, :, 2]
    # don't normalize if there are no yellows in the image
    if np.max(b_channel) > 175:
        b_channel = b_channel * (255 / np.max(b_channel))
    # 4) Apply a threshold to the B channel
    b_binary = np.zeros_like(b_channel)
    b_binary[(b_channel > b_thresh[0]) & (b_channel <= b_thresh[1])] = 1

    # 5) Apply a Sobel x to the x
    sobelx = cv2.Sobel(cv2.cvtColor(img, cv2.COLOR_RGB2GRAY), cv2.CV_64F,
1, 0)  # Take the derivative in x at b_channel
    abs_sobelx = np.absolute(sobelx)  # Absolute x derivative to accentuate
lines away from horizontal
    sobely = cv2.Sobel(cv2.cvtColor(img, cv2.COLOR_RGB2GRAY), cv2.CV_64F,
0, 1)
    abs_sobely = np.absolute(sobely)  # Absolute x derivative to accentuate
lines away from horizontal



    dir_grad = np.arctan2(abs_sobely, abs_sobelx)
    dir_binary = np.zeros_like(dir_grad)
    dir_binary[(dir_grad > 0) & (dir_grad <= 0.1)] = 1

    # scaled_sobel = np.uint8(255 * abs_sobelx / np.max(abs_sobelx))
    # sx_binary = np.zeros_like(scaled_sobel)
    # sx_binary[(scaled_sobel > sx_thresh[0]) & (scaled_sobel <=
sx_thresh[1])] = 1

    combined_binary = np.zeros_like(b_binary)
    combined_binary[(l_binary == 1) | (b_binary == 1)] = 1
    # combined_binary[(l_binary == 1) | (b_binary == 1) | (dir_binary ==
1)] = 1

    return combined_binary
```

3.2 将图像阈值应用于未失真的道路图像

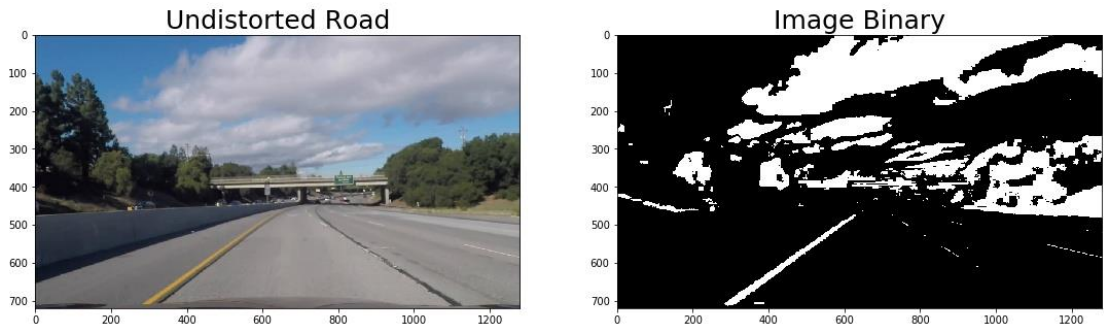

图 3- 3 图像阈值运用

4 透视转换

使用源点和目标点计算透视转换：

M = # 透视矩阵 `cv2.getPerspectiveTransform(src, dst)`

最小值 = # 逆透视矩阵 `cv2.getPerspectiveTransform(dst, src)`

扭曲 = # 变换后的“鸟瞰”视图 `cv2.warpPerspective(img, M, img_size)`

将透视变换应用于图像阈值道路图像

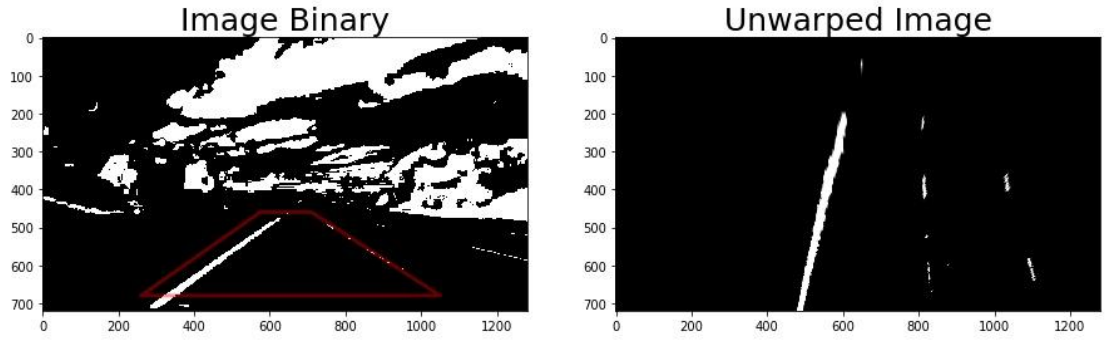

图 3- 4 应用透视变换

5 检测车道线

计算左/右多项式函数和未翘曲（鸟瞰图）图像的曲率半径

查找车道线的步骤如下：

1 制作图像下半部分的**直方图**。

2 找到直方图的左半部分和右半部分的**峰值**。它们是左线和右线的起点。

3 从 0 到图像高度。一旦检测到并安装了线条，就可以使用 跳过滑动窗口的步骤。`sliding_window_polyfitprev_fit_polyfit`

4 用二阶多项式**拟合**结果点。

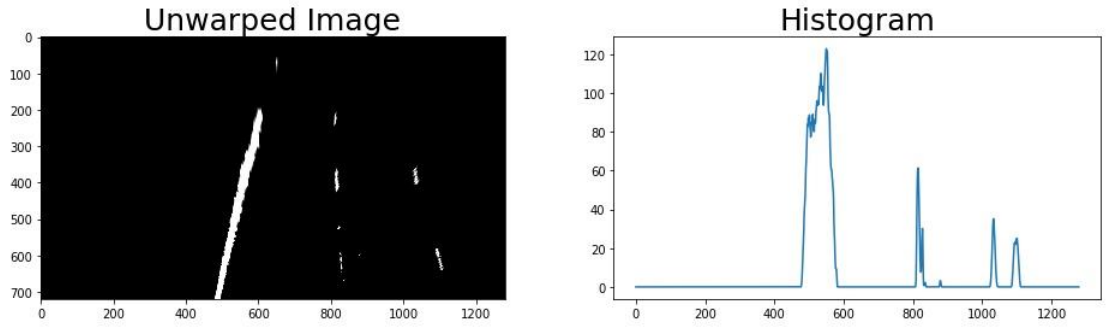

图 3- 5 直方图

5 使用从像素空间到米的 x 和 y 转换来计算曲率半径和交叉跟踪误差。`radius_curvature`

6 在未扭曲的图像上绘制结果。

7 使用逆透视**变换扭曲结果**图像。`Minv`

PS：在视频处理中，不需要搜索所有间隔，算法**会记住上一帧中的线位置**，并且只搜索该线位置附近。使用这种方法，可以大大缩短计算时间。

5.1 滑动窗口查找车道像素

1 定义 10 个窗口，窗口的高度是图像的高度除以窗口的数量。

2 循环浏览这些窗口：

标识 x 和 y（以及右侧和左侧）中的窗口边界。

标识图像中的非零像素及其索引。

标识窗口中 x 和 y 中的非零像素，并将其索引追加到每行。

如果我们发现>minpix 像素，重新定位下一个窗口，则新中心是像素位置的平均值。

3 提取左线和右线像素位置。

4 的输出如下所示。sliding_window_polyfit()

PS：一旦检测到并安装了线条，就可以跳过滑动窗口的步骤。

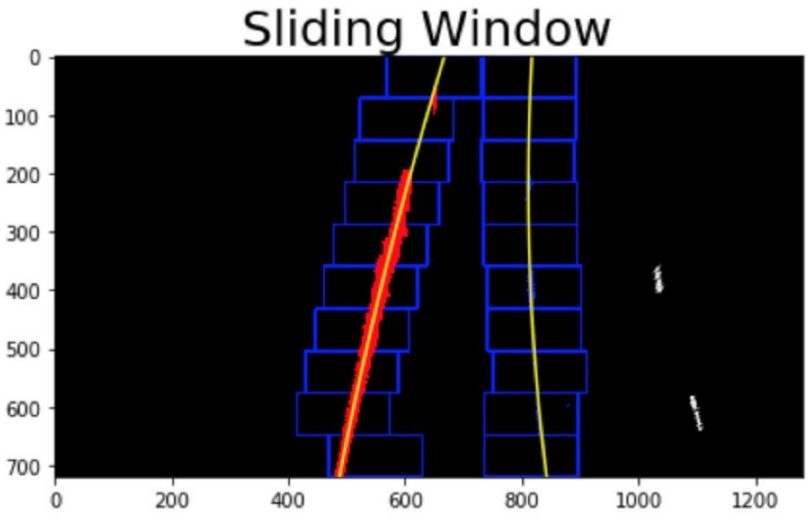

图 3- 6 滑动窗口

5.2 查找通道像素的上一个多项式

`1 prev_fit_polyfit` 函数执行基本相同的任务，但通过利用先前的拟合（例如，来自以前的视频帧）并仅搜索该拟合的特定范围内的通道像素，有效地降低了搜索过程的难度。

2 一旦找到安装的线，**在下一帧**视频中，就没有必要再次进行盲目搜索。相反，在前一**行位置**周围的**边距中搜索**可用于查找车道线。

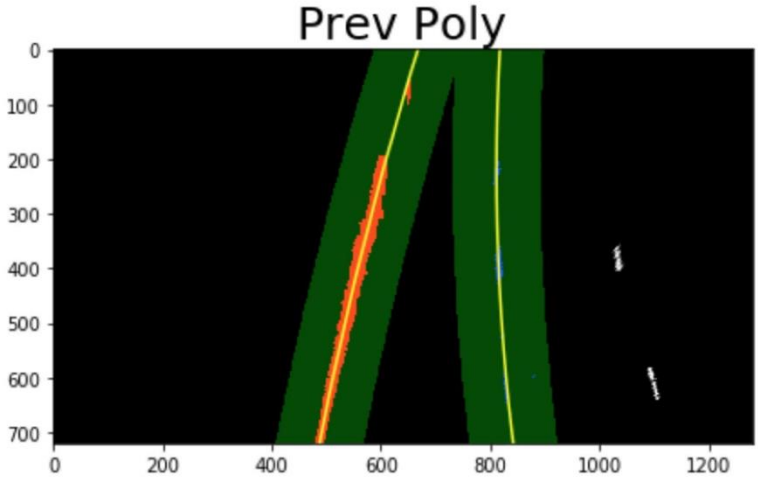

图 3- 7 多项式查找

5.3 车道曲率半径

1 确定 y/x 尺寸中每像素的米数。

2 将新多项式拟合到现实世界空间中的 x，y。

3 计算新的曲率半径，以米为单位。

4 根据车道中心计算车辆的位置。

例如：

```
left_cur, right_cur, ctb = radius_curvature(unwarped_road, left_fit_new,
right_fit_new, left_lane_inds2, right_lane_inds2)
print('radius of curvature: %.3f m, %.3f m' % (left_cur, right_cur))
print('radius of curvature: %.3f m' % ((left_cur + right_cur)/2))
print('center: %.3f m' % (ctb))
radius of curvature: 2186.001 m, 1305.509 m
radius of curvature: 1745.755 m
center: -0.133 m
```

6 在图像上绘图

到目前为止，我们有：

名为“road_imge”的原始道路图像

一个名为 unwarped_road 的未扭曲的二进制图像

具有多项式 left_fit 的拟合车道线，right_fit 表示线的 x 和 y 像素值。

定义一个函数，将这些线作为选定区域投影到原始图像上。`draw_lane`

定义一个函数，用于在图像上写入文本（曲率半径、中心偏置）。`draw_data`

6.1 在扭曲和未覆盖的道路图像中附加车道区域

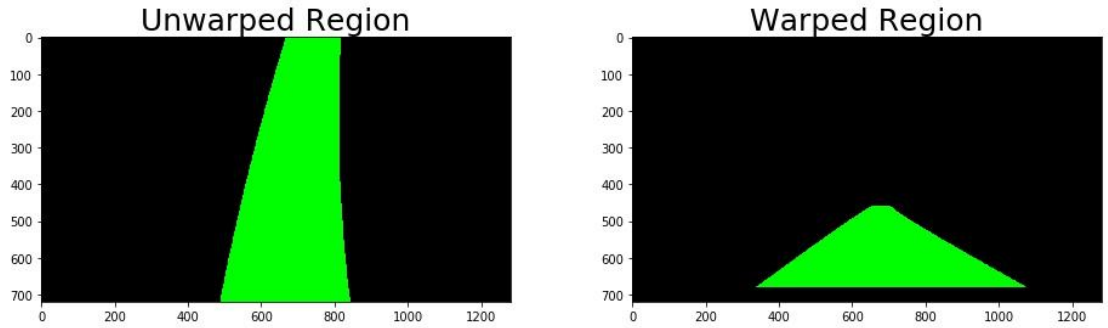

图 3- 8 附加车道区域

6.2 将检测到的车道绘制回原始图像

```
'''Draw onto the original image'''
drawed_img = draw_lane(road_img, unwarped_road, left_fit, right_fit, Minv)
rad_l, rad_r, center_bias = radius_curvature(unwarped_road, left_fit,
right_fit, left_lane_inds, right_lane_inds)
```

```
detected_lane = draw_data(drawed_img, (rad_l + rad_r)/2, center_bias)
```

图 3- 9 检测车道绘回原始图像

7 视频处理

定义图像处理管道，包括 、 、 、 、 。Camera calibrationImage thresholdingPerspective TransformDetect lane linesDrawing

将管道应用于给定的视频。

7.1 导入视频处理所需的模块

```
from moviepy.editor import VideoFileClip
from IPython.display import HTML
```

7.2 定义用于存储行信息的类 Line()

```
# Define a class to receive the characteristics of each line detection
class Line():
    def __init__(self):
        # was the line detected in the last iteration?
        self.detected = False
        # x values of the last n fits of the line
        self.recent_xfitted = []
        # average x values of the fitted line over the last n iterations
        self.bestx = None
        # polynomial coefficients averaged over the last n iterations
        self.best_fit = None
        # polynomial coefficients for the most recent fit
        self.current_fit = []
```

```
        # radius of curvature of the line in some units
        self.radius_of_curvature = None
        # distance in meters of vehicle center from the line
        self.line_base_pos = None
        # difference in fit coefficients between last and new fits
        self.diffs = np.array([0, 0, 0], dtype='float')
        # number of detected pixels
        self.px_count = None

    def add_fit(self, fit, inds):
        # add a found fit to the line, up to n
        n = 9
        if fit is not None:
            if self.best_fit is not None:
                # if we have a best fit, see how this new fit compares
                self.diffs = abs(fit - self.best_fit)
            if (self.diffs[0] > 0.001 or \
                self.diffs[1] > 1.0 or \
                self.diffs[2] > 100.) and \
                    len(self.current_fit) > 0:
                # bad fit! abort! abort! ... well, unless there are no fits
in the current_fit queue, then we'll take it
                self.detected = False
            else:
                self.detected = True
                self.px_count = np.count_nonzero(inds)
                self.current_fit.append(fit)
                if len(self.current_fit) > n:
                    # throw out old fits, keep newest n
                    self.current_fit = self.current_fit[len(self.current_fit)
- n:]
                self.best_fit = np.average(self.current_fit, axis=0)
        # or remove one from the history, if not found
        else:
            self.detected = False
            if len(self.current_fit) > 0:
                # throw out oldest fit
                self.current_fit = self.current_fit[:len(self.current_fit) -
1]
            if len(self.current_fit) > 0:
                # if there are still any fits in the queue, best_fit is
their average
                self.best_fit = np.average(self.current_fit, axis=0)
```

```
l_line = Line()
r_line = Line()
```

7.3 Define a pipeline `image_process()`

In order to integrate all needed functions to process images and then applied in video frames.

```
def image_process(img):
    new_img = np.copy(img)
    # ----------------------------------------
    # Image Undistortion
    # ----------------------------------------
    undistort = cal_undistort(new_img)

    # ----------------------------------------
    # Image Thresholding
    # ----------------------------------------
    img_binary = image_thresh(undistort, l_thresh=(180, 255),
b_thresh=(185,255))

    # ----------------------------------------
    # Perspective Transform
    # ----------------------------------------
    unwarped, _, Minv = perspective_tansform(img_binary)

    # ----------------------------------------
    # Lane Lines Finding
    # ----------------------------------------
    # if both left and right lines were detected last frame,
`polyfit_using_prev_fit` is going to be applied
    # otherwise `sliding window` is selected
    if not l_line.detected or not r_line.detected:
        l_fit, r_fit, l_lane_inds, r_lane_inds, _ =
sliding_window_polyfit(unwarped)
    else:
        l_fit, r_fit, l_lane_inds, r_lane_inds = prev_fit_polyfit(unwarped,
l_line.best_fit, r_line.best_fit)

    # invalidate both fits if the difference in their x-intercepts isn't
around 350 px (+/- 100 px)
    if l_fit is not None and r_fit is not None:
        # calculate x-intercept (bottom of image, x=image_height) for fits
        h = img.shape[0]
        l_fit_x_int = l_fit[0] * h ** 2 + l_fit[1] * h + l_fit[2]
```

```
        r_fit_x_int = r_fit[0] * h ** 2 + r_fit[1] * h + r_fit[2]
        x_int_diff = abs(r_fit_x_int - l_fit_x_int)
        if abs(350 - x_int_diff) > 100:
            l_fit = None
            r_fit = None

    l_line.add_fit(l_fit, l_lane_inds)
    r_line.add_fit(r_fit, r_lane_inds)

    # ----------------------------------------
    # Drawing
    # ----------------------------------------
    # Draw the lane & Write text onto the warped blank image

    if l_line.best_fit is not None and r_line.best_fit is not None:
        # draw the current best fit if it exists
        drawed_img = draw_lane(new_img, unwarped, l_line.best_fit,
r_line.best_fit, Minv)
        rad_l, rad_r, d_center = radius_curvature(unwarped, l_line.best_fit,
r_line.best_fit,
                                                  l_lane_inds,
r_lane_inds)
        result = draw_data(drawed_img, (rad_l + rad_r) / 2, d_center)
    else:
        result = new_img

    return result
```

7.4 在单个映像上进行测试

```
test_image = mpimg.imread('./test_images/challenge01.jpg')
result = image_process(test_image)
```

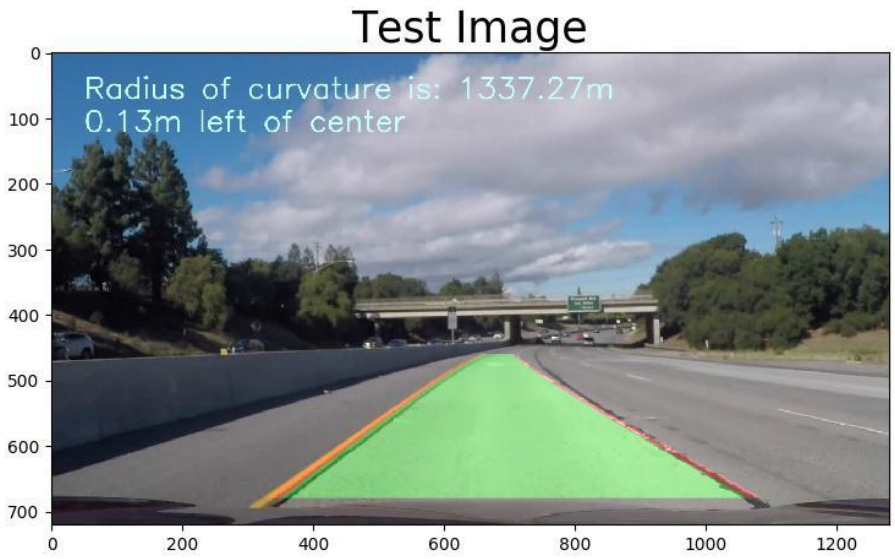

图 3- 10 测试图片

7.5 在 project_video 上测试（简单）

```
output = 'output_videos/project_output.mp4'
clip1 = VideoFileClip('project_video.mp4')
output_clip = clip1.fl_image(image_process)
output_clip.write_videofile(output, audio=False)
```

7.6 challenge_video 测试（硬）

```
challenge_output = "output_videos/challenge_output.mp4"
challenge_input = VideoFileClip("challenge_video.mp4")
video_process = challenge_input.fl_image(image_process)
video_process.write_videofile(challenge_output, audio=False)
```

四、结果讨论与对比分析

很难找到最佳阈值和超参数。

第一次转换透视时，src 点和 dest 点的顺序是不一样的，所以鸟瞰图实际上是上下朝下的，但它很难被注意到，它可以对结果产生很大的影响。

类可以通过计算最后 n（9 个沉降）发现的拟合的平均值的方法，帮助提高车道线检测的性能，从而可以平滑和稳定车道线的移动。`Line()`

五、结论与探讨

1 老师给的原始视频

Digital_Image_Processing/9.mp4 at main · sinary-sys/Digital_Image_Processing (github.com)

2 道路线提取后的视频

Digital_Image_Processing/99.mp4 at main · sinary-sys/Digital_Image_Processing (github.com)

参考文献

[1]彭望碌等. 遥感概论[M]. 北京:高等教育出版社, 2002.

[2]Vosselman G, de Knecht J. Road tracing by profile matching and Kaiman filtering[M]//Automatic Extraction of Man-Made Objects from Aerial and Space Images. Birkhäuser Basel, 1995: 265-274.

[3]汪夕明. 遥感影像道路提取方法研究与实现[D]. 北京:清华大学, 2011.

[4]周秋琳.顾及几何特征的数学形态学高分辨率遥感道路提取方法研究[D]. 长沙:中南大学, 2013.

[5] Hinz S, Baumgartner A. Automatic extraction of urban road networks from multi-view aerial imagery[J]. ISPRS Journal of Photogrammetry and Remote Sensing, 2003, 58(1): 83-98.

[6]Mena J B. State of the art on automatic road extraction for GIS update: a novel classification[J]. Pattern Recognition Letters, 2003, 24(16): 3037-3058.

[7]魏宇峰. 高分辨率遥感影像道路信息提取关键技术研究与实现[D]. 北京:北京理工大学, 2010.

[8]顾丹丹. 基于活动轮廓模型的高分辨率遥感影像道路提取[D]. 西安:陕西师范大学, 2010.

[9] 叶发茂, 苏林, 李树楷, 汤江龙. 高分辨率遥感影像提取道路的方法综述与思考[J]. 国土资源遥感, 2006,18(01): 12-17.

[10] 林宗坚, 刘政荣. 从遥感影像提取道路信息的方法评述[J]. 武汉大学学报:信息科学版, 2003, 28(1): 90-93.

[11] Li X, Zhang S, Pan X, et al. Straight road edge detection from high-resolution remote sensing

images based on the ridgelet transform with the revised parallel-beam Radon transform[J].

International Journal of Remote Sensing, 2010, 31(19): 5041-5059.

[12] Li Y, Xu L, Piao H. Semi-automatic road extraction from high-resolution remote sensing

image: Review and prospects[C]//Hybrid Intelligent Systems, 2009.

HIS'09. Ninth

International Conference on. IEEE, 2009, 1: 204-209.

[13] 王培法, 王丽, 冯学智, 肖鹏峰. 遥感图像道路信息提取方法研究进展[J]. 遥感技术与应用.2009, 24(3):284-288.

[14] Heipke C, Englisch A, Speer T, et al. Semiautomatic extraction of roads from aerial images[C]//Spatial Information from Digital Photogrammetry and Computer Vision: ISPRS Commission III Symposium. International Society for Optics and Photonics, 1994: 353-360.

[15] Shukla V, Chandrakanth R, Ramachandran R. Semi-Automatic Road Extraction Algorithm for High Resolution Images Using Path Following Approach[C]//ICVGIP. 2002, 2: 231-236.

[16] Vosselman G., Knecht de j. Road Tracing by Profile Matching and Kalman Filtering. Automatic Extraction of manmade Objects from Aerial and Space images[C]. Birkhauser Verlag Basel, 1995, 265-274.

[17] 胡翔云, 张祖勋. 航空影像上线状地物的半自动提取[J]. 中国图象图形学报: A 辑, 2002, 7(2): 137-140.

[18] Baumgartner A., Steger C., et al. Automatic Road Extraction in Rural Areas. International Archives of Photogrammetry and Remote Sensing [C], 1999, Part3-2W5, 107-112.

[19] Mayer H, Steger C. A new approach for line extraction and its integration in a multi-scale, multi-abstraction-level road extraction system[M]. na, 1996.

[20] Trinder J C., Wang Y D., Sowmya A., et al. Artificial Intelligence in 3D Feature Extraction [A]. Automatic Extraction of Man-made Objects from Aerial and Space Images (2) [C]. Basel: Birkhaeuser Verlag, 1997. 257-265.

[21] Ton J., Jain A K., Enslin W R., et al. Automatic Road Identification and Labeling in Landsat 4 TM Images [J]. Photogrammetric [PRS], 1989, 43(2): 257-276.

[22] Mayer H, Laptev I, Baumgartner A, et al. Automatic road extraction based on multi-scale modeling, context, and snakes[J]. International Archives of Photogrammetry and Remote Sensing, 1997, 32(Part 3): 106-113.

[23] Baumgartner A, Eckstein W, Heipke C, et al. T-REX: TUM research on road

extraction[J]. Festschrift für Prof. Dr.-Ing. Heinrich Ebner zum 60. Geburtstag, 1999: 43-64.

[24] Rellier G, Descombes X, Zerubia J. Local registration and deformation of a road cartographic database on a SPOT satellite image[J]. Pattern Recognition, 2002, 35(10): 2213-2221.

[25] Monga O, Armande N, Montesinos P. Thin nets and crest lines: Application to satellite data and medical images[C]//Image Processing, 1995. Proceedings., International Conference on. IEEE, 1995, 2: 468-471.

[26] Price K. Road grid extraction and verification[C]//IAPRS. 1999, 32(Part3-2W5): 101-106.

[27] Baumgartner A, Steger C, Maye H, et al. Automatic road extraction based on multi-scale, grouping, and context[J]. Photogrammetric Engineering and Remote Sensing, 1999, 65: 777-786.

[28] 安如, 冯学智, 王慧麟. 基于数学形态学的道路遥感影像特征提取及网络分析[J]. 中国图象图形学报: A 辑, 2004, 8(7): 798-804.

[29] 刘珠妹, 刘亚岚, 谭衢霖, 等. 高分辨率卫星影像车辆检测研究进展遥感技术与应用, 2012, 27(1): 8-14.

[30] Mark McCord, Prem Goel, Zhuojun Jiang, Benjamin Coifman, Yongliang Yang, Carolyn Merry. IMPROVING AADT AND VDT ESTIMATION WITH HIGH-RESOLUTION SATELLITE IMAGERY[R]. Pecora 15/Land Satellite Information IV/ISPRS Commission I/FIEOS 2002 Conference Proceeding, 2002.

[31] Liu H X, Wu X, Ma W, et al. Real-time queue length estimation for congested signalized intersections[J]. Transportation research part C: emerging technologies, 2009, 17(4): 412-427.

[32] 徐春. 基于遥感图像的交通信息提取研究与实现[D]. 北京: 清华大学, 2008.

[33] 刘建鑫. 基于高分辨率卫星影像车辆检测算法研究[D]. 大连: 大连海事大学, 2008.

[34] 张新野. 基于形态学的卫星遥感车辆模式识别方法研究[D]. 大连：大连海

事大学, 2008.

[35] Eikvil L, Aurdal L, Koren H. Classification-based Vehicle Detection in High Resolution Satellite Images [J]. ISPRS Journal of Photogrammetry and Remote Sensing, 2009, 64:65-72.