

# 中山大学硕士学位论文

基于流数据的三维物体检测与追踪  
3D Streaming-based Object Detection and Tracking

学 位 申 请 人 郭叙森  
指 导 老 师 黄凯 教授  
专 业 名 称 计算机科学与技术

答 辩 委 员 会 主 席 (签 名): \_\_\_\_\_

答 辩 委 员 会 委 员 (签 名): \_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

二〇二〇 年 五 月 二十 日

## 论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究作出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名: \_\_\_\_\_

日期:       年   月   日

## 学位论文使用授权声明

本人完全了解中山大学有关保留、使用学位论文的规定，即：学校有权保留学位论文并向国家主管部门或其指定机构送交论文的电子版和纸质版；有权将学位论文用于非赢利目的的少量复制并允许论文进入学校图书馆、院系资料室被查阅；有权将学位论文的内容编入有关数据库进行检索；可以采用复印、缩印或其他方法保存学位论文；可以为建立了馆际合作关系的兄弟高校用户提供文献传递服务和交换服务。

保密论文保密期满后，适用本声明。

学位论文作者签名:                      导师签名:

日期:    年   月   日    日期:    年   月   日

论文题目 : 基于流数据的三维物体检测与追踪

专业 : 计算机科学与技术

硕士生 : 郭叙森

指导老师 : 黄凯 教授

## 摘要

近年来，随着深度学习技术的不断发展，三维物体检测领域取得了许多令人瞩目的成果，这些研究成果带动着众多自动驾驶创业公司蓬勃发展。然而，目前大多数三维物体检测方法都是针对单帧数据的，这些方法忽略了帧数据的时间连续性，无法利用帧与帧之间的时序信息，对检测算法的落地增加了不少难度。本工作旨在探索三维物体检测中时序信息的利用问题，通过深度学习技术去挖掘流数据的帧间相关性。为此，本文针对性地构建了一个基于关键帧的双路三维物体检测网络。该网络首先对关键帧进行物体检测，然后基于神经网络学习到的时序信息，将关键帧的检测结果传播到非关键帧。该传播过程可以将不同帧的同一物体关联，因此在得到检测结果之后，追踪结果也很容易获得，从而使得该网络同时具备三维物体检测与多目标追踪的能力。实验表明，本方法在速度与精度上都要优于同类型的单帧物体检测网络。在 KITTI 数据集的目标追踪比赛中，本方法在三维多目标追踪任务中取得了和当前最好方法相当的性能，分别取得了 76.68% 的 MOTA 和 81.65% 的 MOTP。

**关键词：**三维物体检测；多目标追踪；时序信息；关键帧

Title: 3D Streaming-based Object Detection and Tracking

Major: Computer Science and Technology

Name: Guo Xusen

Supervisor: Prof. Huang Kai

## ABSTRACT

Recent approaches for 3D object detection have made tremendous progresses due to the development of deep learning. However, previous researches for detection are mostly based on individual frames, leading to limited exploitation of information between frames. In this paper, we attempt to leverage the temporal information in streaming data and explore 3D streaming based object detection as well as tracking. Toward this goal, we set up a dual-way network for 3D object detection based on key frames, then propagate predictions to non-key frames through a motion based interpolation algorithm guided by temporal information. Our framework is not only shown to have significant improvements on object detection compared with frame-by-frame paradigm, but also proven to produce competitive results on KITTI Object Tracking Benchmark, with 76.68% in MOTA and 81.65% in MOTP respectively.

**Keywords:** 3D object detection; Multiple object tracking; Temporal information; Key frame

# 目 录

第 1 章 引言 ······	1
1.1 研究背景与意义 ······	1
1.2 国内外研究进展 ······	2
1.3 本文工作及创新点 ······	5
1.4 文章组织与结构 ······	6
第 2 章 目标检测与跟踪技术介绍 ······	7
2.1 目标检测 ······	7
2.2 目标跟踪 ······	14
2.3 本章总结 ······	17
第 3 章 流数据的三维目标检测与跟踪 ······	18
3.1 框架整体结构 ······	18
3.2 三维物体检测模块 ······	19
3.3 Shared RPN 模块 ······	23
3.4 时序信息处理模块 ······	25
3.5 运动插值模块 ······	27
3.6 多目标追踪 ······	30
3.7 本章总结 ······	30
第 4 章 实验过程与结果分析 ······	31
4.1 KITTI 数据集介绍 ······	31
4.2 数据预处理 ······	33
4.3 模型训练 ······	35
4.4 实验结果分析 ······	36
4.5 结果展示 ······	41
4.6 本章总结 ······	46
第 5 章 总结与展望 ······	47
5.1 全文总结 ······	47
5.2 展望 ······	47

参考文献 .....	49
攻读硕士学位期间发表学术论文情况.....	52
致 谢 .....	53

# 第1章 引言

## 1.1 研究背景与意义

近几年来，深度学习的迅猛发展赋予了机器越来越强的智能。在这样的背景下，无人驾驶作为最能够体现机器智能的“潜力股”之一，受到了各行各业的普遍关注。无人驾驶要求车辆能够像人类一样识别道路场景，包括行人，车辆以及车道线等，并且能够根据这些信息高效地规划路径来实现与人类类似的驾驶行为。在无人驾驶的整个技术栈中，车辆的环境感知能力是最为关键的一环。目前无人驾驶车辆依赖多种传感器感知环境，譬如激光雷达(LIDAR)、摄像头、毫米波雷达等。这些传感器的数据能够通过算法融合还原真实的三维环境，得到场景中物体的位姿与运动信息。

三维物体检测是无人驾驶环境感知的一个重要任务。该任务要求无人驾驶系统能够通过分析传感器采集的数据，计算出场景中各个目标的三维坐标以及位姿。相比于二维图像的物体检测任务，三维物体检测要更加困难。这是因为维度诅咒(Curse of dimensionality)的存在：当维度增加时，空间的体积增加得非常之快(以指数增加)，以致于可用的数据变得稀疏。三维物体检测的一大挑战，就是三维数据的稀疏性。目前无人驾驶中三维数据的获得主要是依靠激光雷达-高速旋转的激光发射器向周围发射激光，通过检测接收到反射回来光线的时间间隔来计算反射点的距离。激光雷达的线束对其分辨率有很大影响，特别是对于远距离的物体，在低线束(16线)激光雷达中可能只有稀疏的几个点，完全无法分辨。然而高线束的激光雷达价格十分昂贵(以Velodyne 64线激光雷达为例，其价格高达十几万美金)，因此多数无人驾驶技术方案都需要在性能与价格之间做出权衡。

从使用数据的类型来看，目前针对三维环境的物体检测技术主要分为三大方向：基于图像的方案，基于点云数据的方案，以及多传感器数据融合方案。尽管如此，绝大多数方案都局限于使用一帧数据进行检测。对于真实场景的检测任务来说，数据都是以流的形式连续获取的。相比于单帧数据，流数据可以提供同

一目标在一段时间内的信息，该信息有利于检测算法筛选误检测的目标，这是因为噪声在时序上的连续性较差。另一方面，对于遮挡以及边界截断的目标，在流数据中可以利用前后帧的信息对其进行补全。另外一点值得注意的是，流数据有很强的数据冗余性，如果使用单帧物体检测算法，则需要逐帧进行检测，十分耗时；但是如果使用针对流数据的检测算法，则可以只对少数关键帧进行检测，然后将检测结果传播到非关键帧。因此，开发专门针对流数据的三维物体检测算法，能显著提高物体检测的准确率与效率。这从算法的落地来看，是十分有实践意义的。

## 1.2 国内外研究进展

目前国内外针对流数据的三维物体检测研究还较少，而基于单帧数据的三维物体检测以及基于视频流的二维物体检测的研究较为丰富。因此，本节分别从三维物体检测以及视频流物体检测概述前沿进展。另外，三维场景的多目标追踪研究进展也会进行简要说明。

### 1.2.1 三维物体检测

目前大多数三维检测研究可以归类为三大方向：基于图像，基于点云，以及基于多传感器融合。

(1) 基于图像的方法。基于图像的三维物体检测以 Mono3D [1] 和 3DOP [2] 为代表。这类方法只是用摄像头采集的图像数据，通过多视角图像融合来推断三维信息。由于图像数据没有深度信息，这类方法需要人工设计的几何特征来表征物体的深度信息，虽然数据采集简单，速度快，但是检测精度差。最近越来越多研究者探索基于双目摄像头数据的三维物体检测，例如 [TODO]。这类方法通过神经网络融合左右摄像头的数据来预测物体的三维信息，从生物学上来说很接近人类的双眼视觉系统。然而由于神经网络的不确定性，以及双目视觉盲区的存在，该方法仍然有很多亟待解决的问题。

(2) 基于点云的方法。基于点云的方法可分为一步法和两步法。一步法即直接是端对端预测物体的三维位姿，这类方法根据点云的编码方式不同又可分为基于体素的方法以及基于投影的方法。基于体素的方法使用三维体素网格编码点云特征，每个体素立方体的值由该立方体内的点决定，从而将不规则的三维点云

数据编码成规则的三维体素数据，便于后续使用神经网络进行特征提取。代表有 3D FCN [3], Vote3Deep [4] 以及 VoxelNet [5] 等。这类方法的一大缺点是体素大小不好确定，太大的话信息损失严重，太小则会造成巨大的计算量。另外，这类方法需要使用到三维卷积，因此很耗计算资源。基于投影的方法是将点云在高度方向进行投影，将三维数据降维成二维的俯视图 (BEV, bird eye view) 数据。考虑到驾驶场景中，道路基本是共面且水平的，因此在高度方向上投影对物体的位姿信息基本没有损失。经过投影操作后，就能直接使用二维图像物体检测的方法进行物体检测。PIXOR [6] 以及 Complex-YOLO [7, 8] 等属于这类方法。虽然降维能够带来速度的极大提升，然而由于点云数据的稀疏性，经过投影后目标的特征点损失很严重。特征点的不足会很大程度上影响检测结果的准确性，特别是对于远处的目标以及小目标。两步法的思路是先对点云进行分割，然后对于分割结果的每一个目标进行框回归，得到目标的三维位姿信息。这类方法的代表有 [9]。两步法的不足之处是其检测结果非常依赖于分割结果，且在框回归中，对于点较少的远处目标，三维位姿信息的恢复效果并不好。

(3) 基于多传感器融合的方法。点云的稀疏性以及图像缺少深度信息都限制着相应方法的准确率，一个自然而然的想法就是能不能将这两种数据融合，从而达到更好的检测结果。基于多传感器融合的方法通过算法融合点云数据以及图像数据，从而提升三维物体检测的准确率。这类方法的代表有 F-PointNet [10], MV3D [11], AVOD [12] 等。这些方法的区别主要在于融合的方式不同，F-PointNet 首先使用二维物体检测方法检测出图像中的所有物体，之后对于每个物体，将其反投影回点云中得到一个视锥区域，对该区域内的点进行分割然后再进行框回归。该方法通过先在图像中找出目标，从而减少了在点云空间的搜索区域。然而，该算法的准确率受二维物体检测很大，在第一步没有检测出的物体，之后没有其他办法弥补。MV3D 将二维物体检测中的的区域提取网络 (Region Proposal Network, RPN) 扩展到三维，提出了 3D RPN 分别提取点云以及图像特征，然后通过一个特征融合模块得到融合特征，最后进行三维物体检测。AVOD 在 MV3D 的基础上改进了特征提取模块，引入的金字塔结构从而能够得到全分辨率的特征图，提升了物体的定位精度，特别对于小物体。本文工作基于 AVOD 框架并进行了改进，使其能支持多帧输入，并且引入了时序信息处理模块。

### 1.2.2 视频流物体检测

视频流物体检测与单帧物体检测的主要区别在于是否利用了时序信息。对于视频流物体检测，时序信息是物体的位姿在时间上的连续性的抽象体现。目前，大多数视频流物体检测方法都是在两个层面利用时序信息，特征提取层面以及最终的框回归层面。对于特征处理层面，一般是根据运动信息将前后帧的特征整合到关键帧，以增加关键帧的物体特征。这个过程中需要使用到光流信息，即图像中各像素点的运动方向。代表有 FGFA [13] 系列工作。一般来说光流信息的获取比较困难，这也是限制该方法进一步发展的主要障碍。对于在框回归层面利用时序信息，主要的工作有 T-CNN [?] 与 Seq-NMS [14] 等。T-CNN 使用预先计算的光流信息将关键帧的检测结果传播到临近帧，而 Seq-NMS 则是通过整合连续几帧的高置信度的候选框来提升目标检测中非极大值抑制 (Non-Maximum Suppression, NMS) 算法的性能。最近，也有一些工作试图通过神经网络学习连续帧之间的时序信息，从而避免使用高代价的光流数据。这类方法的代表有 D&T [15]。D&T 提出了一个双路目标检测网络，可以同时进行视频流的目标检测以及目标追踪。该网络可以输入多帧数据进行检测，并且通过互相关操作 (Cross-correlation) 来学习相邻帧之间相同物体的对应关系以及偏移。本文的算法框架在一定程度上也借鉴了 D&T 的结构，不过我们在他的基础上进行了很大的改进，使其能够适应三维物体的流数据检测。

### 1.2.3 三维多目标追踪

目前基本上所有的三维多目标追踪方法都是先对流数据的每一帧进行目标检测，然后再将这些检测框关联起来，这种范式也被称为 *Tracking by Detection* [16]。三维多目标追踪的工作有很多，比较有代表性的有 FaF [17], 3D-CNN/PMBM [18] 以及 DSM [19] 等。FaF 使用首先将点云流数据结构化成四维张量，然后构建了一个简单的特征提取网络提取特征，最后使用不同的网络头分别预测得到三维目标检测，多目标追踪以及运动方向预测结果。该方法能够整合前  $n$  帧的检测结果得到精确的物体运动轨迹。然而该方法计算量巨大，并且网络参数调节需要有很高的技巧。3D-CNN/PMBM 首先构建神经网络从单张图像预测物体的三维位姿，然后将所有帧的检测框送入泊松多重伯努利 (Poisson Multi-Bernoulli Mixture,

PMBM) 混合追踪滤波器进行滤波，得到最终的三维多目标追踪结果。该方法只使用单帧图像进行三维目标检测，效果有限。DSM 首先使用单帧三维物体检测框架 MV3D [11] 对每一帧数据进行物体检测得到三维检测框，然后通过一个匹配网络 (*Matching net*) 以及得分网络 (*Scoring net*) 关联所有的检测框。该方法需要对每一帧数据都进行检测，并且帧与帧之间的时序信息基本上没有被使用，因此不是针对流数据的高效方法。

### 1.3 本文工作及创新点

本文提出了一个双路物体检测与追踪 (**Dual-way Object Detection and Tracking, DODT**) 框架，实现了流数据场景的高效三维物体检测与追踪。本框架的构建是基于以下几个观察：(1) 点云能够与图像融合从而丰富物体的视觉特征，这点在 [11, 12] 中得到了证实；(2) 除了通过光流数据，时序信息也能够通过计算相邻帧间的互相关信息，这点通过 [15] 也能够得到验证；(3) 特征在连续帧之间的变化是连续的，我们可以只对关键帧进行检测，然后将结果传播到非关键帧，这样可以极大的减少计算量。对于第一点，本文借用了 AVOD[12] 中的数据融合方案，将点云数据提供的 BEV 信息与图像融合。对于第二点，本文构建了一个时序模块 (Temporal module)，该模块使用互相光操作在 BEV 空间中计算相邻关键帧的时序特征，然后预测相同物体在不同关键帧中同时出现的概率以及偏移量。与 [15, 20] 不同的是，本模块的互相关操作是在后候选框层面进行的，不需要全局计算，这极大地提高了模块的运行效率。对于最后一点，我们将 DODT 框架的目标检测模块设计成了双路结构，这样该模块就能够同时输入两帧相邻关键帧，以保证后面时序模块的正确运行。另外，为了进一步提高框架的效率，本文还设计了一个共享 RPN (Shared Region Proposal Network, Shared RPN) 模块，该模块可以生成供两检测分支共同使用的三维候选框。最后，为了生成所有帧的检测结果，本文设计了一个基于运动的框插值算法，该算法利用关键帧的检测结果以及时序模块预测的信息，插值生成非关键帧的检测结果。同时，该差值算法还能够将不同帧的候选框关联起来，得到多目标追踪结果。

本文的贡献及创新点如下：

- 本文提出了名为 DODT 的双路网络，该网络能够同时精确地完成基于流

数据的三维物体检测以及多目标追踪任务。

- 本文提出了一个时序模块在候选框层面上编码相邻关键帧之间的时序信息，相比于 [15, 20] 中方法，该方法更加灵活，也更加高效。
- 本文设计了一个共享 RPN 模块，能够显著提高相邻多帧目标检测中候选框提取的效率。
- 本文开发了一个基于运动的框插值算法，能够有效的将关键帧的预测框传播到非关键帧，同时也能够将所有框关联起来，实现多目标追踪。

## 1.4 文章组织与结构

本文的主要内容分为五章。第一章为引言，介绍项目的研究背景和意义，国内外的研究进展以及本文工作的简单介绍和创新点；第二章介绍本工作涉及到的一些技术的基础理论，分为目标检测与目标追踪两大块；第三章详细地介绍了本文提出的框架的构造和原理，是全文的重点内容；第四章主要是介绍了本项目的实验设计，结果展示以及实验结果分析，该部分也是全文的重点内容；最后一章总结本文的工作，然后介绍本文工作的不足之处以及后续的实验计划。

## 第 2 章 目标检测与跟踪技术介绍

由于本工作涉及到的领域包括目标检测以及目标追踪，较为复杂。因此本文单独使用一章来介绍相关的背景知识，希望能为后续本文工作的介绍提供了技术参考。

### 2.1 目标检测

目标检测 (Object Detection) 作为计算机视觉中最基本的任务之一，一直以来都受到了学术界与产业界的极大关注。特别是在最近二十年，随着深度学习技术的飞速发展，神经网络已经是目标检测中必不可少的组成部分。可以说，是深度学习的引入将目标检测推向了新的高度，远远超出了传统目标检测方法的性能。以图像为例，目标检测需要在图片中精确找出物体所在的位置（一般以矩形框出），并标注物体的类别。由于物体的尺寸变化范围很大，摆放物体的角度、姿势等也不确定，并且物体间也会有重叠，这些问题使得目标检测问题不是那么容易解决。现阶段基于深度学习的目标检测框架主要有两类，一类是以 Faster-RCNN 为代表的两阶段目标检测方法，另一类是以 YOLO 为代表的单阶段目标检测方法。本小节详细介绍了这两种框架发展历史以及结构细节与实现原理，此外，也简单介绍了三维目标检测相对于二维目标检测的改进。

#### 2.1.1 两阶段目标检测

目标检测任务包含目标识别以及目标定位两个子任务，一般的实现思路是先用算法截取图片中可能作为目标的不同部分（候选区域），然后分别送入一个分类器进行分类。将截取候选区域算法的实现与后续分类器的实现分开，这就是两阶段物体检测的实现思路。两阶段物体检测算法的代表是何凯明团队的系列工作。该系列历经 RCNN，Fast-RCNN 再到 Faster-RCNN，将两阶段物体检测框架不断完善，成为该领域的经典之作。

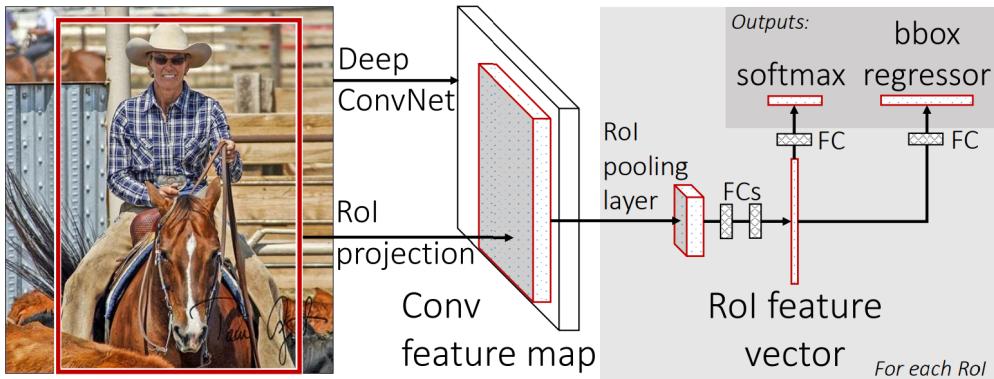


图 2-1 Fast-RCNN 结构示意

### 2.1.1.1 发展过程

RCNN 可以说是利用深度学习技术进行目标检测的开山之作。该工作使用选择性搜索 (Selective Search) 算法代替滑动窗口来截取候选区域，并且利用训练的神经网络对图片进行特征提取，然后使用 SVMs 进行类别判断。该工作奠定了两阶段目标检测框架的基本结构，即一个区域提取 (Region Proposal) 算法用于提取输入图像中可能是目标的区域，一个 CNN 网络用于提取区域特征，最后一个分类算法用于类别鉴定。在 RCNN 的基础上，RBG[TODO] 提出了 Fast-RCNN 算法。该算法借鉴了 SPP-Net 的实现，对 RCNN 做了改进，使得检测性能进一步提高，其结构如图 2-1 所示。具体而言有两个重大改进：（1）将候选区域提取阶段移到了图像特征提取之后，即只对原图做一次卷积，然后在特征图上运行候选区域提取算法得到候选区域特征图，然后将每一个特征图输入 ROI(region of interest) 层来将不同尺度的特征图转化为相同维度的特征向量，之后送入全连接层进行后续处理；（2）RCNN 训练神经网络提取图像特征，用支持向量机分类以及用回归模型精修边界框，然而 Fast-RCNN 利用两个全连接层 (Softmax Classifier, Bounding-Box Regressors) 将三个任务整合到一个模型里联合训练，这为之后的目标检测端对端训练打下了基础。

尽管 Fast-RCNN 相对于 RCNN 已经提速了不少，但要实现实时检测，速度还是不够，主要的速度瓶颈在候选区域提取阶段。基于 CPU 实现的 Selective Search 算法提取一幅图像的所有候选区域 (Proposals) 需要约 2s 时间，效率更高的 EdgeBoxes 算法虽然在一定程度上提高了候选区域提取的准确率和效率，但

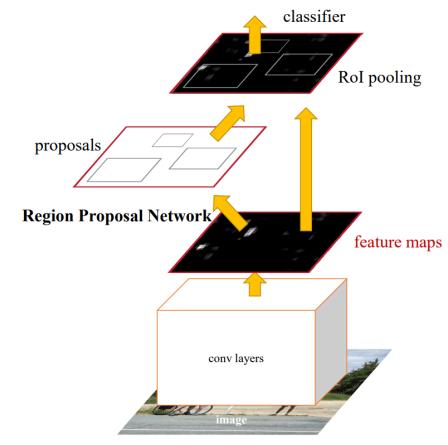


图 2-2 Faster-RCNN 结构示意

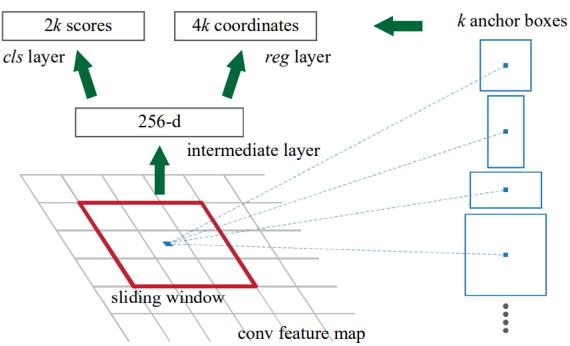


图 2-3 RPN 结构示意

处理一幅图像仍然需要 0.2s。为了解决这个问题，2015 年微软亚洲研究院提出了 Faster-RCNN 算法，该算法引入了 RPN 网络 (Region Proposal Network)，使用神经网络来取代传统的区域提取算法，将单幅图像候选区域提取时间降到了 10ms。接下来本文将以 Faster-RCNN 为代表，重点介绍两阶段物体检测框架的结构与实现细节。

### 2.1.1.2 Faster-RCNN 框架结构

Faster-RCNN 框架由两大模块组成，候选框提取模块 (RPN) 和检测模块，如图 2-2 所示。在 Faster-RCNN 框架中，首先输入一幅图像，图像经过卷积神经网络提取特征得到特征图 (feature maps)，然后 RPN 模块输入特征图预测得到候选框 (proposals)，之后根据候选框去特征图中截取特征块，特征块经过 ROI Pooling 操作得到特征向量，最后送入检测头 (classifier) 和回归头 (regressor) 分别得到分类结果以及预测框。RPN 是 Faster-RCNN 新引进的模块，其结构如图 2-3 所示。在特征提取网络生成的特征图 ( $M \times N$ ) 上，使用  $n \times n$  的卷积核卷积生成  $M \times N \times k$  个维度为 256 的中间特征向量，之后输入分类全连接层 (cls layer) 和回归全连接层 (reg layer)，分别用来预测候选框的类别 (前景/背景， $M \times N \times 2 \times k$ ) 以及候选框的位置信息 (中心点坐标  $x, y$  以及宽高  $w, h$ ， $M \times N \times 4 \times k$ )。其中， $k$  表示对于特征图上的每一个像素点，都负责预测  $k$  个锚点框 (anchor boxes)，为满足候选框的尺寸变化， $k$  个锚点框有不同的尺寸和长宽比，例如 Faster-RCNN 中有三种尺寸： $128 \times 128, 256 \times 256, 512 \times 512$  (px)，三中长宽比： $1 : 1, 1 : 2, 2 : 1$ ，因此

$k = 3 \times 3 = 9$ , 意味着每个锚点负责预测 9 个不同的候选框。

### 2.1.1.3 Faster-RCNN 网络训练

构建了网络结构, 网络的训练还需要准备训练数据, 明确损失函数等。目标检测的训练数据一般有公开的数据集, 如 COCO, PASCAL 等, 然而 Faster-RCNN 还需额外训练 RPN 网络, 因此需要准备 RPN 网络训练的标签。候选框标签的生成需要借助真实标签生成: 首先根据上文中锚点的概念, 以特征图尺寸为参照生成  $M \times N \times k$  个候选框, 然后根据候选框与真实物体框的重叠程度(一般以候选框与真实框的交并比为量化指标)计算每个候选框的得分。得分可作为划分前景和背景的标准, 譬如得分大于 0.65 划分为前景, 小于 0.35 划分为背景, 这样就得到了 RPN 网络的训练数据标签。

Faster-RCNN 的损失函数有两部分构成, RPN 损失和 Fast-RCNN 损失, 并且这两部分损失又分别包括分类损失和回归损失。

$$L(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*) \quad (2-1)$$

RPN 损失函数如公式 2-1 所示, 其中第一部分为分类损失, 第二部分为回归损失。分类损失计算了 RPN 预测生成的候选框的类别与标签的交叉熵损失  $L_{cls}$ , 交叉熵损失公式如 2-2 所示。其中  $p_i$  为预测生成的候选框类别,  $p_i^*$  是标签值。RPN 网络生成的候选框分为前景和背景, 前景标签为 1, 背景为 0, 因此可以用一个 one-hot 向量表示。

$$L_{cls} = -\log[p_i^* p_i + (1 - p_i^*)(1 - p_i)] \quad (2-2)$$

在训练中, RPN 网络生成的  $M \times N \times k$  个候选框, 然而并不是每个候选框都会纳入损失值的计算范围, 因为这些候选框有很多重叠的。Faster-RCNN 使用了非极大值抑制(Non-Maximum Suppression, NMS)算法进行初步筛选, 减少重叠的候选框。非极大值抑制算法如 1 所示: 对于 RPN 生成的候选框集合  $\mathcal{B}$  以及对应的置信度集合  $\mathcal{S}$ , 首先选择最大置信度的候选框  $M$ , 将其从  $\mathcal{B}$  中移除并加入最终候选框集合  $\mathcal{D}$ ; 然后遍历  $\mathcal{B}$ , 移除与  $M$  的交并比(Intersection of union, IoU)大于阈值  $\epsilon$  的框; 重复此过程, 直到  $\mathcal{B}$  为空。通过选择合适的阈值(Faster-RCNN 中为 0.7), NMS 算法可以过滤大部分重叠的候选框, 之后从中随机选取  $N_{cls}$  候选框计算分类损失, 在 Faster-RCNN 中  $N_{cls} = 256$ 。

**Algorithm 1:** 非极大值抑制算法

---

```

1 输入:  $\mathcal{B} = \{b_1, \dots, b_N\}$ , RPN 生成候选框集合;  $\mathcal{S} = \{s_1, \dots, s_N\}$ , 生成候选框
      对应的置信度集合;  $\epsilon$ , 置信度阈值
2 初始化:  $\mathcal{D} \leftarrow \{\}$ 
3 while  $\mathcal{B} \neq \text{empty}$  do
4    $m \leftarrow \text{argmax } \mathcal{S}$ 
5    $\mathcal{M} = b_m$ 
6    $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{M}; \mathcal{B} \leftarrow \mathcal{B} - \mathcal{M}$ 
7   for  $b_i$  in  $\mathcal{B}$  do
8     if  $\text{iou}(\mathcal{M}, b_i) \leq \epsilon$  then
9        $\mathcal{B} \leftarrow \mathcal{B} - b_i; \mathcal{S} \leftarrow \mathcal{S} - s_i$ 
10  输出:  $\mathcal{D}, \mathcal{S}$ 

```

---

RPN 回归损失计算预测的候选框与标签的  $Smooth_{L1}$  损失  $L_{reg}$ , 注意 RPN 回归损失只计算前景的损失, 因此  $L_{reg}$  前需乘以  $p_i^*$ (前景为 1, 背景为 0)。 $N_{reg} = N_{cls}$ , 为经过 NMS 算法过滤后随机选择的候选框数。 $Smooth_{L1}$  损失公式如2-3所示,

$$L_{reg}(t_i, t_i^*) = \begin{cases} 0.5(t_i - t_i^*)^2 & |t_i - t_i^*| \leq 1 \\ |t_i - t_i^*| - 0.5 & \text{otherwise} \end{cases} \quad (2-3)$$

其中  $t_i$  和  $t_i^*$  分别对应预测候选框以及真实候选框的信息。需要注意,  $t_i = (t_i^x, t_i^y, t_i^w, t_i^h)$  为一四维偏移向量, 其计算公式如2-4所示。其中  $(x_a, y_a, w_a, z_a)$  分别是对应锚点框中心的坐标以及宽高。RPN 并不直接对候选框回归, 而是回归候选框与对应的锚点框的偏移, 这有利于网络的训练。

$$t_x = \frac{x - x_a}{w_a}; t_y = \frac{y - y_a}{h_a}; t_w = \log\left(\frac{w}{w_a}\right); t_h = \log\left(\frac{h}{h_a}\right) \quad (2-4)$$

Fast-RCNN 的损失和 RPN 类似, 同样由分类损失和回归损失组成。不过 RPN 的分类损失是二分类的交叉熵损失, 而 Fast-RCNN 是多分类的交叉熵损失, 将类别标签转化为 one-hot 向量后并没有本质区别。和 RPN 类似, Fast-RCNN 也不对所有 ROI 计算损失, 而是先通过更严格的 NMS 算法过滤 (Faster-RCNN 中阈值设置为 0.3), 然后随机选择一定数量的候选框进行损失值计算。Fast-RCNN 的回

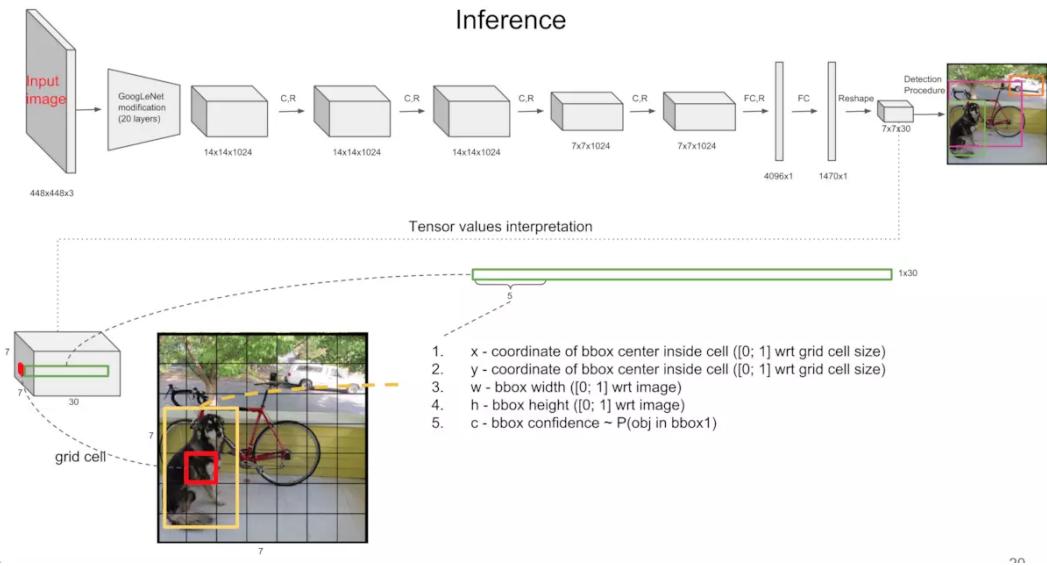


图 2-4 YOLO 结构示意

归损失基本也和 RPN 的一致，不过这次是回归物体真实框与锚点框的偏移。

### 2.1.2 单阶段目标检测

从 RCNN 到 Fast-RCNN 再到 Faster-RCNN，一直采用先计算出候选区域然后再在候选区域上进行检测的设计范式。该模式可以获得很好的检测精度，然而在速度上还不佳，最好的 Faster-RCNN 只能达到 0.2s 一帧的检测速度，离实时检测还有较大差距。单阶段目标检测提供了另一种更直接的设计思路：不显式计算候选框，而是通过神经网络直接预测物体的位置与类别。该模式的代表工作有 YOLO 系列以及 SSD，本节以 YOLO 为例，介绍单阶段目标检测的实现方法以及网络训练。

YOLO 的整体架构如图 2-4 所示。与 RCNN 系列工作不同，YOLO 将目标检测作为回归任务来解决。YOLO 首先将输入图像宽高调整到  $448 \times 448$ ，然后输入到由 GoogLeNet 改进的网络中，最后输出一个维度为  $S \times S \times (B \times 5 + C)$  的张量  $\mathcal{T}$ 。 $\mathcal{T}$  可以这么理解，首先将一幅图像分成  $S \times S$  (YOLO 是  $7 \times 7$ ) 个网格，对于每个网格  $s$ ，其对应着  $\mathcal{T}$  中的一项维度为  $1 \times (B \times 5 + C)$  的向量  $t$ 。如果一个目标物体的中心落在  $s$  上，则  $s$  负责预测该物体，预测结果编码在向量  $t$  中。 $t$  维度为  $B \times 5 + C$ ，其中  $B$  表示每个网格预测两个边界框 (bounding box, bbox)；5 编码了 bbox 的信息，分别是  $(x, y, w, h, c)$ ，代表 bbox 中心点坐标、宽高以及置信

度;  $C$  表示需要预测  $C$  个类, 每个值表示预测为该类的概率值  $Pr(class_i|object)$ 。

和 Faster-RCNN 类似, YOLO 也不是直接回归出物体的实际坐标和宽高, 而是 bbox 相对于单元格的偏移量。对于向量  $t = (x, y, w, h, c)$ ,  $(x, y)$  的计算公式如2-5所示, 其中  $(x_c, y_c)$  是 bbox 中心点的实际坐标,  $(w_i, h_i)$  是图像的宽高,  $x_{col}, y_{col}$  是单元格的坐标。最终预测出来的  $(x, y)$  是经过归一化处理的, 中心相对于单元格的偏移。 $(w, h)$  的计算公式如2-6所示, 是 bbox 相对于整张图像的比例。置信度  $c$  的计算公式如2-7所示, 由两部分组成,  $Pr(object)$  表示单元格内是否有物体, 有为 1, 没有则为 0;  $IoU_{Pred}^{Truth}$  表示 bbox 位置的准确度, 用 IoU 衡量。

$$x = \frac{x_c}{w_i}S - x_{col}; y = \frac{y_c}{h_i}S - y_{row} \quad (2-5)$$

$$w = \frac{w_b}{w_i}; h = \frac{h_b}{h_i} \quad (2-6)$$

$$c = Pr(object) * IoU_{Pred}^{Truth} \quad (2-7)$$

在测试阶段, 网络最终输出为一个  $S \times S \times (B \times 5 + C)$ , 其中包含  $S \times S \times B$  个预测框, 每个预测框的最终概率为  $Pr(class_i|object) * c$ , 为综合了定位误差和分类误差的 20 维向量。最后这  $S \times S \times B \times C$  列的结果送入 NMS 算法去除重复的检测框, 得到最终的检测结果。

YOLO 的损失函数设计比较复杂, 如公式2-8所示。YOLO 的损失函数都采用平方和损失, 可分为五部分来看:

$$\begin{aligned} L_{yolo} &= \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{i,j}^{obj} (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \\ &\quad + \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{i,j}^{obj} (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \\ &\quad + \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{i,j}^{obj} (C_i - \hat{C}_i)^2 \\ &\quad + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{i,j}^{noobj} (C_i - \hat{C}_i)^2 \\ &\quad + \sum_{i=0}^{S^2} \mathbb{1}_i^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2 \end{aligned} \quad (2-8)$$

第一部分为 bbox 中心坐标的误差, 其中  $\mathbb{1}_{i,j}^{obj}$  表示判断第  $i$  个网格中第  $j$  个 bbox 是否负责预测该物体。第二部分为 bbox 宽高的误差, 由于在对不同大小的 bbox

的预测中，小 bbox 预测偏离的容忍程度比大 bbox 要小很多，而平方和损失对 bbox 的尺度并不敏感。为了解决这个问题，YOLO 使用宽高的平方根代替原来的宽高。第三部分为含有物体的 bbox 的置信度预测误差，第四部分为不含物体的 bbox 的置信度预测误差。最后一部分是对 bbox 类别预测的误差，其中  $1_i^{obj}$  表示是否有物体落在单元格  $i$  中。另外，为了平衡各种损失之间的比例，YOLO 还加入了  $\lambda_{coord}, \lambda_{noobj}$  作为超参数平衡各部分损失。

YOLO 由于没有显式的预测候选框的过程，因此检测速度很快。但是由于网格的设定，YOLO 对相互靠的很近的物体（中心可能落在同一个单元格内），还有很小的物体检测效果不好，这是由于一个单元格只预测一个类。另外，YOLO 对于同一类物体出现不常见的长宽比等情况的泛化能力较差。这些问题在其后的 YOLOv2、YOLOv3 中有探究。

### 2.1.3 三维目标检测

三维目标检测目前有两大类，一类是基于二维目标检测进行扩展到三维，一类是先进行点云语意分割，然后对分割后的每个类别的点进行聚类，之后在进行物体框的回归。直接扩展二维目标检测方法的有基于两阶段目标检测的 MV3D、AVOD 等，基于单阶段目标检测的 VoxelNet 等。在模型框架层面，这一大类的三维目标检测和二维的没有多大区别，只是需要先将三维点云编码为适合卷积的数据格式，然后将二维边框编码扩展到三维，以及损失函数做出相应的修改。具体的改动将会在下一章介绍本文方法时详细介绍。另一类先分割再回归框的方法有 PointRCNN 等，由于这类方法与本文方法差距较大，这里不过多介绍。

## 2.2 目标跟踪

由于本工作也涉及到一些目标跟踪方面的应用，因此本文也简单介绍下目标跟踪的一些基本知识。目标跟踪问题是指给定第一帧物体的位置，根据算法预测出后续帧中该物体的位置。根据追踪物体的数量的不同，目标追踪分为单目标追踪 (Single Object Tracking, SOT) 和多目标追踪 (Multiple Object Tracking, MOT)。SOT 与 MOT 虽然都属于目标跟踪，但是确实两个差别很大的研究方向。由于本工作只涉及到多目标跟踪，因此会重点介绍，而单目标跟踪只会简单叙述其基本原理。

### 2.2.1 单目标跟踪

单目标跟踪只针对单个物体进行追踪，其算法实现的基本思路如下：首先根据首帧提供的物体信息提取目标物体特征，例如灰度特征、颜色特征、纹理特征或者使用神经网络提取的视觉特征；然后基于帧间的目标运动关系，构建运动模型，经典的运动模型如均值偏移、滑动窗口、卡尔曼滤波以及粒子滤波等；之后根据运动模型确定的后续帧目标位置信息，提取该图像区域特征后输入外观模型与首帧物体特征进行对比判断；最后需要一个在线更新机制在跟踪过程中不断更新外观模型，使得模型能够捕捉目标的变化，适应长距离的跟踪。

学术界针对单目标追踪有大量的研究，如首次将相关滤波引入单目标跟踪的 MOSSE[21] 以及后续在此基础上发展的 KCF[22]、DSST[23]、C-COT[24] 以及 ECO[25] 等。另外，这些年基于孪生网络的单目标追踪模型发展迅速，如首先开创端对端深度学习式相关滤波方法先河的 SiamFC[26]，以及加入了 RPN 应对尺度变化的 SiamRPN[27] 以及其改进版本 DaSiamRPN[28] 等。从近几年的单目标跟踪研究可以看出，深度学习技术也在该领域也引发了技术革新。

### 2.2.2 多目标跟踪

多目标跟踪同时对多个物体进行跟踪，相比于单目标跟踪，其实现难度更大，因此进展较为缓慢。多目标跟踪一般会涉及到目标出现、遮挡以及离开视野等复杂情况。目前学术界针对多目标跟踪的主流实现思路是基于检测的跟踪 (Detection-Based Tracking)，该方法要求先由一个目标检测器将每一帧的目标都检测出来，然后再使用匹配算法将不同帧的同一物体相关联。按照处理方式，多目标跟踪算法可分为在线跟踪 (Online Tracking) 以及离线跟踪 (Offline Tracking)。在线跟踪要求算法只能根据当前帧以及之前帧的信息得出当前帧的跟踪结果，而离线跟踪允许算法利用所有帧的信息，获取全局最优解。相对于离线跟踪，在线跟踪更适合实际应用场景。例外，还有一种近似在线跟踪 (Near Online Tracking) 方法，该方法利用当前帧一定窗口范围内的帧的信息得出当前帧的跟踪结果，是一种折中方法，只会导致些许延迟。

目前多目标跟踪算法基本是使用现有的检测器模型得到所有帧的检测结果，然后设计数据关联算法得到多目标跟踪结果。例如 SORT[29] 使用 Faster-RCNN

作为目标检测器得到检测结果，然后使用中心坐标、面积、长宽比等对目标进行建模，使用卡尔曼滤波预测目标的后续状态，并将预测结果与实际结果进行匹配从而得到最终结果。在此基础上，作者使用了深度卷积神经网络提取目标特征作为匹配基础改进 SORT 而得到 DeepSORT[[30]]。此外，还有一种比较简单的，使用前后两帧各目标框的 IoU 进行关联的 IoU Tracker[31]，该算法严重依赖于检测框的准确率。为了改善对检测结果精度的依赖，作者之后推出了改进版 V-IoU Tracker[32]，该算法加入了检测框的长程连续性特征，没有检测出的目标不再认为立即消失，而是会根据其运动趋势保持更新一定时间步长。本工作所使用的目标跟踪算法是基于 V-IoU Tracker 算法改进的。

多目标跟踪的性能评价指标和单目标跟踪有很大不同，当前学术界最常使用的衡量指标为 CLEAR MOT[33] 论文提出的多目标追踪准确度 (Multiple Object Tracking Accuracy, MOTA) 和多目标追踪精确度 (Multiple Object Tracking Precision, MOTP)。其中 MOTA 的计算公式如2-9所示，其中  $t$  表示帧数， $m_t, fp_t, mme_t$  分别是  $t$  帧漏检、误检以及错误匹配的数量， $g_t$  为  $t$  帧中的真实标签。从公式可以看出 MOTA 可以分为三部分， $\bar{m}$  为漏检率， $\bar{fp}$  为误检率， $\bar{mme}$  为错误匹配率，这三种错误率的总和就是总错误率。MOTA 直观的给出了衡量算法连续跟踪识别目标的能力，不过没有涉及到目标的检测位置精确度。

$$\begin{aligned} MOTA &= 1 - (\bar{m} + \bar{fp} + \bar{mme}) \\ &= 1 - \left( \frac{\sum_t m_t}{\sum_t g_t} + \frac{\sum_t fp_t}{\sum_t g_t} + \frac{\sum_t mme_t}{\sum_t g_t} \right) \\ &= 1 - \frac{\sum_t (m_t + fp_t + mme_t)}{\sum_t g_t} \end{aligned} \quad (2-9)$$

另一个指标 MOTP 与 MOTA 互补，其衡量跟踪目标位置的精确度，而不衡量跟踪识别目标的能力。MOTP 的计算公式如2-10所示，其中  $t$  表示帧数， $i$  表示第  $t$  帧第  $i$  个目标-预测匹配对， $d_t^i$  表示第  $t$  帧中第  $i$  个目标-预测匹配对的位置误差， $c_t$  表示第  $t$  帧中总匹配对的个数。在计算 MOTA 和 MOTP 时，都是基于整个跟踪过程计算平均值的，而不是基于每一帧的结果，这是因为基于单帧计算然后再求平均会导致和直观上不同的结果。

$$MOTP = \frac{\sum_{i,t} d_t^i}{\sum_t c_t} \quad (2-10)$$

除了 MOTA 与 MOTP，为了更好的在轨迹层面上衡量多目标追踪，学术界一

般还会引入其他指标。例如多数追踪率 (Mostly Tracked, MT)、多数丢失率 (Mostly Lost, ML)、ID 切换次数 (ID-switches, IDS)、片段数 (Fragmentations, FM)。其中 MT 目标的大部分被追踪到的轨迹占比 (大于 80%)，ML 表示目标的大部分跟丢的轨迹占比 (小与 20%)，IDS 为一条跟踪轨迹改变目标编号的次数，FM 为真实轨迹被打断的次数。本工作使用 MOTA、MOTP 以及以上四种指标评估多目标跟踪算法的性能。

### 2.3 本章总结

本章主要介绍了本文涉及到的两个领域，目标检测与目标跟踪的基本技术原理。首先介绍了两类目标检测框架的网络结构和训练方法，其中两阶段目标检测以 Faster-RCNN 作为代表进行介绍，而单阶段目标检测则以 YOLO 为代表进行介绍。之后，本章简单介绍了单目标跟踪算法的实现思路和几种经典算法。本章最后介绍了多目标跟踪的实现方式和几种典型算法，以及多目标追踪性能的评价指标。该章的内容是为下一章中本工作方法介绍的时候提供技术参考，本章涉及到的技术原理，下章将不再深入介绍。

## 第3章 流数据的三维目标检测与跟踪

本章将详细介绍本工作提出的流数据三维目标检测与跟踪框架 **DODT** (**D**ual-way **O**bject **D**etection and **T**racking)。本章先介绍 DODT 的整体网络架构，然后再重点分析各个模块的功能以及实现。

### 3.1 框架整体结构

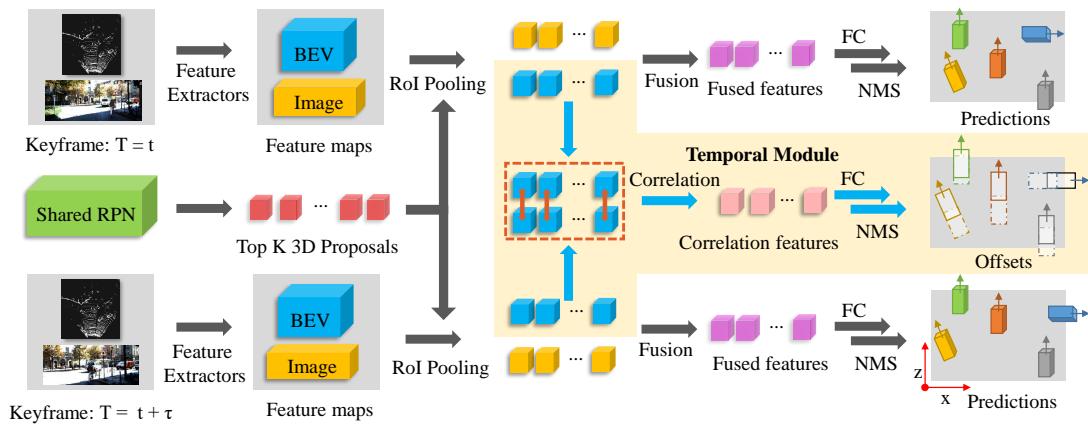


图 3-1 DODT 框架结构。浅黄色区域为时序处理模块。

DODT 网络是双路结构，其结构如图 3-1 所示。DODT 由四个小模块组成：三维物体检测模块、Shared RPN 模块、时序信息处理模块以及运动插值模块。三维物体检测模块有两个分支，分别负责检测两相邻关键帧中的物体，两分支的网络结构相同，并且共享参数。检测模块的网络结构是基于 AVOD[12] 构建的，为二阶段物体检测框架的三维度扩展，该模块的详细结构将在 3.2 节介绍。Shared RPN 模块负责生成二阶段物体检测中的候选框，与传统 RPN 不同的是，Shared RPN 生成的 3D 候选框可以被两个三维检测分支共同使用，该模块将在 3.3 节详细介绍。时序处理模块为图 3-1 的浅黄色区域，该模块通过对相邻关键帧的点云俯视图 (Bird Eye View, BEV) 特征匹配块进行相关 (correlation) 操作提取帧间的时序信息，然后预测同一物体在两关键帧的位置位置偏移。3.4 节将详细介绍时序处理模块的实现原理。运动插值模块主要是由独立于网络结构的运动插值算法构成，该算法使用三维物体检测模块对两关键帧的检测结果以及时

序处理模块输出的位置偏移信息，将关键帧的检测结果传播到非关键帧，实现流数据的三维物体检测与追踪。该算法的详细流程将会在3.5节重点介绍。

## 3.2 三维物体检测模块

DODT框架的三维物体检测模块融合了点云与图像信息，预测自动驾驶场景中车辆的三维位姿信息。由于该检测模块是基于AVOD[12]网络，为二阶段物体检测模块，因此包含候选框提取网络。由于本框架的候选框提取网络与AVOD有所不同，因此将在下一节详细介绍。本节不展开介绍候选框提取环节。本节将依次介绍数据预处理、特征提取、RoI池化、特征融合以及最终的预测框生成等物体检测中的关键步骤。

### 3.2.1 数据预处理

DODT融合RGB图像数据以及激光雷达数据进行三维物体检测，因此每一帧数据都同时包含图像以及点云。RGB图像数据预处理比较简单，首先统一调整长宽到 $1200 \times 360$ px，然后在RGB通道上将去数据集的RGB平均值 $[R_{mean}, G_{mean}, B_{mean}]$ （本工作基于的KITTI下的物体追踪数据集，RGB平均值为 $[92.84, 97.80, 93.58]$ ）。RGB图像数据经过色彩均值归一化后就可以直接用于后续的特征提取操作了。

点云数据的预处理相对于图像要更为复杂，这是因为点云是稀疏的三维数据，需要经过额外的操作将其转换为网络能够处理的形式。本工作采用网格化的方法将点云数据编码成六通道的BEV特征图，特征图包含五个高度切片通道以及一个点云密度信息通道。我们首先将点云投影到相机平面，然后过滤落在图像尺寸之外的点以便让点云视野与图像视野相同【TODO】。对于截取得到的规则三维视野，我们首先使用 $0.1m$ 分辨率的网格将XZ平面网格化。假设XZ平面尺寸为 $[-W, W] \times [0, L]m$ ，则网格化后的尺寸为 $\frac{2W}{0.1} \times \frac{L}{0.1}$ 。对于高度Y方向，我们截取 $[0, 2.5]m$ 的区域，然后平均切片为五等份从而将整个点云体素化。而后，将每个体素格子中所有点高度的最大值作为该格子的值，从而将点云编码为BEV特征图。对于特征图的最后一个通道，我们使用高度切片前的每个格子内的点计算该格子的点云密度 $\rho = \min(1.0, \frac{\log(N+1)}{\log 16})$ ，其中N为格子内点的数目。该方法最先在MV3D[11]中使用，而后AVOD[12]也使用了相同的处理方式。

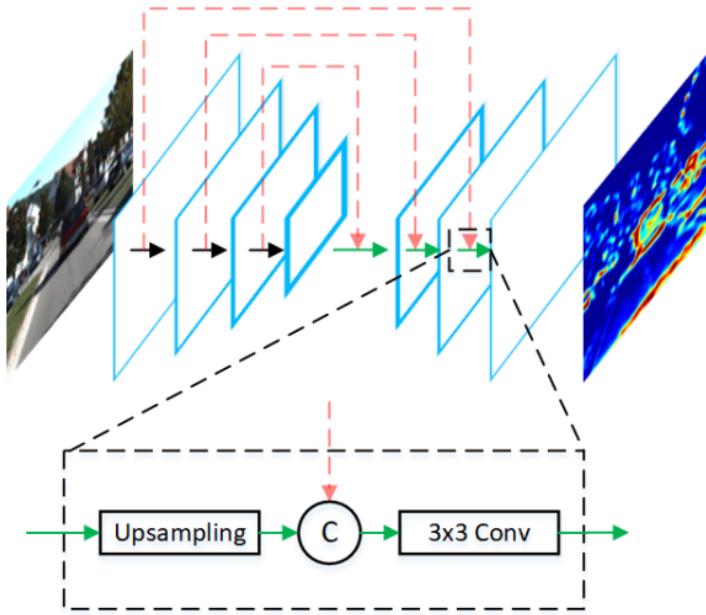


图 3-2 特征提取网络结构。

在自动驾驶场景中，车辆是不停运动的，因此流数据中每一帧数据的参考系也不同。为了更好的关联帧与帧之间的时序信息，除了对每一帧数据进行预处理外，还需将不同帧的数据校准到同一坐标系。DODT 为双路结构，同时处理相邻两帧关键帧数据并关联帧间信息。因此，为了统一两帧数据的坐标系，我们将后一帧关键帧数据校准到前一帧数据的坐标系中。由于图像数据没有准确的三维坐标信息，我们只校准点云数据。点云数据的校准需要用到激光雷达的位姿信息，这些数据 KITTI 数据集有提供。

### 3.2.2 特征提取

DODT 检测模块有两个单独的特征提取网络，分别用于图像特征提取以及点云特征提取。这两个网络的结构类似，只是输入的尺寸不一样。特征提取网络采用的编码器-解码器（Encoder-Decoder）结构，包含了编码器和解码器两部分，如图3-2 所示。编码器基于 VGG-16 网络改造的，首先将网络的通道数减半，并在 conv-4 层截断。编码器输入尺寸为  $M \times N \times D$  的图像数据或是点云 BEV 特征图，然后输出尺寸为  $\frac{M}{8} \times \frac{N}{8} \times D^*$  的特征  $F$ 。特征  $F$  能够表达高层次的语意信息，并且尺寸比输入小了八倍。KITTI 数据集中行人在 BEV 视角平均大小为  $0.8 \times 0.6$  米，在 BEV 特征图上就是  $8 \times 6$  的像素区域（分辨率为  $0.1m$ ）。在编码器八倍下

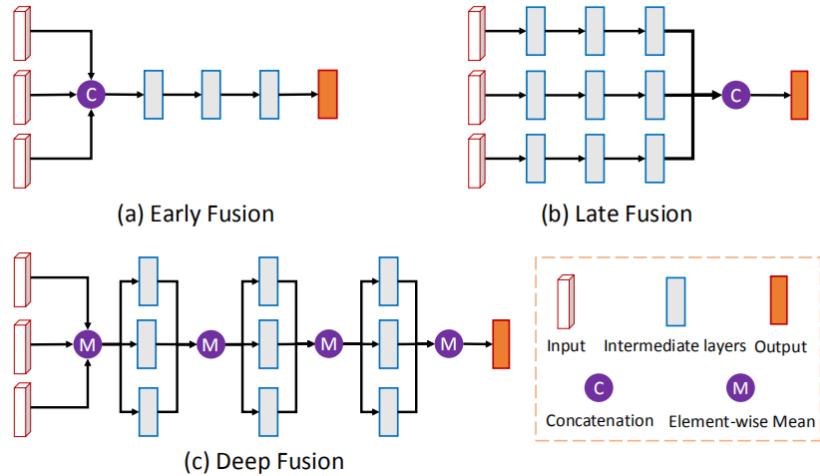


图 3-3 融合网络结构。

采样后，行人在输出特征  $F$  中占据不到一个像素，这还是在不考虑卷积过程中感受野的放大所造成小物体占比缩小的情况。受特征金字塔网络（Feature Pyramid Network）的启发，AVOD 设计了一个自底向上的解码器，解码器可以在几乎不增加运行时间的基础上，将特征  $F$  上采样到输入的尺寸大小。解码器将编码器的输出作为输入，然后生成尺寸为  $M \times N \times D$  的特征图。解码器的上采样是通过反卷积算子实现的，为了更好的补回编码器中下采样造成的细节损失，解码器每一层的输入还额外包含编码器对应层的特征，然后通过一个  $3 \times 3$  的卷积操作融合。最终的特征图在保留了对高层语意的表征能力下，还能有与输入相同的尺寸，这在一定程度上避免了小物体特征丢失的问题。

### 3.2.3 特征融合

在特征提取阶段我们分别得到了图像的特征  $F_{img}$  以及点云的特征  $F_{pc}$ ，接下来就是将这两种特征融合以得到更丰富的视觉特征。DODT 的特征融合也是借鉴了 AVOD 的方式，在候选框层面上进行融合。给定一个 3D 候选框由（Shared RPN 生成），将其投影到  $F_{img}$  以及  $F_{pc}$  上，截取相应的部分就能得到对应的候选区域特征  $f_{img}$  以及  $f_{pc}$ 。之后  $f_{img}$  和  $f_{pc}$  将会通过  $7 \times 7$  的 ROI 池化操作生成相同尺寸的特征向量，之后两特征向量通过一个融合网络生成最终的候选区域特征向量  $F_{fusion}$ 。融合网络结构如图3-3 所示，该网络结构最先由 MV3D 提出，叫

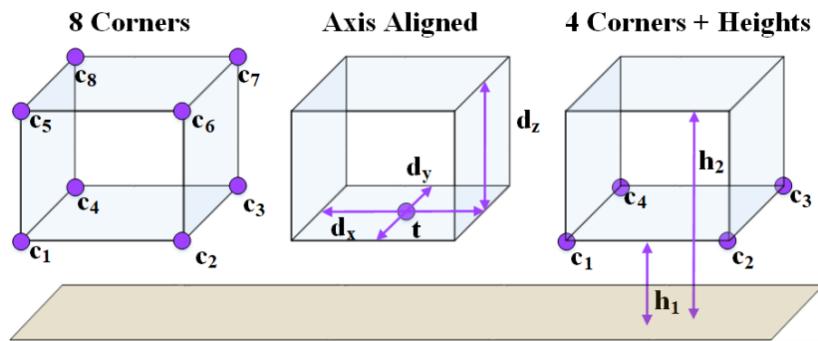


图 3-4 三种三维框编码方式。

做深度融合网络。对于有  $L$  层的网络，深度融合网络按如下方式融合特征：

$$\begin{aligned} F_0 &= F_{img} \oplus F_{pc} \\ F_l &= H_l^{img}(F_{l-1}) \oplus H_l^{pc}(F_{l-1}) \quad (3-1) \\ \forall l &= 1, \dots, L \end{aligned}$$

其中， $\{H_l, l = 1, \dots, L\}$  是特征转移函数（由神经网络拟合）， $\oplus$  为某种融合操作（例如拼接、求和等，DODT 使用的是平均）。深度融合网络可以让两种特征能在更深层次、更高的语义层面进行融合，从而取得更好的融合效果。

### 3.2.4 预测框生成

特征融合之后，融合的特征图将会输入到两个不同任务分支，分别为检测分支和回归分支，这两个分支都是由三层 2048 个神经元组成的全连接层。检测分支负责预测目标的类别，使用交叉熵计算损失值；而回归分支负责预测候选框与真实物体框的差值，使用  $Smooth L1$  函数计算损失值。在回归损失中，只有当候选框与真实框在 BEV 视角的 2D IoU 大于 0.65 才会被计算。之后，DODT 使用 NMS 算法去除重叠的框，阈值设置为 0.01。

在框回归中，对三维框的编码有很多种方式，其中比较常见的有八点编码法和轴对齐（Axis Aligned）编码法，如图3-4 所示。八点编码法直接编码八个顶点的坐标，这种方式没有考虑三维框自身的几何约束，有一定的冗余性。而轴对齐编码限制了三维框要和坐标轴对齐，这是在 RPN 阶段使用的，并不适合最终的框编码。DODT 使用了 AVOD 提出的 10 参数编码法，如图3-4最右侧所示，分别编码底部四点坐标以及底面与顶面和地面的距离，一共有十个参数，因此框回归

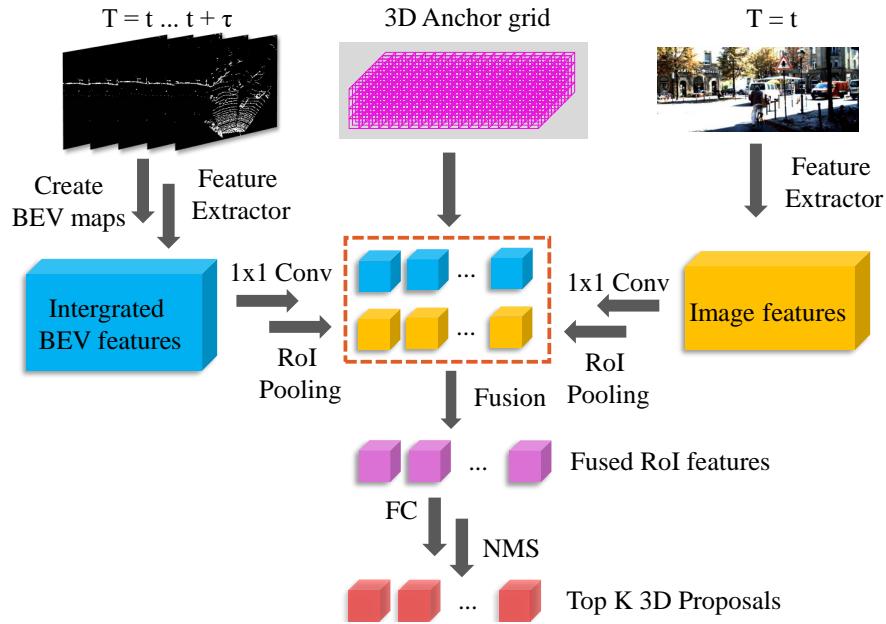


图 3-5 Shared RPN 模块。

的目标为  $(\Delta_{x1}, \dots, \Delta_{y1}, \Delta_{y4}, \Delta_{h1}, \Delta_{h2})$ 。这种编码方式考虑到了三维框顶面四点和底面四点是对齐的，而原来的过参数化的八点法需要编码成 24 维向量。

在 MV3D 中，作者默认三维预测框的朝向为长边方向，可是这种确定朝向的方法是有问题的。首先并不是所有目标都适用于这种确定朝向的方法，比如行人；其次，长边有两个可取的方向，它们相差  $\pm\pi$ 。AVOD 中，作者通过预测转向向量  $(x_\theta, y_\theta) = (\cos(\theta), \sin(\theta))$  来解决这个问题。这种方式可以将每个转向角  $\theta \in [-\pi, \pi]$  都映射到唯一的转向向量。转向向量的预测也是包含在回归分支中。另外，转向向量可以用来解决十参数编码法中预测框与真实框底部四点的对应关系。底部四点的对应有四种，只使用十参数编码时只能通过最近匹配法，有了转向向量之后，转向向量的值可以作为额外的匹配手段。

### 3.3 Shared RPN 模块

DODT 的 *Shared RPN* 模块是以 AVOD 的 RPN 为基础改造的，其结构如图3-5所示。DODT 有两个检测分支，正常来说需要两个单独的 RPN 网络生成各自的候选框。但由于 DODT 处理的是流数据，两分支的关键帧输入有很大的关联性，因为关键帧的特征变化在时间维度上是连续的。受此启发，我们将两分支的 RPN 合并成一个 *Shared RPN*，该 RPN 能够生成供两分支同时使用的 3D 候选框。和

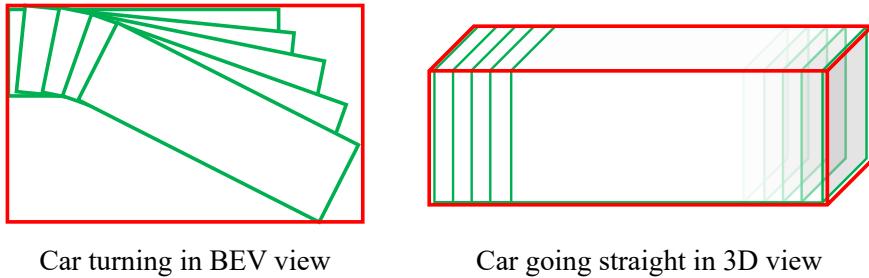


图 3-6 绿色 框是五帧中目标的标签框，红色框是为训练 *Shared RPN* 而新生成的共享候选框标签。

检测模块一样，*Shared RPN* 也融合了点云特征与图像特征来预测 3D 候选框。图像数据的处理和检测模块一样，由于相邻关键帧中图像数据的变化微乎其微，因此我们只使用前一帧关键帧图像数据提取 RGB 特征，特征提取过程和检测模块一样（可以直接使用检测模块处理好的图像特征）。对于点云特征提取，*Shared RPN* 不像检测模块一样只使用一帧点云数据，而是联合了两关键帧以及关键帧之间的点云数据。这是因为点云数据是稀疏的，单帧点云数据不能很好的捕获物体在三维空间的位姿信息，特别是对于远处的物体。但如果将连续时间片段的点云数据联合起来，这在一定层度上是能够增强物体特征的。考虑到在实际驾驶场景中 LIDAR 本身也在不停地运动，造成每帧点云的参考系不一样，直接叠加点云数据效果必然很糟糕，因此需要把这些点云数据校准到同一坐标系上。点云的校准需要用到 LIDAR 在不同时刻的位姿信息，这些可以在 IMU 数据中得到。将连续时间片段的点云数据校准到同一坐标系后，使用和检测分支一样的方式构建 BEV 特征图，然后用相同的特征提取网络提取点云特征。由于点云的稀疏性以及使用体素化的点云编码方法，点云数据的联合并不会带来额外的计算负担，但却可以有效的增强点云特征。

需要额外注意的是，由于物体在不断运动，因此联合的点云数据标签需要重新计算。假设一目标在联合的五帧点云数据中都有出现，那么新的标签应该囊括该目标在这段时间内的运动轨迹。如图3-6所示，我们重新计算了适合 *Shared RPN* 训练的新的候选框标签。新的候选框标签可以同样也是轴对齐的，虽然相比于之前的候选框标签略有增大，但这能够保证覆盖物体在所有融合帧中的运动轨迹，从而使其适合训练生成供两检测分支使用的 3D 候选框。

拿到点云特征与图像特征之后，需要经过和检测模块一样的流程融合两个视角的特征。特征的融合是在锚点框层面的，DODT 锚点框的尺寸是通过对 KITTI 目标检测数据集的标签进行聚类得到。通过将 3D 锚点框投影到两特征图上，然后截取相应的区域再进行 ROI 池化得到特征块，特征块再通过深度融合网络得到最终的融合特征。*Shared RPN* 的特征融合过程和检测模块的基本类似，但是前者的图像特征图与点云特征图在进行 ROI 池化前，都经过了一个  $1 \times 1$  的卷积将特征通道数降为 1。这是因为在某些场景下，锚点框可以多达 100K 个，直接从全通道的特征图上去截取特征块会给 GPU 带来巨大的存储负担。例如，提取 100K 个通道为 256 的  $7 \times 7$  特征块需要将近 5GB 存储 ( $100000 \times 7 \times 7 \times 256 \times 4 \text{ bytes}$ ，假设以 32 位浮点数存储)。此外，处理如此大通道的特征也会让 RPN 耗费巨大的计算资源。因此 *Shared RPN* 通过一个  $1 \times 1$  的卷积操作进行数据降维，能有效减低网络参数量，提升运行效率。而在检测模块中，为追求更高的检测精度，仍维持原来的通道数。

融合网络生成的融合特征最终被用来预测候选框的尺寸和类别。和检测模块类似，有候选框分类和回归两个分支。分类分支是由两层 256 个神经元的全连接层组成，用来预测候选框是目标区域还是背景区域，为二元分类任务。回归分支同样也是由两层 256 个神经元的全连接层组成，回归出预测候选框与真实候选框之间的偏差 ( $\Delta t_x, \Delta t_y, \Delta t_z, \Delta d_x, \Delta d_y, \Delta d_z$ )。候选框的编码方式如图3-4中间所示，包含底部中心点坐标 ( $t_x, t_y, t_z$ ) 以及长宽高 ( $d_x, d_y, d_z$ )。这是因为候选框都是轴对齐的，所有只需要六个参数就能够编码一个三维框。和检测模块一样，分类分支以交叉熵作为损失函数，而回归分支则使用 *Smooth L1* 作为损失函数。当锚点框与真实框在 BEV 视角的 2D IOU 值小于 0.3 时将被视为背景类，而大于 0.5 时被视为目标类，回归损失值的计算不将背景类的损失值。最后，2D NMS 算法会被用于筛选重叠的候选框。网络训练阶段 NMS 算法的阈值是 0.8，保留最好的 1024 个候选框，而在预测阶段则只保留最好的 300 个候选框。

### 3.4 时序信息处理模块

时序信息处理模块输入相邻关键帧的点云 BEV 特征，然后预测两帧中相同物体的位移，其结构如图 3-1 黄色区域所示。关键帧  $t$  以及  $t + \tau$  的特征图经过

ROI 池化后分别得到对应的 RGB 和点云特征块  $(F_{img}^t, F_{pc}^t)$  和  $(F_{img}^{t+\tau}, F_{pc}^{t+\tau})$ ，时序模块只使用点云特征块  $F_{pc}^t$  和  $F_{pc}^{t+\tau}$  预测偏移量。这是因为 DODT 只考虑 BEV 视角的偏移量，因为在自动驾驶场景中，车辆行驶区域基本可以视为二维平面，高度变化不会太大，故只预测车辆在 BEV 视角的偏移量就能够很好的捕获车辆的运动信息。给定两相邻关键帧的点云 BEV 特征块集合  $F_{pc}^t$  和  $F_{pc}^{t+\tau}$ ，时序处理模块首先构造一个特征块对集合  $P = \{(f_i^t, f_i^{t+\tau}), i \in \{0, 1, \dots, N\}\}$ ，其中  $f_i^t$  和  $f_i^{t+\tau}$  分别是  $F_{pc}^t$  和  $F_{pc}^{t+\tau}$  的第  $i$  个特征块， $N$  是 3D 候选框的总数，也是两关键帧特征块的总数。由于两检测分支使用相同的 3D 候选框（由 *Shared RPN* 生成），因此两特征块集合  $F_{pc}^t$  和  $F_{pc}^{t+\tau}$  的元素是相互对应的，因此  $P$  很容易生成。

构建好特征块对集合  $P$  之后，模块对  $P$  的每一个元素  $(f_i^t, f_i^{t+\tau})$  使用相关操作，如公式3-2所示，其中  $p, q \in [-d, d]$  是相关计算的偏移量， $d$  是窗口大小， $j, k$  是窗口中心点坐标。相关特征的输出  $C \in \mathbb{R}^{h \times w \times (2d+1) \times (2d+1)}$ ，其中  $h, w$  是特征块的宽和高。由于总共有  $N$  对特征块对，最终会有  $N$  个相关特征输出。这些特征会输入全连接层进行后续的分类与回归任务。

$$C^{t,t+\tau}(j, k, p, q) = \langle f_i^t(j, k), f_i^{t+\tau}(j + p, k + q) \rangle \quad (3-2)$$

时序信息处理模块有两个输出分支，一个是分类分支，预测该物体是否同时存在在两帧关键帧中；另一个分支是回归分支，预测该物体在两关键帧中的相对位移（BEV 视角）。分类分支由两层 256 个神经单元的全连接层组成，它的输出是一个概率值  $p_{co}$ ，表示两帧中都存在该物体的概率（Object co-occurrence probability），当物体同时存在两帧中时， $p_{co} = 1$ ，否则  $p_{co} = 0$ 。 $p_{co}$  的值可以用来判断物体轨迹的开始或终止，以及是否存在漏检现象。 $p_{co}$  的计算主要是为后续的运动插值模块的正确运行提供额外的保障，这部分内容将在运动插值模块详细介绍。

回归分支的任务比较简单，它也是由两层 256 个神经单元的全连接层组成，负责预测同一个物体在两帧的相对位移。假设  $d^t = (x^t, z^t, \theta^t)$  为第  $t$  帧中一物体的 X 轴和 Z 轴（深度方向）坐标以及转向角，相同的， $d^{t+\tau} = (x^{t+\tau}, z^{t+\tau}, \theta^{t+\tau})$  为  $t + \tau$  帧中该物体相应的数据，则回归分支负责预测  $d^t$  与  $d^{t+\tau}$  的归一化偏差  $\delta^{t,t+\tau}$ ，

如公式3-3所示，

$$\delta^{t,t+\tau} = (\delta_x^{t,t+\tau}, \delta_z^{t,t+\tau}, \delta_\theta^{t,t+\tau}) = \begin{cases} \left(\frac{x^{t+\tau}-x^t}{w^t}, \frac{z^{t+\tau}-z^t}{l^t}, \frac{\theta^{t+\tau}-\theta^t}{\theta^t}\right) & p_{co} = 1.0 \\ (0.0, 0.0, 0.0) & otherwise \end{cases} \quad (3-3)$$

其中  $w^t$  和  $l^t$  分别是物体在第  $t$  帧的宽和长。当  $p_{co}$  为 1.0 时，意味着该物体同时存在两帧中，故可按公式正常计算相对偏差；而当  $p_{co} = 0$  时，意味着该物体只存在某一帧中，这时  $\delta^{t,t+\tau}$  的值将无法计算，我们将其置为 0。在网络训练中，分类分支的损失函数是交叉熵，而回归分支的损失为 *Smooth L1*。

### 3.5 运动插值模块

运动插值模块的任务主要是将两目标检测模块的输出与时序信息处理模块的输出整合，在时序信息的引导下，将关键帧的物体检测结果传播到非关键帧。运动插值模块主要由一个基于运动模型的插值算法（Motion based Interpolation Algorithm, MoI）组成，其流程如算法 2 所示。MoI 算法的输入为第  $t$  帧以及  $t+\tau$  帧的三维物体检测结果  $D^t$  和  $D^{t+\tau}$ ，以及这两帧中各物体的相对偏移  $\Delta^{t,t+\tau}$ 。输出为  $t$  到  $t+\tau$  帧的所有目标检测结果。当  $t$  帧与  $t+\tau$  帧检测的目标个数相同时， $D^t \square D^{t+\tau}$  以及  $\Delta^{t,t+\tau}$  元素个数相同；当有目标不同时存在于  $t$  帧与  $t+\tau$  帧时， $D^t$  与  $D^{t+\tau}$  的元素不相等，而  $\Delta^{t,t+\tau}$  元素个数为  $D^t$  与  $D^{t+\tau}$  的最大值，只是不同时存在前后两帧赶关键帧的目标偏移量为 0。

MoI 算法的主要思想是对于两帧中都存在的目标，使用线性插值算法生成中间帧的结果；对于只存在某一帧的目标，则使用基于时序信息的运动模型生成中间帧结果。具体流程如下：（1）对于  $D^t$  中的每一个预测框  $d_i^t$ ，首先让其加上对应的偏移量  $\Delta d_i^{t,t+\tau}$ （该偏移量由  $\delta_i^{t,t+\tau}$  解码得到），得到预测的运动后的物体框  $d_i^t + \Delta d_i^{t,t+\tau}$ 。然后使用 `getMatched` 匹配函数去  $D_{temp}$  中找到匹配度最高的候选框  $d'$ 。`getMatched` 函数简单的计算  $d_i^t + \Delta d_i^{t,t+\tau}$  与  $D_{temp}$  中所有框的 3D IOU 值，然后在高于某一阈值（具体实现时为 0.5）的 IOU 中筛选出最大者，然后返回其对应的预测框  $d'$ 。（2）如果匹配成功，即  $d'$  存在，即证明该目标在两帧中同时存在，然后算法使用线性插值生成  $t+1$  到  $t+\tau-1$  帧预测框结果，同时将  $d'$  从  $D_{temp}$  中移除，防止后续再次被匹配。（3）对于匹配失败的情况，有两种可能的原因，发生漏检以及轨迹终止。这两种情况的判断可以通过时序信息处理模块的输出  $p_{co}$ ，

**Algorithm 2:** 基于运动模型的插值算法 (MoI Algorithm)

---

```

1 输入:  $D^t = [d_0^t, d_1^t, \dots, d_{N_t}^t]$ ,  $D^{t+\tau} = [d_0^{t+\tau}, d_1^{t+\tau}, \dots, d_{N_{t+\tau}}^{t+\tau}]$ ,
        $\Delta^{t,t+\tau} = [\delta_0^{t,t+\tau}, \delta_1^{t,t+\tau}, \dots, \delta_{N_{max}}^{t,t+\tau}]$ ,  $N_{max} = \max\{N_t, N_{t+\tau}\}$ 
2 输出:  $D = [D^t, D^{t+1}, \dots, D^{t+\tau}]$ 
3 初始化:  $D_{temp} = D^{t+\tau}$ ,  $D$ ,  $p_{co}^{max} = 0.5$ 
4 for  $d_i^t$  in  $D^t$  do
5    $\Delta d_i^{t,t+\tau} = (d_{i,w}^t \cdot \delta_{i,x}^{t,t+\tau}, 0, d_{i,l}^t \cdot \delta_{i,z}^{t,t+\tau}, 0, 0, d_{i,ry}^t \cdot \delta_{i,ry}^{t,t+\tau})$ 
     $d' = getMatched(d_i^t + \Delta d_i^{t,t+\tau}, D_{temp})$ 
6   if  $d'$  存在 then
7      $d_i^{t+1}, \dots, d_i^{t+\tau-1} = Interpolate(d_i^t, d')$ 
8     将  $d'$  从  $D_{temp}$  中移除
9   else if  $p_{co}^i \geq p_{co}^{max}$  then
10     $d_i^{t+1}, \dots, d_i^{t+\tau} = Interpolate(d_i^t, d_i^t + \Delta d_i^{t,t+\tau})$ 
11 else
12  使用运动模型生成  $(d_i^{t+1}, \dots, d_i^{t+\tau-1})$ 
13 if  $D_{temp}$  非空 then
14  for  $d_j^{t+\tau}$  in  $D_{temp}$  do
15    if  $p_{co}^j \geq p_{co}^{max}$  then
16       $d_j^t, \dots, d_j^{t+\tau-1} = Interpolate(d_j^{t+\tau} - \Delta d_j^{t,t+\tau}, d_j^{t+\tau})$ 
17    else
18      使用运动模型生成  $(d_j^{t+1}, \dots, d_j^{t+\tau-1})$ 

```

---

如果  $p_{co}^i \geq p_{co}^{max}$ , 则表明该目标很大概率在  $t + \tau$  中存在, 只是由于检测模型漏检了。这种情况下可通过偏移量信息计算出下一帧对应的物体框为  $d_i^t + \Delta d_i^{t,t+\tau}$ , 之后通过线性插值算法就能够生成中间帧的结果。如果  $p_{co}^i < p_{co}^{max}$ , 则表明轨迹已经终止, 这种情况算法会根据目标的历史运动信息进行适当的轨迹扩展, 下一段将重点介绍该过程。(4) 循环完  $D^t$  中元素后, 有可能  $D_{temp}$  中还存在预测框没

有被匹配。这时也同样会出现两种情况，一种是前一帧中发生了漏检，另外就是该目标刚刚出现，这可以通过  $p_{co}$  的值加以区分。对于漏检的情况，算法会根据偏移信息生成前一帧的预测框，然后再通过线性插值算法生成中间帧的结果；对于目标刚出现的情况，同样的算法使用运动模型进行轨迹扩展。

运动模型的使用是基于假设“物体的速度在短时间内是不变的，与相机自身的运动无关”开展的。在物体运动时，模型会根据物体的历史状态以及运动增量来计算物体在下一时刻的状态。具体而言，模型记录着每条轨迹片段在 BEV 视角的全局速度  $v = (v_x, v_z, v_\theta)$ ，当下一帧有某个物体匹配到该轨迹时，该全局速度会被更新，更新方法如公式3-4所示。其中  $\alpha$  衡量了轨迹下一时刻物体速度对轨迹全局速度的影响，在实际运行  $\alpha = 0.8$ 。对于轨迹的终止，由于 DODT 只对关键帧进行检测，因此我们无法判断轨迹具体在  $t$  帧到  $t + \tau$  帧中的哪一帧终止。因此，对于在  $t$  帧存在而  $t + \tau$  中不存在的物体，我们统一将轨迹向后延伸 3 帧，延伸过程使用该轨迹的全局速度进行插值。如果在延伸过程中轨迹超出点云数据范围，则延伸会提前终止。同样，对于轨迹的起始，轨迹也会根据后续帧的信息进行前向延伸。这样做好处是确保算法不会出现漏检的情况，因为在实际应用中，漏检往往要付出更高的代价。

$$\begin{aligned} v_x &= \alpha v_x + (1 - \alpha) \frac{x^{t+\tau} - x^t}{\tau} = \alpha v_x + (1 - \alpha) \frac{w^t \delta_x^{t,t+\tau}}{\tau} \\ v_z &= \alpha v_z + (1 - \alpha) \frac{z^{t+\tau} - z^t}{\tau} = \alpha v_z + (1 - \alpha) \frac{l^t \delta_z^{t,t+\tau}}{\tau} \\ v_\theta &= \alpha v_\theta + (1 - \alpha) \frac{\theta^{t+\tau} - \theta^t}{\tau} = \alpha v_\theta + (1 - \alpha) \frac{\theta^t \delta_\theta^{t,t+\tau}}{\tau} \end{aligned} \quad (3-4)$$

在具体实验过程中，我们发现物体运动方向的预测有时候在相邻关键帧会发生 180 度的转变，这使得在物体朝向插值过程中出现很大的偏差。为了解决该问题，我们使用了一个简单的朝向校准算法来维持轨迹朝向的连续性。对于一条轨迹  $T_{match}^i$  和下一帧中与该轨迹匹配的预测框  $D_{match}^i$ ，如果这两者的朝向之差大于  $\frac{\pi}{2}$  ( $T_{match}^i$  的朝向由该物体最后时刻的朝向决定)，则将  $D_{match}^i$  的朝向角度加上  $\pi$ ，即将  $D_{match}^i$  旋转 180 度，使其能够与轨迹的朝向基本保持一致，这样就不会出现物体朝向在相邻帧中突然变化很大的情况。

### 3.6 多目标追踪

在时序信息处理模块，我们已经把相邻关键帧间的中间帧的所有检测目标关联起来了，要关联所有帧的检测结果，只需要再将各关键帧关联就行了。有两种方案可以实现关键帧的关联，一种是在输入关键帧对上做文章，例如连续输入关键帧对为  $(i, i + \tau), (i + \tau, i + 2\tau), \dots$ ，这样的话时序信息处理模块就可以将所有帧的检测结果关联；另一种方案是将关键帧结果单独进行数据关联，关联算法可以使用 IOU 匹配法或是卡尔曼滤波。当相邻关键帧的间隔  $\tau$  比较大时，第二种方法的匹配效果会下降，考虑到第一种方案的简便和高效性，DODT 最终使用了第一种方案。由于本算法一次可以完成一个时间窗口内数据的多目标目标跟踪，因此为近似在线（Near Online）算法。

### 3.7 本章总结

本章重点介绍了本工作提出的流数据三维物体检测与跟踪框架 DODT 的结构与原理实现。首先 3.1 节介绍了 DODT 的整体框架，DODT 由四个模块组成，分别是三维物体检测模块、Shared RPN 模块、时序信息处理模块以及运动插值模块。这四个模块分别在 3.2 节、3.3 节、3.4 节以及 3.5 节分别介绍。特别的，本章最后还探讨了基于检测结果的多目标跟踪算法实现。本章内容为本文的重点，DODT 集成了本工作的主要创新。下一章将介绍本工作如何通过具体实验，定性与定量的验证本框架的性能。

## 第4章 实验过程与结果分析

本章将介绍本项目的实验部分，包括对 KITTI 数据集、数据预处理、模型训练以及实验结果分析等内容。在结果分析中，我们首先比较 DODT 各模块对结果的影响，以考察每个模块的有效性；然后我们探讨了关键帧的选取步长对三维物体检测结果的影响，以便确定最优的步长；最后我们也测试了 DODT 在多目标追踪任务上的性能，并与前沿方法对比。结果显示 DODT 框架能很好的完成流数据的三维物体检测以及多目标跟踪任务。

### 4.1 KITTI 数据集介绍

本项目的所有实验都是基于无人驾驶领域中广泛使用的 KITTI 公开数据集开展的。KITTI 数据集是由德国卡尔斯鲁厄理工学院和丰田美国技术研究院联合采集的，该数据集包含多种传感器数据：一个惯性导航系统（GPS/IMU，型号为 OXTS RT 3003）数据，一个激光雷达（Velodyne HDL-64E）数据，两个灰度相机数据（140 万像素）以及两个彩色相机数据（140 万像素）。其中激光雷达扫描频率为 10 帧/秒，相机基本与地平面保持平行，图像采集的尺寸为  $1382 \times 512$  像素。所有传感器的整体布局如图4-1。

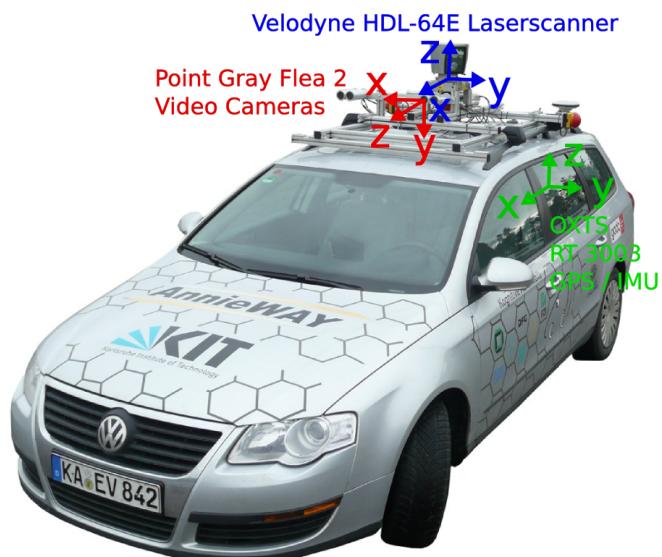


图 4-1 KITTI 数据集传感器整体布局。

KITTI 数据集根据不同的任务分为 stero、flow、scenceflow、depth、odometry、object 以及 tracking 等部分，对应着场景流估计、深度估计、路径规划、物体检测以及目标追踪等任务。每一个任务数据包都包含了海量的训练数据以及测试数据，供研究者使用。本项目主要使用了 KITTI 数据集的 tracking 数据包，该数据包由 21 段训练视频流（共 8004 帧数据）以及 29 段测试视频流（共 11095 帧数据）组成。每一段视频流都是由连续的 RGB 图像帧以及三维点云帧组成，在训练数据集中，还包含了每一帧数据对应的二维目标框（针对图像数据）以及三维目标框（针对点云数据）。此外，针对每一段视频流，KITTI 还提供了传感器的标定信息以及每一帧的 GPS/IMU 数据，供研究者数据标注时使用。

多目标追踪的标签共有 10 项，记录了帧的信息以及帧中每个目标的信息。信息列举如下：

- frame id: 帧的编号；
- object id: 每一帧中目标的编号，也是轨迹的编号；
- type: 目标的类别，有”Car“，”Van“，”Trunk“，”Pedestrian“，”Cyclist“等类别，本实验将”Car“和”Van“合并为”Car“类，并只针对”Car“类进行检测和追踪；
- truncated: 标记目标是否被图像边界截断，”0“表示不截断，”1“表示截断；
- occluded: 标记目标被遮挡的程度，共有 0 到 3 四个取值，”0“表示完全可视，”1“表示部分遮挡，”2“表示大部分遮挡，”3“表示完全遮挡；
- alpha: 目标的观测角， $\alpha \in [-\pi, \pi]$ ；
- bbox: 物体在图像上的 2D 边界框，包含左上角，右下角的坐标值；
- dimensions: 物体的高、宽和长，单位为米；
- location: 物体底部中心点在相机坐标系的三维坐标 x, y, z，单位为米；
- ry: 物体在相机坐标系沿 Y 轴的旋转角， $r_y \in [-\pi, \pi]$ 。

其中目标的观测角  $\alpha = -[(\pi + r_y) + (\pi + \beta)]$ ,  $r_y$  与  $\beta$  如图4-2所示，而目标的 location 坐标如图4-3所示。

传感器的标定信息保存在”calib.txt“文件中，其中包含了相机的参数矩阵以及各传感器之间的旋转矩阵。信息列举如下：

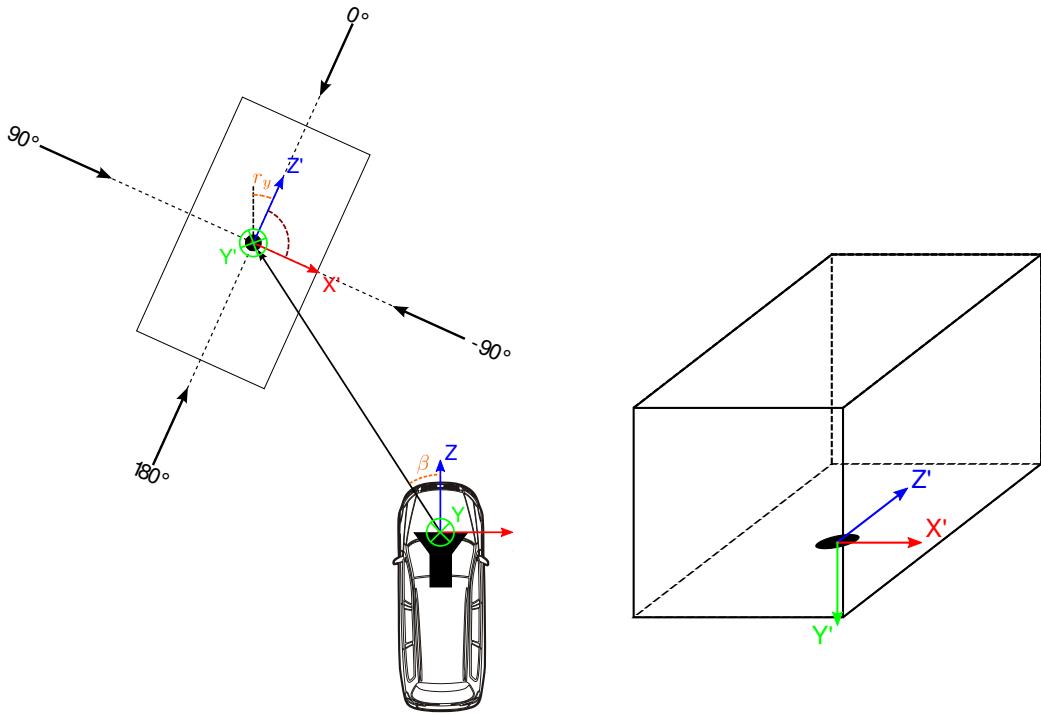


图 4-2 KITTI 数据集观测角与转向角示意图。图 4-3 KITTI 数据集目标的位置坐标。

- P0-P3: 四个相机的内参矩阵  $P \in \mathcal{R}^{3 \times 4}$ ;
- R0\_rect:  $\mathcal{R}^{3 \times 3}$ , 将摄像机坐标系转换到图像坐标系的校准矩阵;
- Tr\_velo\_to\_cam:  $\mathcal{R}^{3 \times 4}$ , 激光雷达坐标系到摄像机坐标系的旋转矩阵;
- Tr\_imu\_to\_velo:  $\mathcal{R}^{3 \times 4}$ , IMU 坐标系到激光雷达坐标系的旋转矩阵。

GPS/IMU 数据提供了 30 项信息，其中包含每一帧中自身车辆的经纬度、海拔、三个欧拉角（roll, yaw 以及 pitch）、速度、加速度、角速度等信息。本实验主要使用到了经纬度以及欧拉角信息将不同帧之间的信息校准到同一坐标系。欧拉角包含了偏航角（yaw，表示机体轴在水平面上的投影与地轴之间的夹角，右偏为正）、俯仰角（pitch，表示机体轴与地平面之间的夹角，抬头为正）以及翻滚角（roll，表示机体对称面绕机轴转动的角度，右滚为正），如图4-4所示。

## 4.2 数据预处理

DODT 框架融合了点云数据信息以及 RGB 图像数据信息，由于激光雷达的视野和摄像机的视野不同，因此在融合之前需要将激光雷达坐标系下的点云转

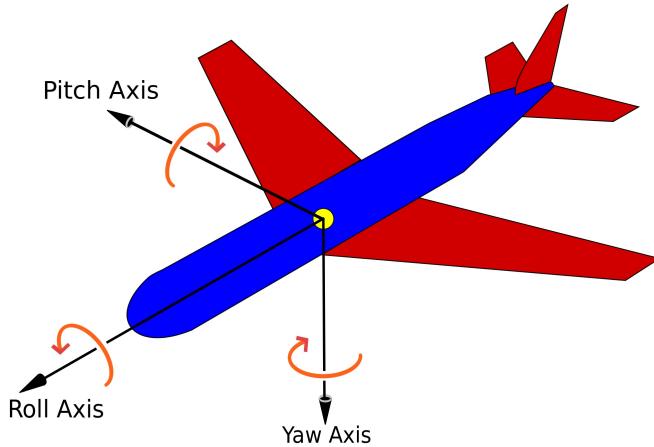


图 4-4 欧拉角示意图。

换到图像坐标系中。假设激光雷达坐标系下点云的齐次坐标为  $P_{pc} = [X, Y, Z, 1]^T$ , 其中  $X = (x_1, x_2, \dots, x_n)$ ,  $Y = (y_1, y_2, \dots, y_n)$ ,  $Z = (z_1, z_2, \dots, z_n)$ , 表示点云中所有  $n$  个点的坐标向量。点云中所有点对应到图像坐标系上的像素点齐次坐标为  $P_{img} = [U, V, 1]^T$  ( $U, V$  同样为坐标向量), 则转换关系如公式4-1所示。其中  $P_{cam}$  为相机的参数矩阵,  $R0\_rect$  以及  $Tr\_velo\_to\_cam$  为上一小节校准文件中的转换矩阵。

$$P_{img} = P_{cam} * R0\_rect * Tr\_velo\_to\_cam * P_{pc} \quad (4-1)$$

经过这步转换后, 点云与图像都在相同坐标系下。之后我们对点云的预处理方式和 AVOD[12]一样, 首先将落在图像尺寸之外的点过滤掉, 然后分别沿 X, Z, Y 轴截取  $[-40, 40] \times [0, 70] \times [0, 2.5]$  米的区域作为最终的点云数据。

为了让 DODT 输入的两帧关键帧数据能够很好的融合从而得到时序信息, 我们还需要将两帧数据校准到同一坐标系, 该过程需要用到上文提及的 GPS/IMU 数据。假设 DODT 输入的相邻关键帧为  $F_1$  和  $F_2$ ,  $F_1$  对应时刻车体的经纬度为  $(lat_1, lon_1)$ , 欧拉角 pitch、roll、yaw 为  $(p_1, r_1, y_1)$ ,  $F_2$  对应时刻车体的经纬度为  $(lat_2, lon_2)$ , 欧拉角 pitch、roll、yaw 为  $(p_2, r_2, y_2)$ 。则  $F_1$  和  $F_2$  坐标系原点的球面距离计算公式如4-2所示, 其中  $R = 6378137.0$  米, 为地球的半径。

$$d = 2R * \arcsin\left(\sqrt{\sin^2\left(\frac{lat_2 - lat_1}{2}\right) + \cos(lat_1)\cos(lat_2)\sin^2\left(\frac{lon_2 - lon_1}{2}\right)}\right) \quad (4-2)$$

为了将  $F_2$  的点云数据转换到  $F_1$  坐标系下, 需要求得两坐标系的位移  $\Delta$  以及旋转矩阵  $M$ 。根据计算的两坐标原点的球面距离以及各自的欧拉角, 位移  $\Delta$  计算如

公式4-3所示，旋转矩阵  $\mathcal{M} = R_Z * R_X * R_Y$ ，计算如公式4-4所示，其中  $\delta_p = (p_2 - p_1)$ ， $\delta_r = (r_2 - r_1)$ ， $\delta_y = (y_2 - y_1)$ ，为两帧车体欧拉角的偏差。假设  $F_2$  中所有点云的原始坐标矩阵为  $P_{origin} = [X, Y, Z]^T \in \mathbb{R}^{n \times 3}$ ，则其转换到  $F_1$  坐标系下的坐标矩阵  $P_{trans} = (P_{origin} + \Delta) * \mathcal{M}$ 。坐标系转换前后两帧点云数据叠加可视化对比如下图所示，由图可知坐标系转换有利于两帧信息的关联。

$$\Delta = [\delta_x, \delta_y, \delta_z]^T = [d \cos(\delta_y), d \sin(\delta_y), d \sin(\delta_p)] \quad (4-3)$$

$$\mathcal{M} = \begin{bmatrix} \cos(\delta_p) & 0 & \sin(\delta_p) \\ 0 & 1 & 0 \\ -\sin(\delta_p) & 0 & \cos(\delta_p) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\delta_r) & -\sin(\delta_r) \\ 0 & \sin(\delta_r) & \cos(\delta_r) \end{bmatrix} \begin{bmatrix} \cos(\delta_y) & -\sin(\delta_y) & 0 \\ \sin(\delta_y) & \cos(\delta_y) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (4-4)$$

在数据可视化时，我们注意到 KITTI 官方提供的多目标追踪标签并不完备。例如，在图4-5第一行中，我们可视化了训练数据集第 0 段视频的几帧数据以及官方提供的标签，可以看到有些目标（由红色虚线框圈出）在 128 帧中有标签，但是在 118 以及 120 帧中却没有标签，尽管这些目标在 118 帧以及 120 帧中都能很好识别。这种情况在整个训练数据集中都经常出现，因此，为了更好的衡量 DODT 模型的检测与追踪性能，我们手动添加了这些漏打的标签。

### 4.3 模型训练

我们的实验在一一台配置有 Tesla P100 的 GPU 以及 Intel(R) Xeon(R) E5-2667 v4 @ 3.20GHz 的 32 核 CPU 的服务器上进行。为了训练并验证 DODT 的性能，我们将 KITTI 提供的 21 段训练视频流分成两份，视频编号为奇数为训练集，偶数的为验证集。我们将官方标记的“Car”以及“Van”目标都当做“Car”，只做该类别的检测与追踪。网络训练过程中的大部分超参数设置我们都借鉴了 AVOD[12]。具体而言，DODT 在训练数据集中迭代次数为 120K，批处理大小为 1。DODT 使用 ADAM[34] 作为优化器，并且使用了初始值为 0.0001，每迭代 30K 步以 0.8 的因子指数下降的可变学习率。网络的损失值由四部分组成，目标检测的分类和回归损失以及时序信息模块的分类与回归损失，如公式4-5所示。其中训练时权重  $w_{cls}, w_{reg}, w_{co}, w_{corr}$  的值分别为 1.0, 5.0, 1.0, 1.0。RPN 的参数设置以及 NMS 算法的阈值设置在第三章方法部分有详细介绍，这里不再赘述。

$$L_{total} = w_{cls} L_{cls} + w_{reg} L_{reg} + w_{co} L_{co} + w_{corr} L_{corr} \quad (4-5)$$

## 4.4 实验结果分析

在实验阶段，我们首先测试了 *Shared RPN* 模块对候选框预测的性能提升，然后使用控制变量法分析了 DODT 各模块对最终检测结果的影响，这些模块包括 *Shared RPN* 模块、时序信息处理模块以及运动插值模块。接着我们探究了不同关键帧选取步长对流数据三维物体检测的性能影响，最后我们探究了不同模块对 DODT 在多目标追踪任务上性能的影响。

### 4.4.1 三维物体检测结果分析

表 4-1 候选框预测性能比较。

方法	原始 RPN	<i>Shared RPN</i>
准确率 (%)	97.81	<b>98.47</b>

RPN 在目标检测任务中用于候选框的提取，为了衡量 *Shared RPN* 相对于原始 RPN 的性能提升，我们比较了两者预测候选框的准确率，结果如表4-1所示。由表格可知，原始的 RPN 模块（直接使用 AVOD[12] 中的 RPN）对候选框的预测准确率为 97.81%，而 *Shared RPN* 的预测准确率为 98.47%，比前者提升了 0.66%。这说明 *Shared RPN* 使用多帧数据能够生成更为准确的候选框。此外，由于 *Shared RPN* 在不引入额外计算量的情况下一次性可生成供两帧使用的候选框，因此其运行帧率是原始 RPN 的两倍。

为了验证各模块对 DODT 三维物体检测任务性能的影响，我们进行了消融实验，实验结果如表4-2 上半部分所示， $\tau = 1$  表示输入的两关键帧为相邻帧。目标检测的评价指标为平均精度（Average Precision, AP），即计算检测正确的目标个数占总目标个数的百分比，然后对目标种类做平均。衡量物体检测是否正确需要计算预测框与真实框的 IOU，若 IOU 大于某一阈值则判断为检测正确。本实验统计了两种不同阈值的实验结果，分别为 0.5 和 0.7。此外，KITTI 根据目标遮挡情况的不同将检测任务分为“Easy”、“Moderate”和“Hard”三类，其中“Easy”对应无遮挡目标 ( $occluded = 0$ ) 的检测结果，“Moderate”对应部分遮挡 ( $occluded = 1$ ) 目标的检测结果，而“Hard”则对应大部分遮挡 ( $occluded = 2$ )

表 4-2 DODT 的不同设置在验证数据集上的结果（只预测“Car”类别）。每项的指标为  $AP_{3D}/AP_{BEV}$  (%)，为三维物体检测在 3D 视角和 BEV 视角的平均精度。“S”表示 *Shared RPN* 模块，“T”表示时序信息处理模块，“M”表示运动插值模块。 $\tau$  是关键帧选取步长。

方法	模块	IOU = 0.5			IOU = 0.7			FPS
		Easy	Moderate	Hard	Easy	Moderate	Hard	
AVOD[12]	-	90.13 / 90.91	80.00 / 81.79	71.61 / 81.79	76.00 / 90.90	57.23 / 81.73	56.13 / 72.69	10.0
DODT( $\tau = 1$ )	S	88.28 / 99.97	85.74 / 90.90	86.14 / 90.89	83.44 / 90.82	67.48 / 90.79	61.24 / 90.80	6.7
DODT( $\tau = 1$ )	S+T	<b>88.32 / 99.99</b>	86.53 / 90.90	86.71 / <b>90.90</b>	83.60 / 90.82	68.93 / 90.80	62.69 / 90.81	5.9
DODT( $\tau = 1$ )	S+M	89.99 / 99.95	87.86 / 90.87	87.81 / 90.86	86.89 / 90.89	73.96 / 90.83	67.07 / 81.79	6.5
DODT( $\tau = 1$ )	S+T+M	<b>90.63 / 99.95</b>	89.07 / 90.90	88.79 / <b>90.90</b>	88.74 / 90.91	75.27 / 90.84	68.75 / 90.57	5.7
DODT( $\tau = 2$ )	S+T+M	90.60 / 99.94	<b>89.19 / 90.91</b>	<b>88.91 / 90.88</b>	<b>88.90 / 90.92</b>	<b>76.64 / 90.85</b>	75.81 / 90.83	8.6
DODT( $\tau = 3$ )	S+T+M	90.61 / 99.98	89.01 / 90.89	88.84 / 90.89	88.81 / 90.91	76.38 / <b>90.86</b>	<b>75.83 / 90.85</b>	11.4
DODT( $\tau = 4$ )	S+T+M	90.55 / 99.94	88.82 / 90.88	88.34 / 90.87	88.43 / 90.91	75.70 / 90.82	68.75 / 90.82	14.3
DODT( $\tau = 5$ )	S+T+M	87.98 / 90.91	85.57 / 90.87	86.01 / 90.87	81.59 / 90.81	67.30 / 90.76	61.35 / 81.73	17.1
DODT( $\tau = 6$ )	S+T+M	78.77 / 90.75	70.88 / 90.71	71.65 / 81.70	71.71 / 90.44	55.86 / 81.50	56.80 / 81.51	<b>20.0</b>

目标的检测结果。

*Shared RPN* 的对比结果对应与表格第一行与第二行，在使用其他新模块的基础上，使用原始 RPN 模块的 DODT 相当于 AVOD 架构，因此我们直接对比了 AVOD 和只使用了 *Shared RPN* 模块的 DODT ( $\tau = 1$ ) 的实验结果。由结果可知，*Shared RPN* 模块基本在所有指标上都能带来检测性能的提升。特别的，对于  $IOU = 0.7$  的阈值下，*Shared RPN* 在“Easy”、“Moderate”和“Hard”检测级别上分别为  $AP_{3D}$  带来了 7.44%、10.26% 和 5.11% 的性能提升。这些显著的性能提升要归功于 *Shared RPN* 模块能够结合多帧信息预测更为精确的候选框。另外，从结果可知使用 *Shared RPN* 可以显著提高检测速度，这主要是相邻两帧复用了候选框，节省了不少时间。

在基础 DODT 框架（双路结构 + *Shared RPN*）的基础上，我们加入了时序信息处理模块（T），实验结果如表4-2第三行所示。对比第二行的基础 DODT 框架结果，在  $IOU=0.7$  的阈值下，时序信息处理模块在“Easy”、“Moderate”和“Hard”检测级别上分别为  $AP_{3D}$  带来了 0.16%、1.45% 和 1.45% 的性能提升。结果显示时序信息模块能够提升遮挡目标的检测性能，这是因为时序模块能够融合两帧关键帧的信息，目标在某一帧被遮挡的话很有可能在下一帧不被遮挡。由

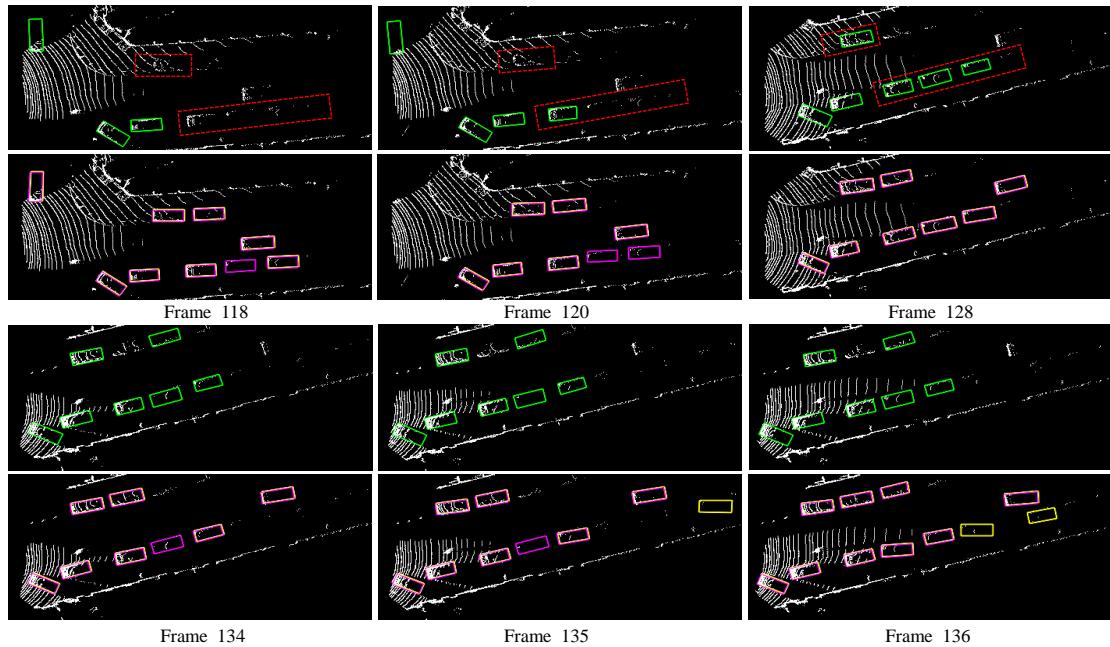


图 4-5 视频序列 0 的标签以及预测结果可视化。绿色为官方提供的标签框，黄色是时序步长  $\tau = 1$  的预测结果，洋红色为时序步长  $\tau = 3$  的预测结果。混合颜色的框是黄色框和洋红色框重叠造成的，该结果最好彩色打印查看。

于在 BEV 视角上目标遮挡很少，且候选框层次的特征融合让两关键帧信息可以互补，使得 DODT 对于困难样本的预测更加准确。

为探究运动插值模块对 DODT 目标检测性能的影响，我们在基础 DDOT 框架的基础上加入了运动插值模块 (M)，实验结果如表4-2第四行所示。结果显示在  $IOU=0.5$  的阈值下，运动插值模块在“Easy”、“Moderate”和“Hard”检测级别上分别为  $AP_{3D}$  带来了 1.71%、2.12% 和 1.67% 的性能提升；而在  $IOU=0.7$  的阈值下则分别带来了 3.45%、6.48% 和 5.83% 的性能提升。这些结果表明运动插值模块能够显著提高 DODT 对车辆的检测性能，特别是对于困难目标的检测。对比不同 IOU 阈值的结果还可以看出，运动插值模块能够显著改善预测框偏离真实框较大的情况。这些提升显示了我们的 MoI 算法能够根据前后帧的检测结果筛选检测错误的样本，并且能够通过轨迹的历史状态进一步延伸轨迹。此外，时序信息处理模块和运动插值模块联合使用的话能够额外提升 12% 的性能，如表格4-2第五行所示。这是因为运动插值模块在时序信息处理模块输出结果的基础上能够运行的更好，详情可参阅第三章 3.5 小节中 MoI 算法的介绍。

#### 4.4.2 流数据物体检测结果分析

DODT 框架同时处理两帧相邻的关键帧，非关键帧的检测结果是通过关键帧结果插值得到，因此关键帧的选取对于流数据物体检测影响很大。由于在三维空间中，车辆是连续运动，因此不存在像图像数据中的突变现象，每一帧都可作为关键帧使用。但是输入两帧关键帧的时间跨度对 DODT 的检测性能影响很大，如果步长太小，则会造成检测速度慢，时序信息处理变得很累赘；如果步长太大，则两帧关键帧数据信息关联度不高，时序信息提取困难。为了探索最佳的关键帧选取步长  $\tau$ ，我们一系列对比实验。在完整版 DODT 框架（*Shared RPN + 时序信息处理模块 + 运动插值模块*）框架的基础上，通过改变步长  $\tau$  的值 ( $\tau \in \{1, 2, 3, 4, 5, 6\}$ )，我们得到了六组不同的实验结果，如表4-2下半部分所示。从结果可以看出，IOU=0.7 的阈值下，DODT ( $\tau = 2$ ) 在“Easy”和“Moderate”检测级别上表现最佳，分别为 88.90% 和 76.64% ( $AP_{3D}$ )。而 DODT ( $\tau = 3$ ) 在“Hard”检测级别上表现最佳，为 75.84% ( $AP_{3D}$ )，并且在“Easy”和“Moderate”检测级别上也与  $\tau = 2$  时有相当的性能。对比于 DODT ( $\tau = 1$ )，当  $\tau = \{2, 3\}$  时 DODT 在困难样本上的检测性能显著提升，“Moderate”任务上提升约 1% 而“Hard”任务提升有约 7%。这说明当选取一个较大的步长时，DODT 对遮挡目标的检测能力显著提高。这些提升很大程度上是由 MoI 算法带来的，我们可视化了几帧数据，如图4-5下半部分所示。其中洋红色的预测框为 DODT ( $\tau = 3$ ) 的预测结果，黄色框为  $\tau = 1$  的预测结果。可以看出，较大的步长能够筛选错误的检测结果，并且也能够更好的补全轨迹两端缺失的目标。

然而并不是步长  $\tau$  越大检测结果越好，从表4-2可以看出，当  $\tau = \{4, 5, 6\}$  时，DODT 的检测性能下降很快。这一方面是因为长距离的时序信息更难捕捉，另一方面是因为大步长不利于 MoI 算法对非关键帧的插值以及轨迹的延伸。当步长太长时，两关键帧之间有很多非关键帧，目标的跳跃较大，不利于插值算法的运行。我们也计算了不同步长时 DODT 的运行帧率，结果如表4-2最后一列所示。DODT 两三维目标检测分支的总运行时间为 175ms/帧（Tesla P100 GPU），当步长  $\tau$  增大时，检测分支以及时序信息处理模块的运行时间保持不变，而插值算法的运行时间增加可忽略不计，因此总运行时间基本不变，但是帧率却按倍数增

表 4-3 DODT 的不同设置在 KITTI 多目标追踪验证数据集上的结果。S 表示 *Shared RPN* 模块，T 表示时序信息处理模块，M 表示运动插值模块。 $\tau$  是关键帧选取时间步长。

方法	模块	MOTA(%) $\uparrow$	MOTP(%) $\uparrow$	MT(%) $\uparrow$	ML(%) $\downarrow$	IDS $\downarrow$	FM $\downarrow$
AVOD[12]	-	66.05	82.97	46.22	12.18	<b>2</b>	113
DODT( $\tau = 3$ )	S	76.53	<b>83.93</b>	68.91	7.14	32	80
DODT( $\tau = 3$ )	S+T	77.52	83.75	69.33	7.56	37	77
DODT( $\tau = 3$ )	S+M	78.73	<b>83.93</b>	68.49	9.55	<b>2</b>	<b>48</b>
DODT( $\tau = 3$ )	S+T+M	<b>79.72</b>	83.55	<b>71.85</b>	<b>5.46</b>	7	66

大。权衡了运行时间以及检测精度，我们最终选择  $\tau = 3$  作为最终的模型步长。

#### 4.4.3 多目标跟踪实验分析

在多目标跟踪实验中，我们选取了  $\tau = 3$  作为最终的关键帧选取步长，并且和目标检测实验一样探索了 *Shared RPN* 模块、时序信息处理模块以及运动插值模块对跟踪性能的影响，实验结果如表4-3所示。跟踪性能使用在第二章中介绍过的六种指标衡量，分别是 MOTA、MOTP、MT、ML、IDS 以及 FM。对于不使用运动插值模块的实验，我们使用线性插值算法进行预测框的传播，然后使用基于 IOU 的框匹配算法生成轨迹。从实验结果可以看出：（1）相比于 AVOD，*Shared RPN* 模块的加入几乎给所有指标带来了显著提升，特别的，MOTA 指标提升 10.48%，MT 指标提升 22.69%。这是因为使用插值生成非关键帧的预测结果可以减少目标突变，不过由于没有物体运动信息的支持，轨迹的连续性会降低，这点在 IDS 指标的下降可以看出。（2）时序信息处理模块的加入进一步提升了 MOTA 和 ML 指标，这是因为时序信息的加入使得预测框的传播更为准确。不过由于传统的基于 IOU 匹配的算法不能很好利用时序信息，因此造成 IDS 指标的进一步升高。（3）运动插值模块可使所有指标显著提升，特别是在 IDS 与 FM 两个指标上，性能超过了 AVOD。这是因为我们的运动插值模块有专门处理时序信息的机制，能够根据帧间信息去除假正例检测结果并进行轨迹的扩展。（4）相比于原始的 AVOD 模型，完整版 DODT 模型在 MOTA 指标上提升了 13.67%，MOTP 指标提升 0.58%，MT 指标提升了 25.63%，ML 提升了 6.72%。虽然 IDS

表 4-4 DODT 与现有的前沿方法在 KITTI 三维多目标追踪公开排行榜中的对比。FPS 的计算不包含目标检测时间。

方法	MOTA(%)↑	MOTP(%)↑	MT(%)↑	ML(%)↓	IDS↓	FM↓	FPS↑
DSM[19]	76.15	83.42	60.00	8.31	296	868	10.0 (GPU)
3DT[35]	<b>84.52</b>	<b>85.64</b>	<b>73.38</b>	<b>2.77</b>	377	847	33.3
Complexer-YOLO[8]	75.70	78.46	58.00	5.08	1186	2096	<b>100.0</b>
3D-CNN/PMBM[18]	80.39	81.26	62.77	6.15	121	613	71.4
DODT(ours)	76.68	81.65	60.77	11.69	<b>63</b>	<b>384</b>	76.9

略有变差，但是 FM 指标略有提升。整体而言，我们的 DODT 模型在多目标检测任务上相比原始的 AVOD 模型有很大的提升。

最后我们在 KITTI 的多目标跟踪测试数据集上比较了 DODT 方法和三维目标检测的前沿方法，结果如表4-4所示。可以看到，在 IDS 与 FM 指标上，我们的方法要显著优于其他方法，这得益于 DODT 的运动插值模块对假正例的筛选以及对轨迹两端的补全。对于 MOTA 和 MT 指标，我们的方法要优于 Complexer-YOLO 和 DSM，但是比 3D-CNN/PMBM 和 3DT 差；对于 MOTP 和 ML，DODT 也不及 3D-CNN/PMBM 和 3DT。需要注意的是，3D-CNN/PMBM 使用了复杂的 PMBM 滤波器进行数据关联，而我们只是使用非常简单的基于 IOU 的数据匹配算法；而 3DT 使用了在仿真环境中额外采集的数据集进行训练，我们的方法只使用了 KITTI 官方提供的数据集进行训练。另外，测试数据集上也可能存在和训练数据集一样的标签缺失，这对于我们的方法来说是一种劣势。因此如果完善标签，DODT 将取得更好的结果。在运行时间上，我们的方法也取得了非常可观的性能，每秒可处理 76.9 帧，仅次于 Complexer-YOLO。注意这里的运行时间不包括目标检测过程，但是我们很难去分别统计 DODT 的 MoI 算法中预测框传播与数据关联的时间，因此这里的 DODT 的帧数是 MoI 算法的运行帧数。

## 4.5 结果展示

这一小节我们选择了测试数据集中四段视频的连续四帧进行可视化，有 BEV 视角，3D 视角以及图像视角。相同颜色表示同一辆车在不同时间的状态，最好在彩色图的基础上观察。

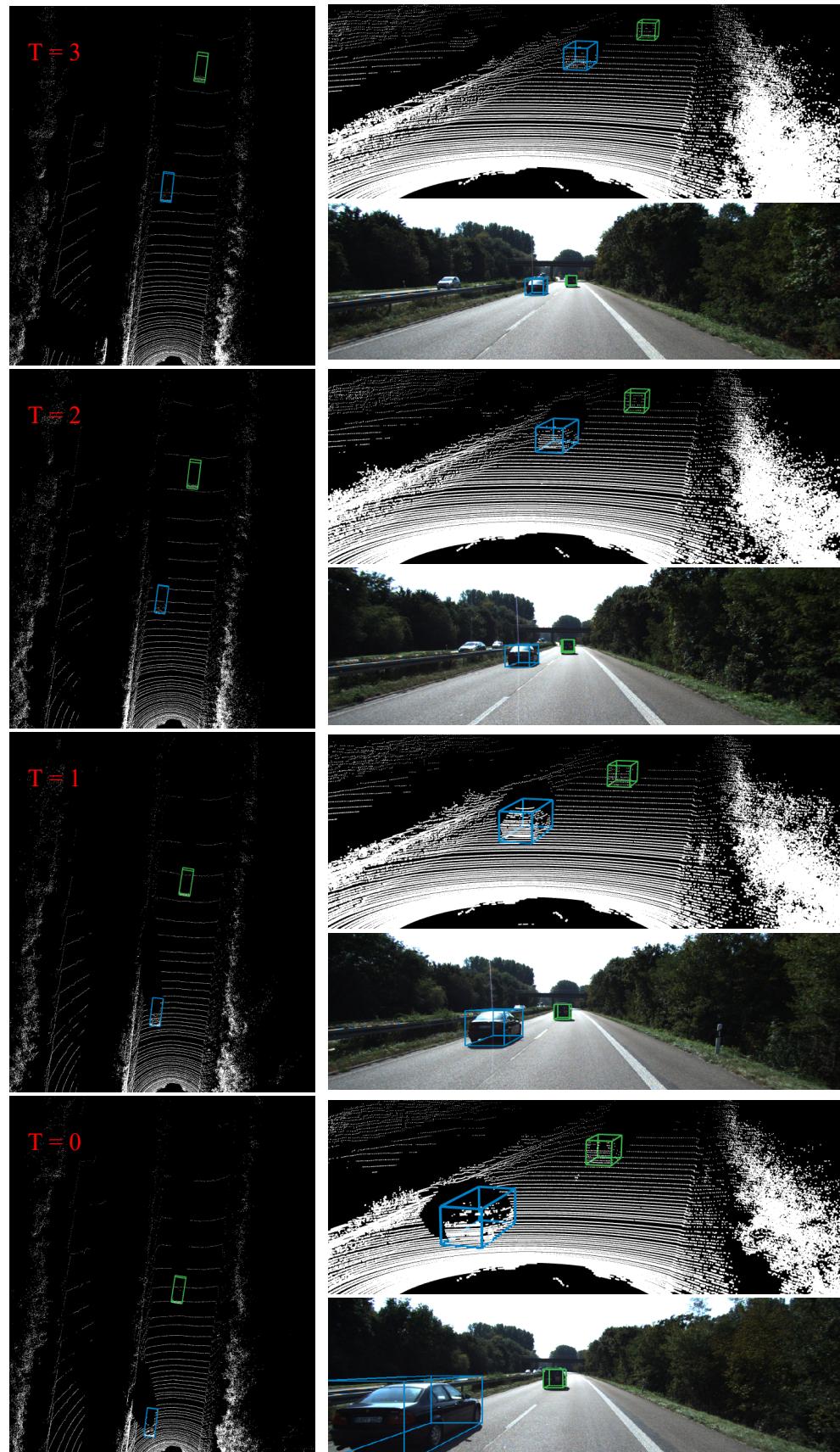


图 4-6 KITTI 多目标追踪测试集视频片段 6 的一段轨迹结果。

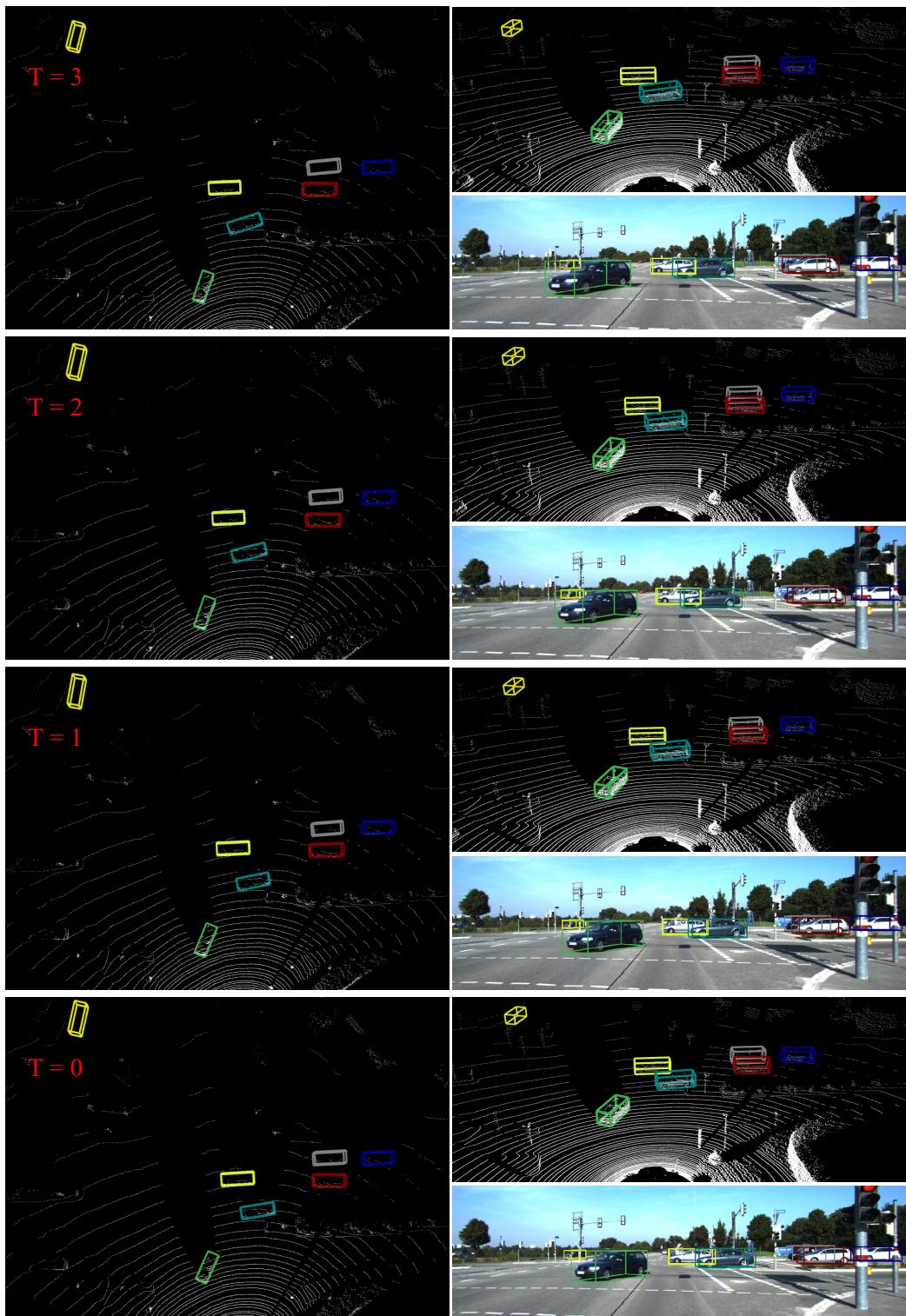


图 4-7 KITTI 多目标追踪测试集视频片段 10 的一段轨迹结果。



图 4-8 KITTI 多目标追踪测试集视频片段 11 的一段轨迹结果。

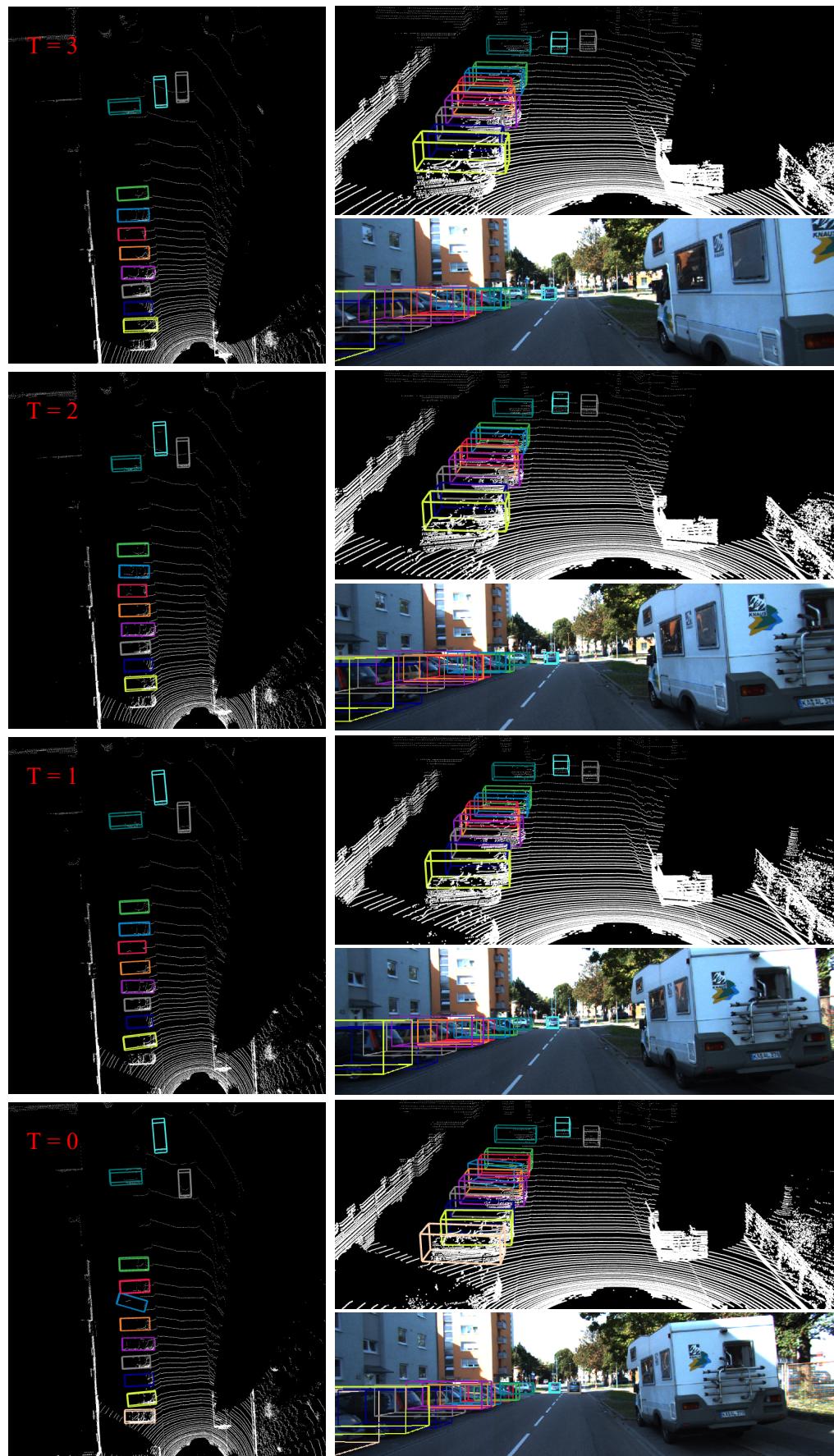


图 4-9 KITTI 多目标追踪测试集视频片段 12 的一段轨迹结果。

## 4.6 本章总结

本章我们首先介绍了实验中用于训练 DODT 的 KITTI 数据集，包括它的传感器设置以及数据格式（4.1 节）；然后介绍了实验过程中使用到的数据预处理，主要是点云数据的裁剪与配准（4.2 节）；之后简单说明了 DODT 训练的超参数（4.3 节）；再之后本章花了大量笔墨分析实验结果，包括 *Shared RPN* 模块、时序信息处理模块以及运动插值模块对 DODT 模型在三维目标检测以及多目标跟踪任务中的性能影响，并分析了这些影响的原因；最后我们还将 DODT 在 KITTI 跟踪测试数据集上的性能与当前最前沿的方法进行比较，阐述了 DODT 框架在三维流数据物体跟踪中的优势（4.4 节）。在本章结束之前，我们还提供了几段 DODT 方法在 KITTI 测试数据集上的可视化结果，以便读者能够更好的了解 DODT 的性能（4.5 节）。

## 第 5 章 总结与展望

### 5.1 全文总结

本文提出了一个双路物体检测与跟踪（Dual-way Object Detection and Tracking, DODT）框架，旨在将三维物体检测从单帧推广到多帧连续场景，从而推动前沿三维物体检测算法在自动驾驶领域的落地。DODT 的主要思想是基于流数据的连续性以及冗余性，通过只对关键帧做物体检测，然后在时序信息的引导下将关键帧检测结果传播到非关键帧，最后再将帧间数据关联，完成多目标追踪任务。在此思想的引导下，本文分了四部分详细介绍了 DODT 的研究背景、研究基础、原理实现以及实验验证。在第一章中，本文详细阐述了近年来国内外在三维目标检测领域的进展，并分析了各流派的优缺点，以及将单帧方法推广到多帧流数据场景的重要意义。第二章则详细介绍了基于深度学习技术的目标检测技术进展，重点介绍了以 Faster-RCNN 为代表的两阶段目标检测以及以 YOLO 为代表的单阶段目标检测的原理和实现方式，为读者了解 DODT 的原理提供基础参考，之后本章还简要介绍了目标追踪的方法进展以及多目标追踪的性能衡量指标。本文第三章开始重点介绍了 DODT 的网络架构与实现原理，先后详细介绍了组成 DODT 的四个基本模块：三维物体检测模块、*Shared RPN* 模块、时序信息处理模块和运动插值模块，以及这些模块如何相互合作完成流数据的物体检测与多目标跟踪。第四章阐述了 DODT 的实验验证环节，该章首先介绍了实验所用的 KITTI 数据集以及数据预处理步骤，然后使用控制变量法分析了 DODT 各模块对最终检测与追踪性能的影响，并分析出了最佳的关键帧选取步长。另外，该章还介绍了 DODT 在三维多目标追踪领域与前沿方法的性能对比，结果表明 DODT 能够取得相当可观的性能。

### 5.2 展望

DODT 虽然在流数据物体检测与跟踪任务取得了显著的效果，但它离落实到自动驾驶场景还有一段距离。DODT 框架的落地还需要解决以下四个问题：

- 目前的框架只是局限与 AVOD 检测模型，能否将其扩展到能够嵌入任何三维物体检测模型？
- 目前 DODT 是以近似在线跟踪的方式实现多目标跟踪，能否在 DODT 的基础上设计出在线的三维多目标跟踪算法？
- DODT 模型在长关键帧选取步长上效果不佳，是否有更为合理的关键帧选取算法？是否能够有更加高效的时序信息提取方式？
- DODT 能否迁移到嵌入式设备上？能否使用模型压缩方法进一步提高运行效率？

本项目后续工作将重点从这四个方面入手，继续改进现有的 DODT 框架。对于第一个问题，设计适应性更广，扩展性更强的 DODT 框架，目前我们已开展相关工作。由于三维物体检测领域算法日新月异，目前 KITTI 排行榜的前几位算法都是只基于点云数据的，并且也有先进行点云分割再回归目标框的算法。因此扩展版的 DODT 应将这些算法囊括进来，构建一个通用的三维流数据物体检测框架。在该基础上，在线多目标跟踪算法、关键帧选取算法以及更加高效的时序信息提取算法的探索可以同步进行。DODT 的嵌入式迁移是本项目最后的工作，目前我们考虑了参数压缩以及二值化网络的方案。只有将 DODT 在嵌入式设备上运行，才能够真正实现算法落地。

## 参考文献

- [1] Chen X, Kundu K, Zhang Z, *et al.* Monocular 3D Object Detection for Autonomous Driving [C]. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016: 2147–2156.
- [2] Chen X, Kundu K, Zhu Y, *et al.* 3d object proposals using stereo imagery for accurate object class detection [J]. IEEE transactions on pattern analysis and machine intelligence, 2018, 40 (5): 1259–1272.
- [3] Li B. 3d fully convolutional network for vehicle detection in point cloud [C]. In 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017: 1513–1518.
- [4] Engelcke M, Rao D, Wang D Z, *et al.* Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks [C]. In 2017 IEEE International Conference on Robotics and Automation (ICRA), 2017: 1355–1361.
- [5] Zhou Y, Tuzel O. Voxelenet: End-to-end learning for point cloud based 3d object detection [C]. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018: 4490–4499.
- [6] Yang B, Luo W, Urtasun R. Pixor: Real-time 3d object detection from point clouds [C]. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018: 7652–7660.
- [7] Simon M, Milz S, Amende K, *et al.* Complex-YOLO: An Euler-Region-Proposal for Real-Time 3D Object Detection on Point Clouds [C]. In European Conference on Computer Vision, 2018: 197–209.
- [8] Simon M, Amende K, Kraus A, *et al.* Complexer-YOLO: Real-Time 3D Object Detection and Tracking on Semantic Point Clouds [C]. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, June 2019.
- [9] Shi S, Wang X, Li H. Pointrcnn: 3d object proposal generation and detection from point cloud [C]. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019: 770–779.
- [10] Qi C R, Liu W, Wu C, *et al.* Frustum pointnets for 3d object detection from rgb-d data [C]. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018: 918–927.
- [11] Chen X, Ma H, Wan J, *et al.* Multi-view 3d object detection network for autonomous driving [C]. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017: 1907–1915.
- [12] Ku J, Mozifian M, Lee J, *et al.* Joint 3d proposal generation and object detection from view aggregation [C]. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018: 1–8.
- [13] Zhu X, Wang Y, Dai J, *et al.* Flow-guided feature aggregation for video object detection [C]. In Proceedings of the IEEE International Conference on Computer Vision, 2017: 408–417.

- [14] Han W, Khorrami P, Paine T L, *et al.* Seq-nms for video object detection [J]. arXiv preprint arXiv:1602.08465, 2016.
- [15] Feichtenhofer C, Pinz A, Zisserman A. Detect to track and track to detect [C]. In Proceedings of the IEEE International Conference on Computer Vision, 2017: 3038–3046.
- [16] Lenz P, Geiger A, Urtasun R. FollowMe: Efficient online min-cost flow tracking with bounded memory and computation [C]. In Proceedings of the IEEE International Conference on Computer Vision, 2015: 4364–4372.
- [17] Luo W, Yang B, Urtasun R. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net [C]. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018: 3569–3577.
- [18] Scheidegger S, Benjaminsson J, Rosenberg E, *et al.* Mono-camera 3d multi-object tracking using deep learning detections and pmbm filtering [C]. In 2018 IEEE Intelligent Vehicles Symposium (IV), 2018: 433–440.
- [19] Frossard D, Urtasun R. End-to-end Learning of Multi-sensor 3D Tracking by Detection [C]. In 2018 IEEE International Conference on Robotics and Automation (ICRA), 2018: 635–642.
- [20] Dosovitskiy A, Fischer P, Ilg E, *et al.* Flownet: Learning optical flow with convolutional networks [C]. In Proceedings of the IEEE international conference on computer vision, 2015: 2758–2766.
- [21] Bolme D S, Beveridge J R, Draper B A, *et al.* Visual object tracking using adaptive correlation filters [C]. In 2010 IEEE computer society conference on computer vision and pattern recognition, 2010: 2544–2550.
- [22] Henriques J F, Caseiro R, Martins P, *et al.* High-speed tracking with kernelized correlation filters [J]. IEEE transactions on pattern analysis and machine intelligence, 2014, 37 (3): 583–596.
- [23] Danelljan M, Häger G, Khan F, *et al.* Accurate scale estimation for robust visual tracking [C]. In British Machine Vision Conference, Nottingham, September 1-5, 2014, 2014.
- [24] Danelljan M, Robinson A, Khan F S, *et al.* Beyond correlation filters: Learning continuous convolution operators for visual tracking [C]. In European conference on computer vision, 2016: 472–488.
- [25] Danelljan M, Bhat G, Shahbaz Khan F, *et al.* Eco: Efficient convolution operators for tracking [C]. In Proceedings of the IEEE conference on computer vision and pattern recognition, 2017: 6638–6646.
- [26] Bertinetto L, Valmadre J, Henriques J F, *et al.* Fully-convolutional siamese networks for object tracking [C]. In European conference on computer vision, 2016: 850–865.
- [27] Li B, Yan J, Wu W, *et al.* High performance visual tracking with siamese region proposal network [C]. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018: 8971–8980.
- [28] Zhu Z, Wang Q, Li B, *et al.* Distractor-aware siamese networks for visual object tracking [C]. In Proceedings of the European Conference on Computer Vision (ECCV), 2018: 101–117.
- [29] Bewley A, Ge Z, Ott L, *et al.* Simple online and realtime tracking [C]. In 2016 IEEE International Conference on Image Processing (ICIP), 2016: 3464–3468.

- [30] Wojke N, Bewley A, Paulus D. Simple online and realtime tracking with a deep association metric [C]. In 2017 IEEE international conference on image processing (ICIP), 2017: 3645–3649.
- [31] Bochinski E, Eiselein V, Sikora T. High-speed tracking-by-detection without using image information [C]. In 2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), 2017: 1–6.
- [32] Bochinski E, Senst T, Sikora T. Extending IOU based multi-object tracking by visual information [J]. AVSS. IEEE, 2018.
- [33] Bernardin K, Stiefelhagen R. Evaluating multiple object tracking performance: the CLEAR MOT metrics [J]. EURASIP Journal on Image and Video Processing, 2008, 2008: 1–10.
- [34] Kingma D P, Ba J. Adam: A method for stochastic optimization [J]. arXiv preprint arXiv:1412.6980, 2014.
- [35] Hu H-N, Cai Q-Z, Wang D, *et al.* Joint Monocular 3D Detection and Tracking [C]. 2019.

## 攻读硕士学位期间发表学术论文情况

1. 3D Object Detection and Tracking Based on Streaming Data, ICRA 2020, CCF B。 (第一作者) (与学位论文第三、四章相关)
2. 一种基于流数据的三维物体检测与跟踪方法。发明专利, 已公开。专利号: 201910725207.8。第二发明人, 导师为第一发明人。
3. 一种融合图像分割与分类的细胞图像语意分割方法。发明专利, 已公开。专利号: 201910819365.X。第二发明人, 导师为第一发明人。
4. 一种基于关键帧的三维物体检测与跟踪方法。发明专利, 已公开。专利号: 201910818311.1。第二发明人, 导师为第一发明人。

## 致 谢

时光荏苒，白驹过隙，转眼间又一个三年过去了。三年前，作为跨专业保研生，我对计算机相关的基础知识一知半解，对计算机领域的前沿进展更是一无所知。很庆幸黄凯老师能够毫无顾虑地接纳我为实验室的一份子，在学习和生活上都尽心尽责帮助我。本项目是在黄凯老师的悉心指导下完成的，黄凯老师对问题的深刻见解以及对科研的严谨态度给我留下了深刻的印象，使我受益终身。

在攻读硕士学位的三年里，很庆幸能够得到很多人的帮助，使我不至于因为自身基础知识的薄弱而在“巨大”的科研压力面前丧失信心，自暴自弃。在此，我要感谢康德开、陈胜杰以及张文权三位师兄，在我研究生第一年时，是他们在学习和生活上给予我众多帮助，让我能够尽快融入实验室的生活，尽快开展科研任务。特别地，我要感谢康德开师兄带领我开展我的第一个科研项目，并悉心指导我实验细节，教会我做研究的方法。在本项目中，我要感谢古剑锋、郭思璐、杨铖章以及许子潇同学的全心投入，特别是古剑锋同学为本项目结果可视化所付出的努力。另外，我还要感谢白善荣同学在我会议论文撰写期间帮忙反复校对文章语法，才使得论文能够成功发表。

在研究生生活上，我要感谢宋日辉同学一直以来的开导与鼓励，也感谢刘上华师弟陪着一起准备托福，一起健身。研三一边学习一边健身的那段时间是我整个硕士期间最充实的，也是最快乐的日子。我还要感谢以前的本科室友们，在我心生迷茫的时候耐心开导我；特别要感谢郭磊金同学，虽然相隔千里，但仍能定期和我一起探讨各种政治、哲学问题，让我在繁忙的研究生活中稍微透透气。另外，我还要感谢《进击的巨人》作者稷山创，他的作品让我惊叹，痴迷，疯狂，也让我接受了自身的平凡性，然后脚踏实地的前行。

在毕业论文撰写期间，刚好是新型冠状病毒爆发的时候。由衷感谢所有为抗击疫情付出的人们，他们有的比我年长，有的是同龄人，有的比我年轻。他们都是普普通通的人，有着自己的梦想、自己的家庭，但是为了全人类的利益，他们甘愿冒着生命危险奔赴前线，他们是我辈之楷模，是真正中华民族的脊梁。

衷心的感谢我的家人一直以来的付出，一直以来对我的关心、支持和理解。

也感谢我的女朋友彭铭杏一直以来的陪伴和支持，感谢为我一字一句的校对毕业论文。没有你们的辛勤付出和支持，就没有我今天所取得的成果。

最后，感谢所有曾经教育和帮助过我的所有老师，也衷心地感谢为评阅本论文而付出宝贵时间和辛勤劳动的专家和教授们！

郭叙森

二〇二〇年五月二十日