一种基于深度学习的单目测距实现

18364067 罗淦元
(中山大学智能工程学院 智能科学与技术专业 518000)

摘　要： 基于单摄像头的道路测距是高级驾驶辅助系统(ADAS)的主要功能之一，传统的单目测距方法主要使用 OpenCV 的图像处理方法实现。本文主要基于图像分割的神经网络，探究鲁棒性更强的道路测距算法。

关键词： ADAS; 单目测距; 神经网络

0 引言

传统的道路单目测距算法主要通过将图像阈值化，并通过 Canny 算子等进行边缘提取，利用霍夫线变换得到道路的车道信息。然后利用图像特征，如车牌、阴影等，使用图像处理方法提取疑似车辆的区域。基于这些信息和标定过的摄像头内参，估测前车车距[1]。这种做法具有鲁棒性不强的明显缺点，环境的光影、车辆对行车道的遮挡、天气等都会对单目测距过程造成严重干扰。训练良好的神经网络则具有鲁棒性良好的特点。本研究通过现有的神经网络识别和分割方法，探究基于神经网络的道路单目测距方法。

1 摄像机标定

1.1 摄像机标定背景

由于拍摄设备的镜头在生产和组装中畸变程度各异。对三维物体在二维照片上的映射关系求解造成了很大干扰。因此作为初期的数据处理，需要对摄像机进行内参标定，从而在后续处理中对获取图像进行矫正。内参主要由像素物理尺寸 dx、dy、焦距 f、扭曲因子 r、图像原点相较于光心成像点偏移量 u、v 组成。其中有 fx、fy 为摄像机焦距的像素表示，即 fx = f/dx,fy=f/dy。在小孔成像的模型下，fx 和 fy 近似相等。一般扭曲因子为 0，内参矩阵为

$$mtx = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

外参则由旋转矩阵 R 和平移矩阵 T 组成，为

$$\begin{bmatrix} R & T \\ 0^T & 1 \end{bmatrix}$$

用于将世界坐标系转换为相机坐标系。本实验中假定了相机外参以完成单目测距。而畸变系数由径向畸变 ki 和切向畸变 pi 组成，有

$$dist = [K1,\ K2,\ P1,\ P2,\ K3]$$

通过 openCV 能够将图像进行反畸变操作来修正摄像头畸变。

最后，我们可以得到世界坐标系到像素坐标系的映射关系有

$$Z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} * [R|T] * \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

1.2 张正友棋盘格标定法

本研究使用了常见的张正友棋盘格标定法，多角度拍摄一张每个小方格为 2cm 宽的棋盘格。如果使用设备为手机需要注意两点。一是多摄像头设备在使用录像功能和拍照功能时，可能会使用不同的摄像头。为了控制变量，最好手动调节使用的摄像头，或是遮蔽掉特定摄像头。

另外，如果如本例使用手机作为防抖手段，还需要注意手机是否搭载了 EIS 电子防抖技术。EIS 防抖技术通过对摄像模组进行位姿估计以动态裁切视频画面，从而达到使画面更稳定的目的。为了控制变量，本例中使用谷歌相机并关闭了视频防抖功能，同时使用 1080p/24fps 模式进行道路画面录制以及棋盘格标定数据采集，并从后者视频中抽取清晰的帧完成标定。

为了标定时更清晰准确提取角点，进行标定时，实验使用了屏幕可视角度较大的一款平板电脑完成角点标定，通过在全黑环境下多角度录制平板电脑上显示的棋盘格，并从中提取清晰帧，取得了 30 张类似如图 1 所示图片的标定数据。

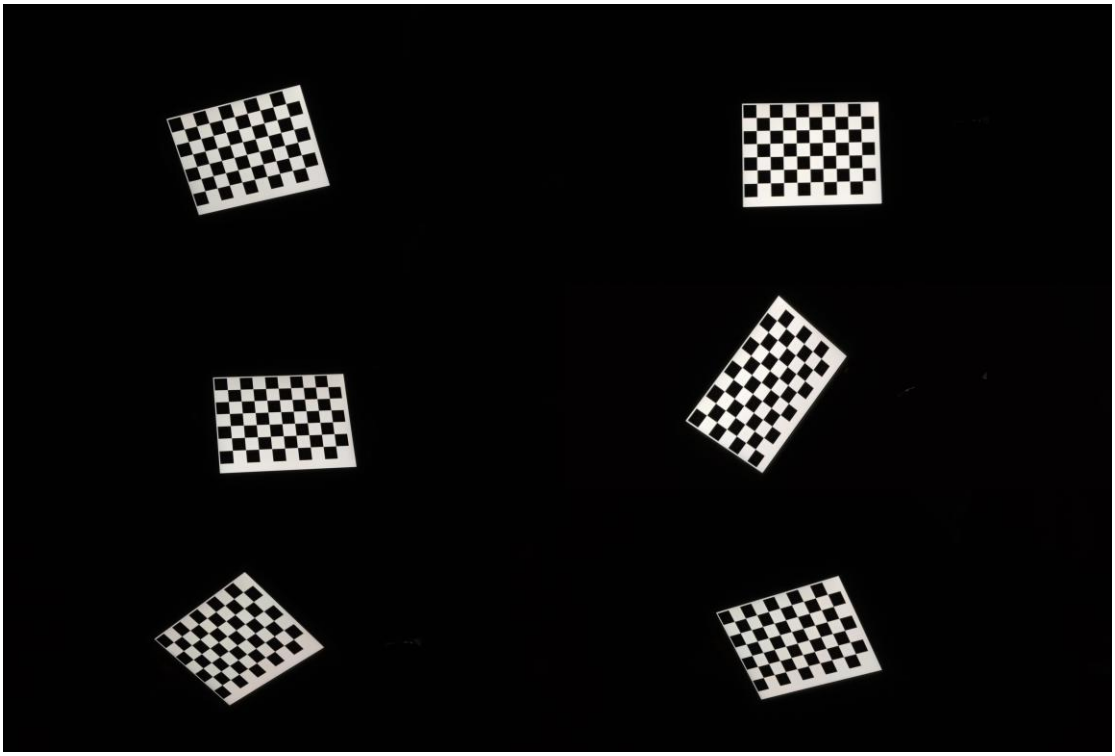

图 1 棋盘格标定部分样例

需要注意的是，棋盘格和摄像机角度过大时会导致无法识别棋盘格角点，为了提高标定数据的有效性，选取数据时要避免选取倾角过大的极端数据。

通过 openCV 库中的 cv2.findChessboardCorner()函数提取棋盘格角点。

```
for fname in tqdm(images):
    img = cv2.imread(fname)
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    # 查找棋盘格角点
    ret, corners = cv2.findChessboardCorners(gray, (w,h),None)
```

```
    # 如果找到了目标点组，则存储其世界坐标和像素坐标
    if ret == True:
        cv2.cornerSubPix(gray,corners,(11,11),(-1,-1),criteria)
        objpoints.append(objp)
        imgpoints.append(corners)
        ii+=1
```

得到的结果如图 2 所示。

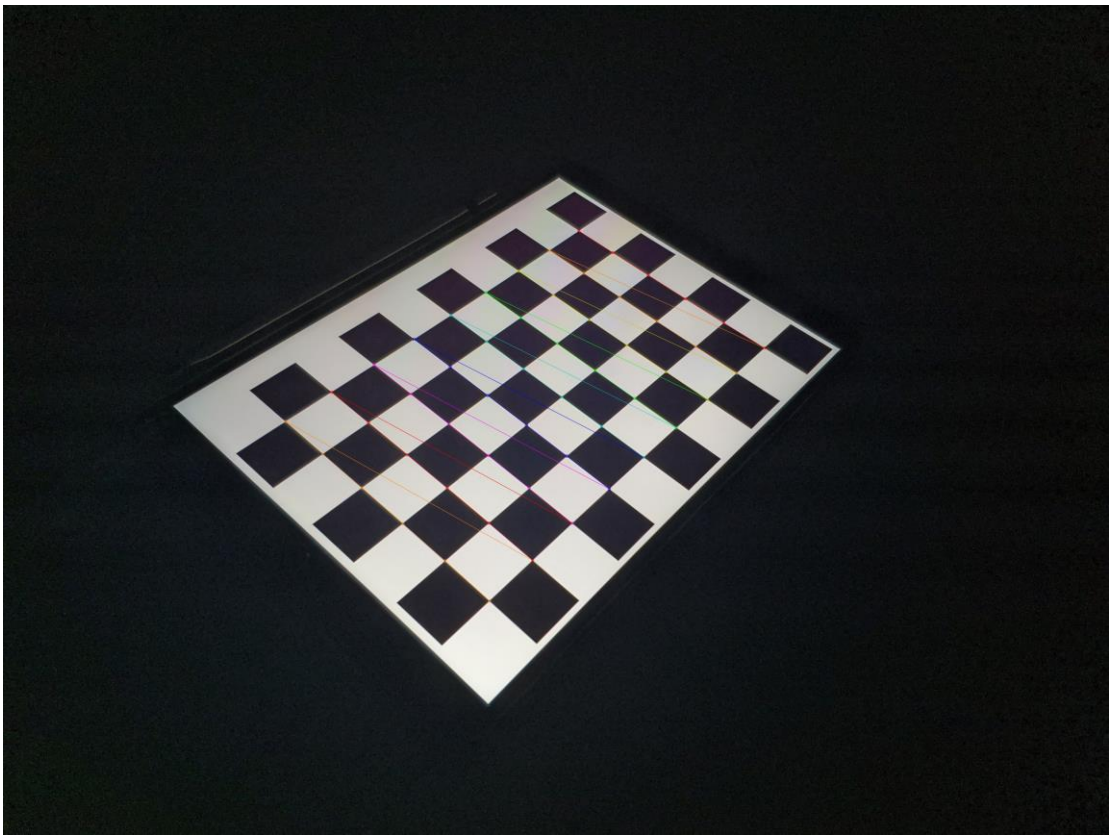

图 2 提取棋盘格角点

然后利用 openCV 的内置相机标定函数 cv2.calibrateCamera(),就能够标定相机内参数矩阵。

考虑到真实使用环境下，相机的内参标定不由用户完成，因此本实验单独设计了一段程序用于给出内参参数，程序另附于附件中。得到的内参矩阵如下。

$$mtx = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 3127.5753 & 0 & 1026.48203e \\ 0 & 3127.44985 & 506.658577e \\ 0 & 0 & 1 \end{bmatrix}$$

得到的畸变系数如下。

$$dist = [[0.39769363 \quad -3.15565604 \quad -0.01308121 \quad 0.01180472 \quad 4.92799998]]$$

我们可以通过反畸变和内参矩阵，将图像中的目标从像素坐标系转换到相机坐标系中，进而利用相似三角形测距。具体方法为

```
img_undist = cv2.undistort(img, mtx, dist, None, newcameramtx)
```

效果如图 3 所示

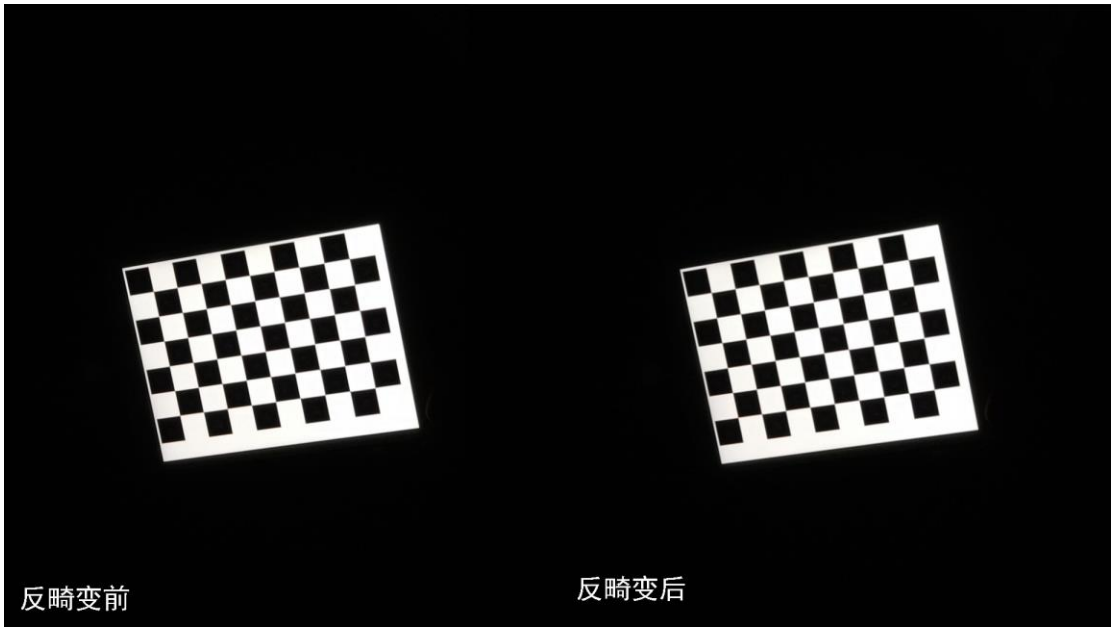

图 3 反畸变效果

可以看到，对于手机摄像头而言，畸变程度并不大。

2 道路物体检测

2.1 目标检测神经网络

在本研究中引用了著名的开源神经网络 YOLOv3[2]。YOLOv3 主要取了 darknet-53 的前 52 层，如图 4 所示。由于使用了全卷积层的结构，YOLOv3 有着非常好的运算性能。结合残差网络结构形成了较深的网络层次，从而大幅提高了较小物体的识别能力。同时基于输出特征图数量和尺度变化，使用 K-means 聚类方法获得先验框尺寸。最后基于 9 种先验框得到对象的 bounding box。

同时相较于 YOLOv2，YOLOv3 不再使用 softmax，而是该用 logistic 对输出进行预测，使得网络支持多标签对象。

	Type	Filters	Size		Output	
	Convolutional	32	3	3	256	256
	Convolutional	64	3	3 / 2	128	128
	Convolutional	32	1	1		
1	Convolutional	64	3	3		
	Residual				128	128
	Convolutional	128	3	3 / 2	64	64
	Convolutional	64	1	1		
2	Convolutional	128	3	3		
	Residual				64	64
	Convolutional	256	3	3 / 2	32	32
	Convolutional	128	1	1		
8	Convolutional	256	3	3		
	Residual				32	32
	Convolutional	512	3	3 / 2	16	16
	Convolutional	256	1	1		
8	Convolutional	512	3	3		
	Residual				16	16
	Convolutional	1024	3	3 / 2	8	8
	Convolutional	512	1	1		
4	Convolutional	1024	3	3		
	Residual				8	8
	Avgpool		Global			
	Connected		1000			
	Softmax					

图 4 53-Darknet 结构

可以看到，YOLOv3 还直接抛弃了 pooling 层这种节省算力的下采样方法，而是使用 stride 较大的大卷积核实现下采样，进一步提高了模型对特征的描述能力。

在很多数据集上，YOLOv3 都有着非常良好的性价比，能够以数分之一的性能消耗得到和复杂网络类似的精度水平。考虑到本研究未来搭载的平台可能具有一定的算力限制，且需要使用另一个网络实现车道线检测，故使用了进一步缩减特征层的 YOLOv3-Tiny 网络。该网络能够在功耗 8W 的搭载 MX150 设备上以每秒数帧的速率完成目标检测，能够很好地满足车距检测需求。而在如图 5 所示的我们采集的道路数据上，即使光照条件苛刻，其也能表现出良好的目标检测性能。

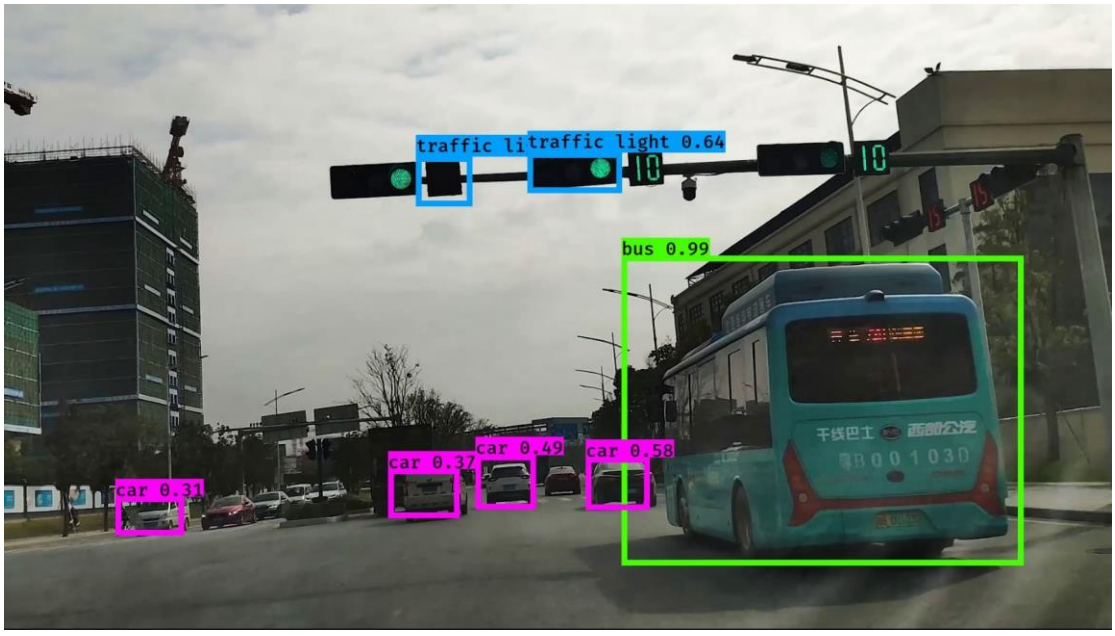

图 5 YOLOv3 Tiny 实际道路测试

最后我们计划使用 bdd100k 数据集对 YOLOv3 Tiny 模型进行细微的调整，使 YOLOv3 能够在特定的道路环境下取得更好的目标识别表现。但调参过程并不顺利，因此进一步的改进将作为未来的课题。

2.2 车道线检测

作为车道线检测的实现，本研究首先参考了一些基于 OpenCV 方法实现的车道检测方法[34]。主要通过对图片进行高斯滤波降噪、转换为灰度图、使用 Canny 算子提取图像边缘信息、划定兴趣区域及利用霍夫变换提取车道线端点得到。其在自带的测试图片，如图 6 上获得了很好的表现，但是当我们将其作用于自采集和 CULane 的测试数据上，如图 7 时，其表现效果非常差。

图 6 来自项目自有测试数据的车道检测结果

图 7 自采集数据和 CULane 数据集上的 openCV 测试样例

在部分数据上，OpenCV 方法在不调参数的前提下，辨识车道线严重偏离。在更多的数据上则干脆无法识别出任何一条车道线。

根据原理分析和项目叙述，可以判断这是由于 OpenCV 阈值的设置较为苛刻，对于不同环境需要单独测试和调整参数才能够正确识别车道范围，这显然在鲁棒性上需要提高。因此联想到了神经网络。

仔细阅读分析了一些论文后，我们选择了在 ECCV2020 上发布的一种快速行车道检测网络[5]。深度学习在图像分割上的实际上也一直存在很大的局限性。一方面是为了对车道进行分割，需要进行大量高复杂度的的特征提取和非线性映射，在高分辨率数据的基础上产生了非常巨大的运算量；另一方面是分割时采用的卷积核大小不能够很大，使得网络只具有非常局限的感受野区域。而在对于车道尤其是弯道等非直线车道的检测上，大的感受野是非常必要的。而本车道线检测算法的分割原理如图 8 所示。

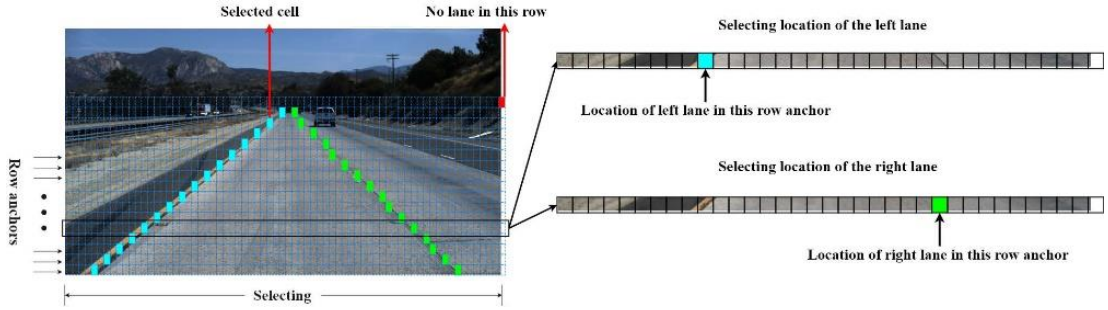

图 8 Ultra Fast Structure-aware Deep Lane Detection

该网络首先在图片上不同行设置了多个锚点，每个锚点上，将附近一定行数内的像素划分为多个单元格，而车道线检测被细分为在图片上选择包含车道线的单元格操作。从而将需要进行的操作数量大幅度减少，保证了其在准确率良好的基础上，能够在专业运算卡上达到 300FPS 以上的检测速率。

然后，需要解决的是当车道线在部分范围，因为空隙和遮挡产生的没有信息的情况。本方法使用了一些特殊的损失函数方法来解决这一问题。首先是相似度损失函数，被定义为

$$L_{similarity} = \sum_{i=1}^{C}\sum_{j=1}^{h-1} ||P_{ij} - P_{i,j+1}||_1$$

其描述了图片中相邻锚点行上的分布连续性，将相邻行上分类的 L1 范数定义为车道线平滑性，也就是希望车道线位置在相邻行上时平滑变换的。

第二点是形状损失函数，使用二阶差分方程约束了车道形状。基于车道线大部分是直的，而弯曲车道线在透视效果下也近乎是直线的假设，希望预测的车道线尽可能是直线。损失定义为

$$L_{shape} = \sum_{i=1}^{C}\sum_{j=1}^{h-2} ||(Loc_{ij} - Loc_{i(j+1)}) - (Loc_{i(j+1)} - Loc_{i(j+2)})||_1$$

最终损失则定义为上述损失和预测损失的加权和。经过一些参数和接口的调整，通过最小二乘法拟合得到的车道识别检测相 openCV 方法来说有着非常好的准确性和优秀的鲁棒性。如图 9 所示。

图 9 自采集数据和 CULane 数据集上的神经网络测试样例

3 视频流处理

对于视频流的处理，实验使用了 Image 和 openCV 库。由于模型使用的图片格式不同，因此需要在处理的过程中对格式进行转换。首先通过 openCV 库从视频中获取一帧，并转换为 PIL 库格式。需要注意的是，OpenCV 读取的图像是 BGR 格式，需要使用 cv2.cvtColor()函数将其转换为 RGB，再封装为 Image 库图片。然后将其喂给 YOLOv3 获取所有目标的 bounding box，并存储在 numpy 数组 out_j 中。重新将原图输入到行车道检测网络，获取车道识别点。然后在图片上分别绘制车道检测信息和 bounding box 及标签。

由于行车道检测网络将给出多个车道分割点坐标，为了在后续的距离估计时给出车道线，实验定义了一个 OLS 最小二乘法函数用于求解车道点组的拟合直线作为车道线标识。

```
def OLS(Xi, Yi):
    def func(p, x):
        k, b = p
        return k*x+b

    def error(p, x, y):
        return (func(p, x)-y)**2

    p0 = [1, 10]
    Para=leastsq(error,p0,args=(Xi,Yi))
    return Para[0]
```

对于 bounding box 和车道线的绘制，则分别调用了 PIL 库的 ImageDraw 和 openCV 的 line 和 circle 函数。

在较轻量级的神经网络选择下，本系统能够在 8W 功耗的 Mx150 GPU 上以约每秒五帧的速度对 1080p 视频流进行处理，在专用的 RTX2080ti 计算平台上，能够满足 24 帧以上的实时视频流运算。

4 车距估计

本实验使用的车距估计方法是基于参考车道宽度和相似三角形的估计方法。首先，手机摄像头的 x 和 y 方向焦距近乎相同，可以近似为针孔相机模型。如图 10 所示

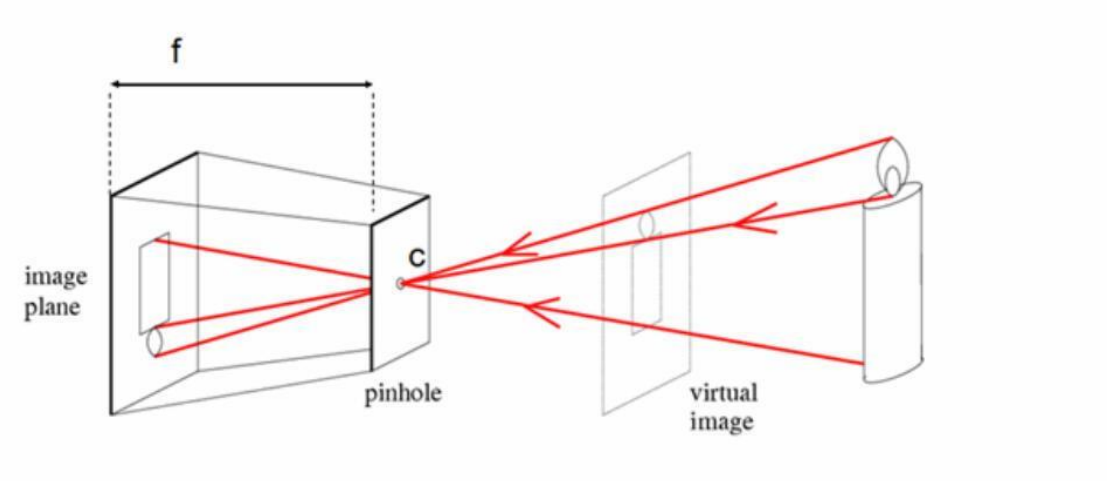

图 10 针孔成像模型

可以看到，成像物体和其在成像平面的像的比例与物体距离和焦距的比例有关。因此在实际道路上，我们能够通过车道线估测标准路面宽度在图像中所占像素，进而推测该路面位置到摄像机的

距离。如图 11 所示。

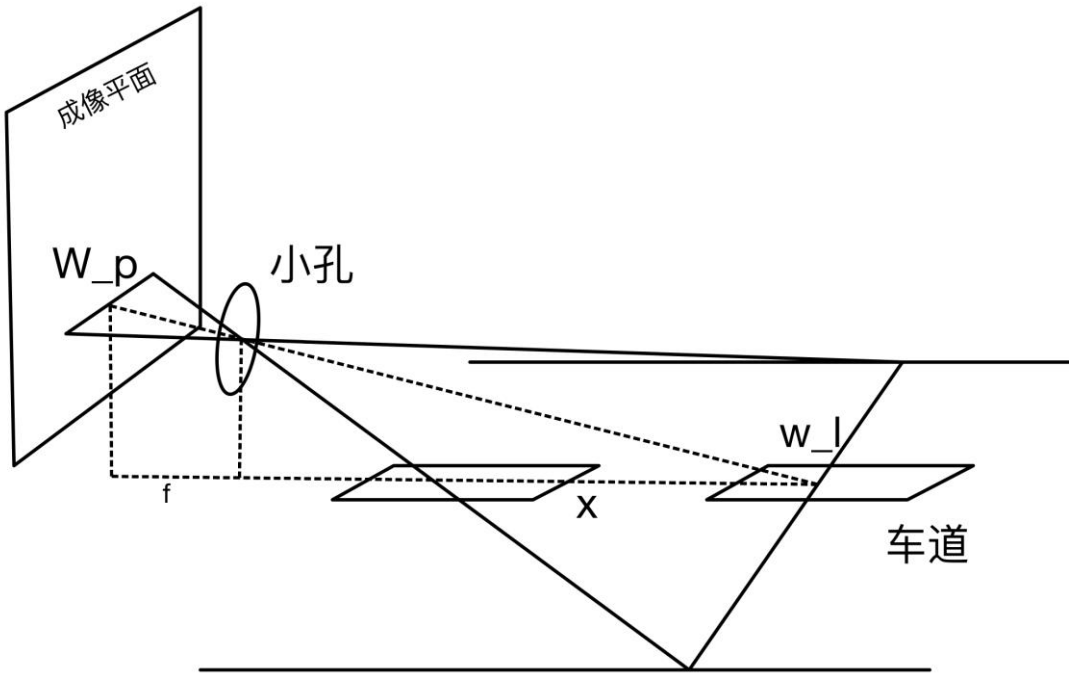

图 11 车道与成像的相似三角形关系

进而有

$$\frac{W_p}{W_l} = \frac{f}{x}$$

而根据内参矩阵 fx 的定义有

$$f_x = \frac{f}{dx}$$

联立得

$$x = \frac{f_x W_l}{n}$$

因而在直到内参 fx 的前提下，可以通过假设 W_l 得到前车的估计距离。具体而言，基于前任务获得的车道线拟合直线和 bounding box 的底部坐标，我们能够得到前车成像底部车道的像素数量，从而代入公式估计前车距离。至此，实验完成了一种基于神经网络的单目视觉前车测距方法。

5 实例测试

本实验使用了一段于中山大学深圳校区附近拍摄的测试视频。该视频包含了有交通信号灯和不同车辆的道路环境。得到如附件的测试视频。一部分截取画面如图 12 所示。

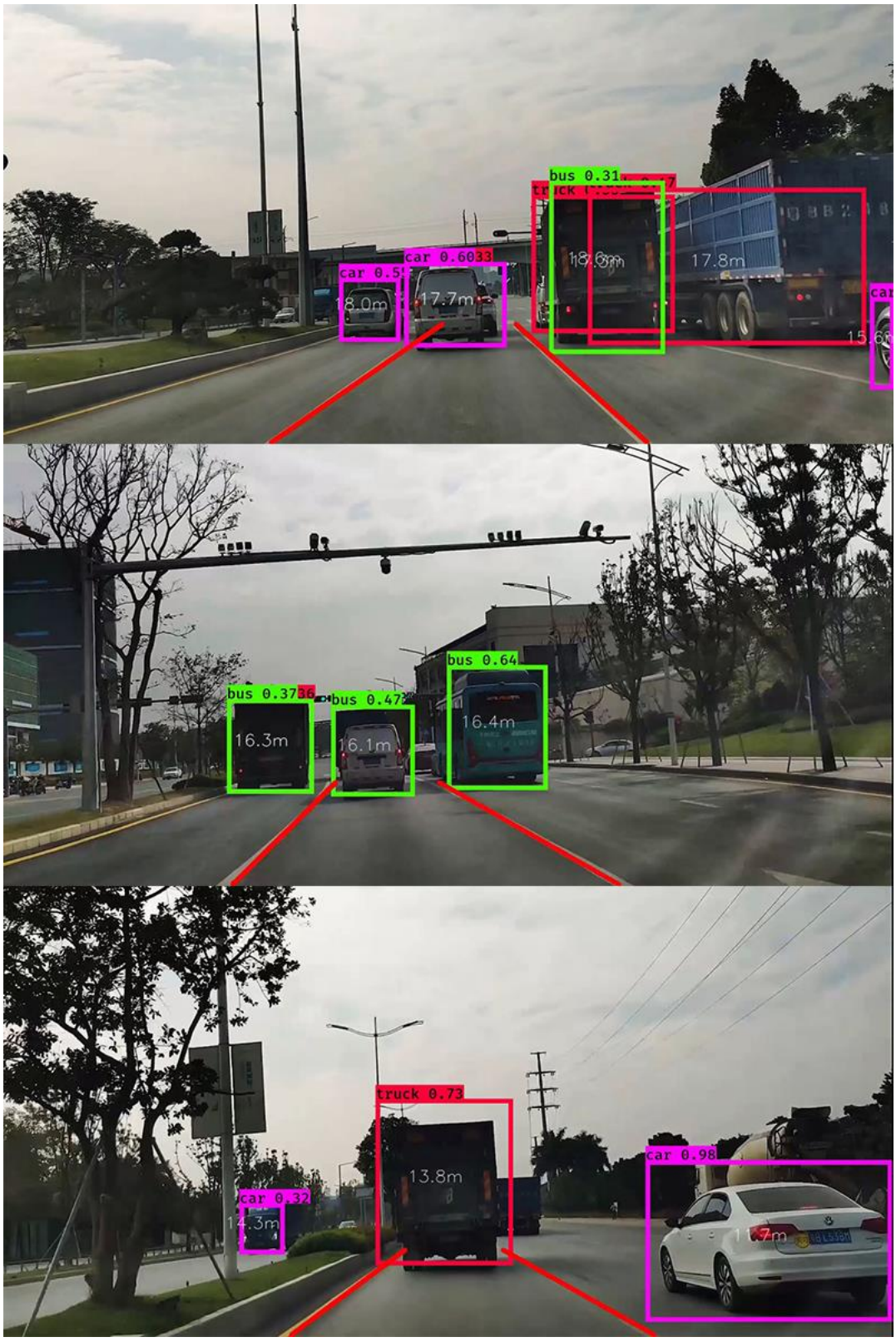

图 11 实例测试片段

6 总结与展望

本实验通过张正友棋盘标定法对摄像机内参进行标定，并基于神经网络进行车道识别和道路物

体检测，参照路面成像的几何关系，从单摄像机成像中衍生出当前车道的前车距离信息。本实验可以进一步改进的地方有：

1. 通过 bdd100k 等数据集进一步训练 YOLOv3 Tiny 和车道线检测神经网络，进一步提高检测的精度和鲁棒性
2. 在测量前车距离时，进一步考虑车辆与成像面中心的夹角以提高估计精度，以及考虑结合假定摄像机水平的估计方法，综合降低测量误差。

[1] 鲁威威,肖志涛,雷美琳.基于单目视觉的前方车辆检测与测距方法研究[J].电视技术,2011,35(01):125-128.

[2] YOLOv3: An Incremental Improvement Joseph Redmon, Ali Farhadi University of Washington

[3] https://github.com/yanjanoo/Lane_Deteciton_Basic

[4] https://github.com/feixia586/zhihu_material/tree/master/car_lane_detection

[5] Ultra Fast Structure-aware Deep Lane Detection, Zequn Qin, Huanyu Wang, and Xi Li, College of Computer Science & Technology, Zhejiang University, Hangzhou, China