课程论文题目

许晓旭

摘要

随着三维数据采集技术飞快发展，由于三维数据带有独有的空间信息，三维点云目标检测技术也逐渐在某些领域替换掉原先的二维目标检测算法，例如无人车领域等。由于三维点云比二维图像数据具有更为复杂的特性，且三维点云数据规模较为庞大，利用传统三维点云目标检测算法易造成计算成本过高或者识别精度过低等问题，因此三维点云目标检测技术具有极高的研究意义与应用价值。本文基于工业界常用的 PintPillars 网络作为基本框架，提出了一种新的三维目标检测模型，在保证识别速率的基础上提高其目标识别性能。首先结合细粒度化思想提出细粒度化柱化，充分获取目标检测对象的空间特征，其次对于网络骨干网对点云特征提取不充分问题提出 Mini-HRNet block，提高对点云特征获取能力。KITTI 数据集实验对比以及消融实验分析表明，本文提出的算法可以有效提高目标识别性能，能够在识别性能与识别速率达到 trade-off 的效果。

关键词：三维点云目标检测；多尺度特征；空间结构信息

1 引言

随着 3D 采集技术的快速发展，RGB-D 深度相机、激光雷达等 3D 传感器在无人驾驶等领域广泛应用。相比于 2D 视觉数据缺乏空间信息，3D 数据对于真实 3D 世界的表示能力更强，包含了例如物体旋转角和物体深度等信息。这些数据在不同领域中发挥着不同的作用，包括医疗，自动驾驶，无人配送等各方面行业。

随着 3D 目标检测算法的性能逐渐提升，自动驾驶逐渐成为国内外公司聚焦的下一个风口。现阶段，国外自动驾驶领头羊 Waymo 已经将高度自动驾驶（L4 等级）无人汽车投放到市场上；国内百度也已经在全国部分区域进行无人汽车试点；阿里用无人配送车小蛮驴完成“最后一公里配送”。在疫情阶段深圳市大规模采用了无人货车进行物资配送，在避免了人员流动给疫情管理带来的不可控性的情况下保证了正常物资的配送。而这些预示着无人车的发展是未来的一个风口。

尽管现有的 3D 点云目标检测算法在识别精确度上已经基本满足车辆上路的要求，然而在推理速度过慢成为了 3D 点云目标检测算法的掣肘。如 PV-RCNN[1]算法在 KITTI 数据集上 BEV 指标的 mAP 达到了 77%，但其推理速度仅为 13fps。目前，业界主要是基于 PointPillars 方法进行优化，但其 BEV 指标的 mAP 仅为 71%。因此，如何在实时性的要求下（大于 30fps），尽可能提升 3D 点云目标检测模型精度，具有非常广泛的应用价值。

2 相关工作

随着 3D 数据采样技术的成熟以及算力的提高，同时由于三维数据对现实世界有较强的可解释性，3D 目标检测开始受到研究者的关注。传统的三维目标检测算法是基于手工特征的，因此需要花费大量的人力对数据进行单独的标注，而近年来随着深度学习的进一步发展，一些端到端的深度学习算法

也被提出。基于上述的点云的表示方式，现如今流行的 3D 点云目标检测方法有三种，分别为投影方法、点云方法和体素方法。

2.1 投影方法

早期鉴于 2D 图像目标检测方法较为成熟，因此研究者将 3D 点云数据投影成 2D 图像，比如转化为鸟瞰图（Bev）与前视图（Rv）后，直接利用 2D 图像的目标检测方法对转化后的图像进行检测。MV3D[2]通过对为点云设置鸟瞰图和前视图的转化编码方式，将原始三维点云分别转化为鸟瞰图和前视图，由于在转化为两种视图的过程中不可或缺的丢失了原始点云的深度信息，因此采用了原始的 RGBD 图像作为输入信息，以来补充点云缺失的深度信息。

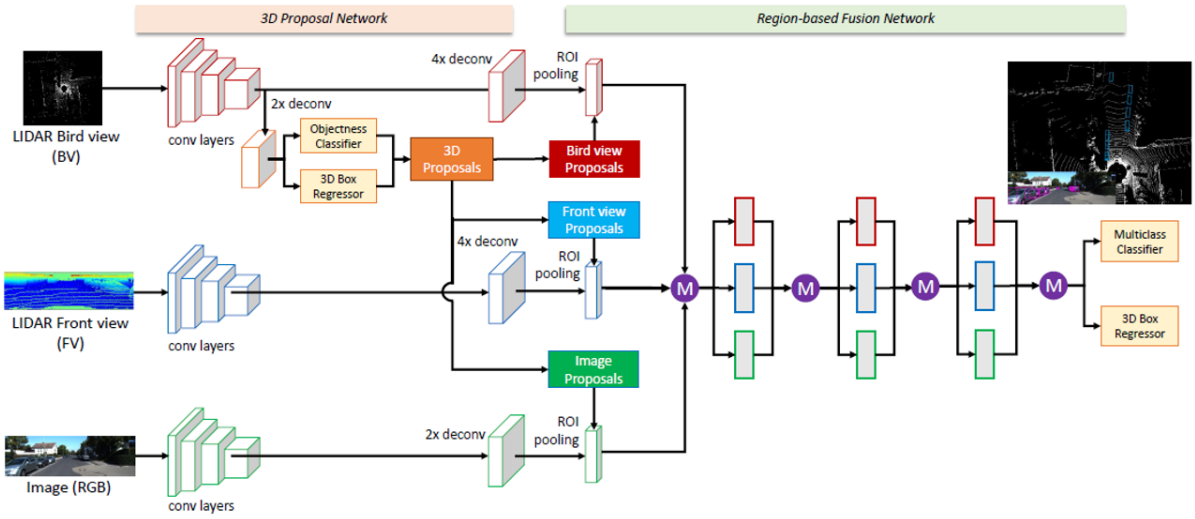

图 1: 多视图识别网络，MV3D

2.2 点方法

点云方法直接对三维点云数据进行利用。然而点云数据的无序性和旋转性便是网络端到端训练的第一大难题，因此研究者提出旋转矩阵 STN[3]将点云旋转到一个最有利于网络训练的角度，解决点云旋转性问题，而后巧妙使用最大值池化解决了点云的无序性问题。

而由于全局信息由最大池化层进行提取，信息提取过程中损失是较为严重，且对于点云点的局部结构信息的获取能力较差，因此 PointRCNN[4]提出了 sampling 和 grouping 网络结构，通过将点云划分为点云圈，便可对点云的局部特征信息进行进一步的信息整合，能更好地对点云地局部特征信息进行提取；并且采用了层次特征学习结构，对于每一层都重新进行点云圈的划分，便于对于信息进行多次融合，能使得局部特征提取更加完整；且网络采用了 encoder-deconder 编码方式以及 skip-connection 结构，能获取不同尺度下的点云特征信息。

由于点云的数据规模较为庞大，假如直接对三维点云数据进行卷积计算，会消耗巨大的计算成本；且对于点云数据而言也如二维图像数据一样，存在着一些噪声点，假如对点云数据进行全部使用的话，会使得噪声对目标检测任务造成严重干扰，造成性能下降。

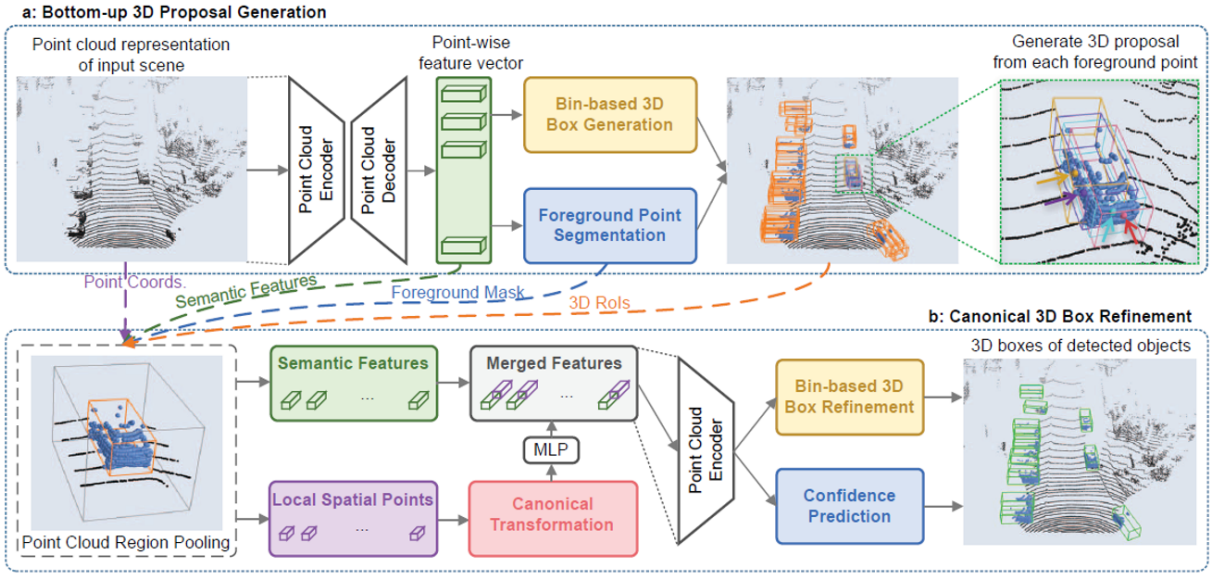

图 2: 点的信息多次聚合与提取，PointRCNN

2.3 体素方法

鉴于原始的点云数据是不规则的，VoxelNet[5]从二维规则像素表示图像中获得灵感，将三维不规则点云转化为规则的体素进行表示，实现了端到端的学习方法。首先将点云进行多个体素的划分，然后对每个体素进行重新编排，得到一个不同于点云的规则的数据格式。因此利用这种规则的数据便可以端到端进行 3D 点云的学习。并且针对点云稀疏性的问题，提出了稀疏张量表示方法，即只记录有点云的体素序号，便于后续对其进行计算操作，降低了反向传播的内存消耗和计算成本。

然而三维卷积的计算成本是巨大的，且目标检测头为两阶段的 RPN 网络，两者给网络带来了巨大的计算成本 (4.4fps)，不能满足实时检测。PointPillars[6]随之提出，对 VoxelNet 的网络框架进行改进，首先对于体素表示，它提出了一种新的点云编码方式，即将点云进行柱化操作，即在空间中绘制固定大小的网格。与 VoxelNet 对点云的编码方式不相同的是，每一个网格对应的只是一个点云柱子，而不是多个堆叠在一起的体素块，而后对每一个点云信息柱进行信息提取，再进行维度转化，便可以成功将点云转化为 2D 的伪图像。因此后续便可以不使用计算成本较高的 3D 卷积进行计算，转而直接使用计算成本较低的 2D 卷积，减少了计算成本。在转化为 2D 伪图像之后便可以直接使用传统 2D 目标检测方法对对象继续进行目标检测识别。

PointPillars 虽性能有所提升，仍存在一定的不足。首先是它的简单柱化操作导致的空间信息损失。由于采样过程是对一根较高的信息柱（KITTI 数据集中 z 轴高度为 4m）进行不均匀的随机采样，故目标物体的空间信息存在某种程度的损失。同时，由于采样点的数目存在一定的阈值，柱化会导致较多点云数据被抛弃，信息丢失较为严重。其次是 2D 伪图像信息提取的骨干网络结构简单，精度有限。为了识别的效率，它的骨干网络部分结构比较简单，虽然采用了一个三层的类似于 top-down 结构对转化的 2D 伪图像进行多尺度图像信息的特征提取，能得到不同尺度的图像特征信息，然而提取的方式过于简单，且对不同层次的特征图信息没有很好地进行信息融合。因此对于 2D 伪图像的特征信息的提取不足。

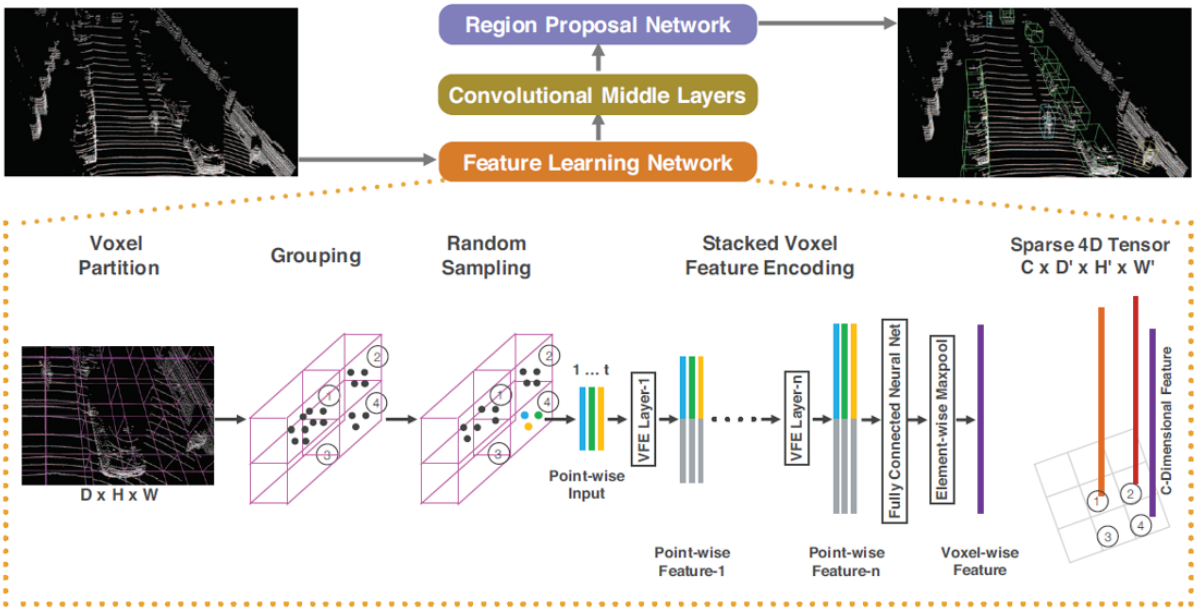

图 3: 体素网格卷积，VoxelNet

3 本文方法

3.1 本文方法概述

本文以 PointPillars 为网络基本骨架，结合细粒度化柱化方法和 Mini-HRNet block，提出一个新的三维点云目标检测网络 PiFHNet。如图 4 所示，本文所提出的模型首先接受点云数据并通过细粒度化柱化方法将不规则点云变成点云信息块并进行合并，其次通过简单的卷积转化成二维的伪图像；接着经过高分辨率网络块 Mini-HRNet block 进行进一步的特征信息提取；最后将提取得到的信息送入 SSD 检测头中进行目标边框定位，从而完成目标检测。

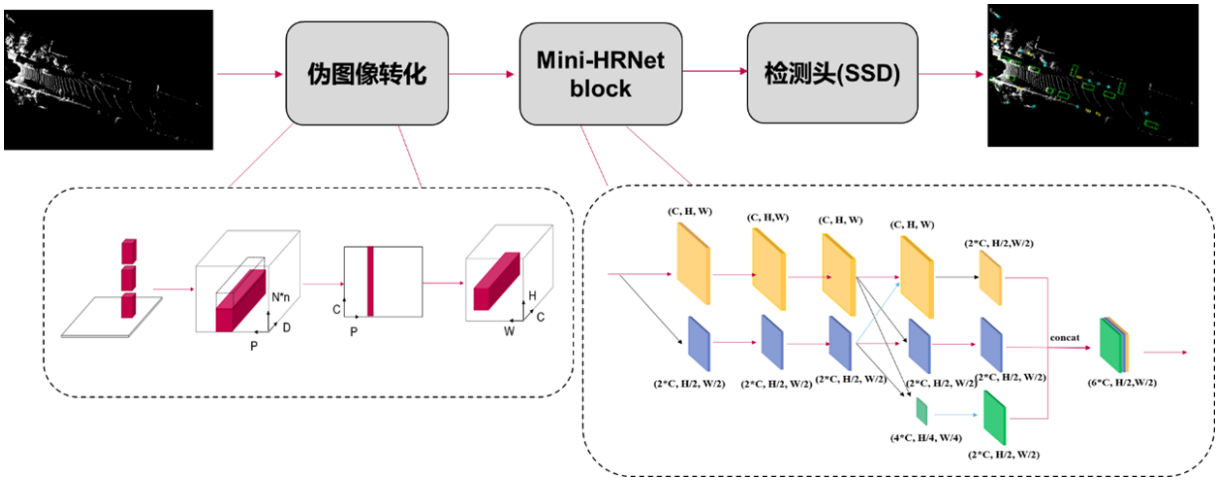

图 4: PiFHNet

3.2 细粒度化柱化

由于三维点云数据是无规则的，而三维卷积会带来巨大的计算成本，因此本文提出一种新的点云编码方式——细粒度化柱化方法，来将点云转化为二维伪图像。

假设现有一个点云 l，三维坐标系分别为 x，y 和 z，点云中的点附带信息为 x_i, y_i, z_i, r，其中前三个分别表示点云的空间坐标信息，第四个表示点的反射率。首先在 $x-y$ 平面上均匀划分为 $H \times W$ 个网格，并将点分配到每一个网格中，而每个网格上便形成了一根点云信息柱。对对象进行层次划分并分层处理

能获取对象各个局部部分更加详细的信息，即对对象进行空间的细粒度化操作有利于充分采集对象的空间特征信息。于是在 z 轴方向对所有之前形成的点云信息柱进行细粒度，每根点云信息柱均匀划分为 n 个等高的点云块，这样便可避免由于随机采样不均匀而带来的空间信息丢失问题。其次计算每一个点云块的空间中心坐标 $x_{center}, y_{center}, z_{center}$ 以及块内所有点的中心坐标 $x_{cluster}, y_{cluster}, z_{cluster}$。因此点云信息块的点便附带着 $D = 10$ 个维度的信息 $x_i, y_i, z_i, r, x_{center}, y_{center}, z_{center}, x_{cluster}, y_{cluster}, z_{cluster}$。

由于点云的稀疏性，划分得到的点云信息柱以及信息柱上的点云信息块大部分都是空的，且非空的信息柱和信息块上的点也是非常少的。因此本文对每个样本的非空柱数和信息块中的点进行阈值设定。我们将一个点云最少拥有的点云信息柱的数量设置为 P，每个点云信息块的点的最小数量设置为 N。

如图 5 所示为点云信息提取示意图，假设 $P = 1, N = 5, n = 3$，因此首先先在 $x - y$ 平面上均匀划分网格，在对网格上的点云信息柱划分成三份等高的点云信息块。其次对点云信息柱进行随机采样，这里是随机选中了 P1 点云信息柱，而假如采样的 P 值大于现有的信息柱的数目，则对新的点云信息柱进行 0 填充。接着对 P1 中的点云信息块中的点进行随机采样，这里设置阈值 $N = 5$，即对于类似超过五个点的（例如 V2），则对其中点随机采样五个作为采样最终信息；而对于点云信息块中不超过五个点的（例如 $V1$ 和 $V3$），则在尾部进行 0 填充。对于同一根点云信息柱上的 n 个点云块，将其提取后的特征信息在 z 轴方向进行叠加，因此一根点云信息柱便形成 $(D, N * n)$ 的规则信息块，对所有的点云信息柱进行聚合，最终形成一个规则的信息块 $(D, P, N * n)$。

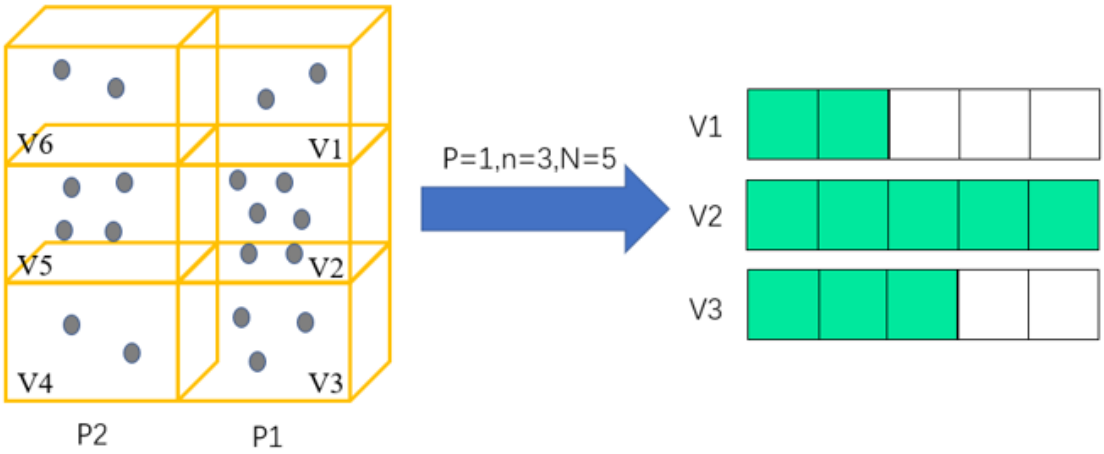

图 5: 细粒度化柱化点云信息提取

接着使用一个简单的卷积操作进行进一步的特征提取，将 $(D, P, N*n)$ 卷积得到 $(C, P, N*n)$，对最后一维进行最大池化操作，即得到 (C, P)。由于 $P = H \times W$，因此可以得到一张二维伪图像 (C, H, W)，其中 H 和 W 分别对应鸟瞰图的 x 轴和 y 轴长度。

经上述点云编码操作之后，即可将 3D 点云转化为 2D 伪图像，后续便可直接使用 2D 卷积对点云特征进行进一步提取。总的点云转化为伪图像流程如图 6 所示。

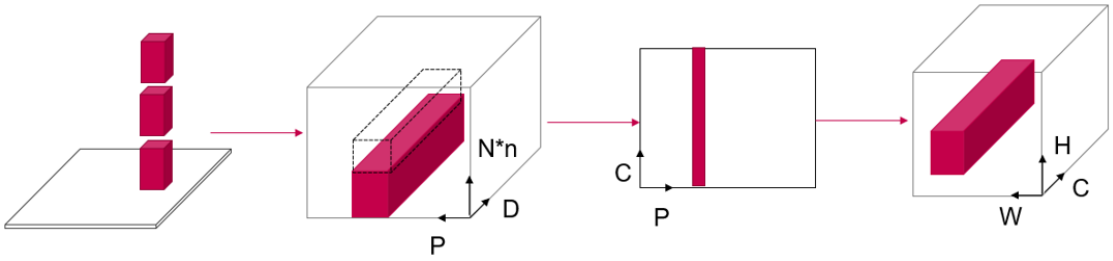

图 6: 点云转化为伪图像

3.3 Mini-HRNet block

在本文模型的骨干部分提出了高分辨率视觉骨干网络架构 Mini-HRNet block，如图 7 所示，为 HRNet[7]的一个小型架构网络，采用了并行和串行的方式来进行子网连接，能接收前面网络处理得到的二维伪图像信息，再对其进行进一步特征提取，最后在送进目标定位网络进行定位。

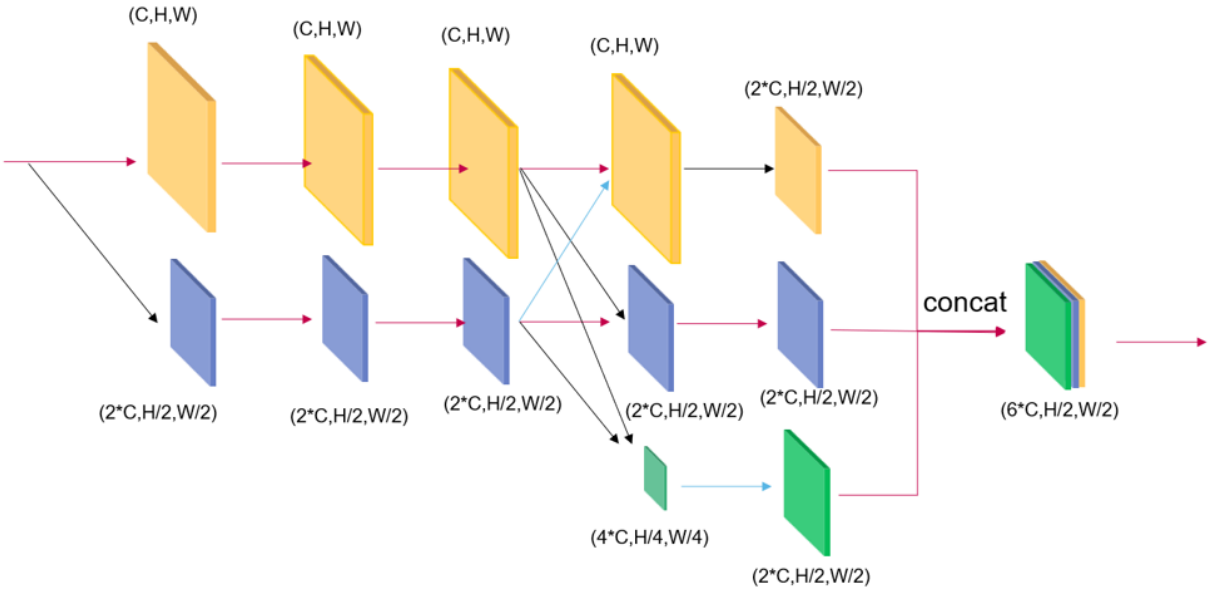

图 7: Mini-HRNet block

Mini-HRNet block 得到 2D 伪图像之后，首先进行一次步长为 2，卷积核大小为 3×3 的下采样操作，将原始伪图像大小降采样为原始大小的一半，并将通道数增加到原先的两倍。并将两个不同尺度的特征图像进行进一步步长为 1，卷积核大小为 3×3 的普通卷积，在不改变特征图像信息的大小的基础上进行进一步特征提取。

其次进行不同尺度的特征信息的融合，本文将对不同尺度特征信息进行融合的网络架构定义为交换单元。前一个阶段与后一个阶段之间便存在一个交换单元，将多层次特征信息进行融合，得到下一个阶段的输入特征数据。具体的交换单元如图 8 所示。

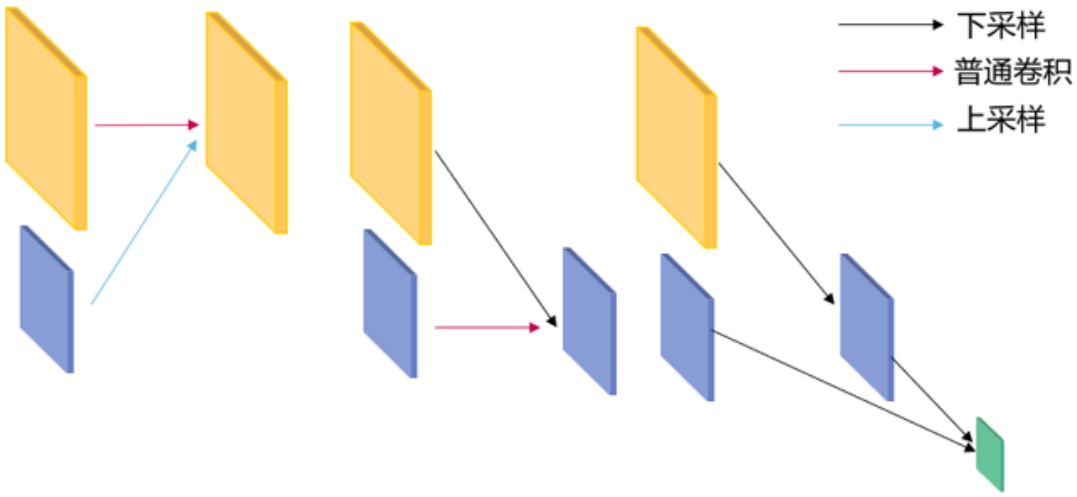

图 8: 交换单元

交换单元的卷积方式可以分为两种情况，一种是高分辨率特征图通过下采样方式得到低分辨率特征图。假设当前图像分辨率与输出图像分辨率的大小一致，则不进行任何处理直接对特征图像数据进行累加；假设当前图像分辨率比输出图像分辨率的大 2 倍，则通过使用 3×3 的卷积核并且步长为 2 进行下采样操作；假如当前图像分辨率比输出图像分辨率大 4 倍甚至更多，则继续使用 3×3 卷积步长为 2 进行下采样至输出图像分辨率大小为止。

另外一种是低分辨率特征图像通过上采样得到高分辨率图像。假设当前图像分辨率与输出图像分辨率的大小一致，则不进行任何处理直接对特征图像数据进行累加；假设当前图像分辨率是输出图像分辨率的 1/2，则通过使用 1x1 卷积核进行双线性插值上采样至输出图像分辨率大小。对于交换单元处理过程中通道数数量大小保持不变。

将经过交换单元融合后的三个不同尺度的图像特征分别使用 3x3 卷积步长为 2 进行下采样，3x3 卷积步长为 1 进行普通卷积和 1x1 卷积核进行双线性插值上采样，并统一 3 个输出图像特征通道为 2C。最终将三者进行合并，得到包含着不同尺度的图像特征信息。

3.4 损失函数定义

首先对检测框进行损失函数定义。假设检测框定义为 $(x, y, z, l, w, h,)$，其中 x, y, z 为检测框的框中心坐标，l, w, h 分别表示检测框的长宽高，表示检测框的旋转角。因此真实检测框 $(x^{gt}, y^{gt}, z^{gt}, l^{gt}, w^{gt}, h^{gt}, ^{gt})$ 与预测检测框 $(x^a, y^a, z^a, l^a, w^a, h^a, ^a)$ 在各个维度对应的残差可以计算表示为：

$$\begin{gathered} \triangle x = \frac{(x^{gt} - x^a)}{d^a}, \triangle y = \frac{(y^{gt} - y^a)}{d^a}, \triangle z = \frac{(z^{gt} - z^a)}{h^a} \\ \triangle w = \log\left(\frac{w^{gt}}{w^a}\right), \triangle l = \log\left(\frac{l^{gt}}{l^a}\right), \triangle h = \log\left(\frac{h^{gt}}{h^a}\right) \\ \triangle\theta = sin(\theta^{gt} - \theta^a), d^a = \sqrt{(w^a)^2 + (l^a)^2} \end{gathered} \tag{1}$$

因此检测框回归损失函数采用 $SmoothL_1$ 作为网络损失函数，可以计算为：

$$SmoothL_1 = \begin{cases} 0.5x^2 \ \ if \ |x| < 1 \\ |x| - 0.5 \ \ otherwise \end{cases} \tag{2}$$

$$L_{loc} = \sum_{b\in(x,y,z,w,l,h,\theta)} SmoothL_1(\triangle b) \tag{3}$$

接着对类别分类函数进行定义。而由于类别分布不均匀问题，假如直接使用普通的交叉损失函数（CrossEntropy Loss）会导致由于类别不均匀导致某一类的损失值偏大，不利于对于小类别的目标进行分类。因此采用焦点损失函数（Focal Loss[8]）来解决类别不均匀问题。

$$L_{cls} = -\alpha_a(1-p^a)^\gamma \log(p^a) \tag{4}$$

其中 p^a 表示该类别的框出现的概率，^{a} 和 为常数参数。由于前面对于检测框信息的回归损失无法判断目标物体的朝向，因此最后添加 softmax 来进行角度方向判断，对应的损失函数为二则交叉熵损失函数，标记为 L_{dir}。

$$L_dir = -[p\log(p) + (1-p)\log(1-p)] \tag{5}$$

因此网络总的损失函数可以定义为：

$$L = \frac{1}{N_{pos}}(\beta_{loc}L_{loc} + \beta_{cls}L_{cls} + \beta_{dir}L_{dir}) \tag{6}$$

其中 N_{pos} 为正样本框的数量，$\beta_{loc}, \beta_{cls}, \beta_{dir}$ 别为三个常数参数。

4 复现细节

4.1 与已有开源代码对比

本工作参考了 OpenPCDet 开源代码库，里面包含 PointPillars 等较为常用的 3D 目标检测算法，在复现过程中参考了其中的数据的生成，并在原始的基础上对 PointPillars 的柱化结构和后置的编码结构进行重构。

4.2 实验环境搭建

本工作代码实验环境主要依托 pytorch、spconv 等源代码库进行搭建，具体依赖库可见附件。

4.3 创新点

- 本文提出了细粒度化柱化方法来对 3D 点云进行信息初始化提取，即在柱化过程中结合细粒度化思想，提升对目标物体空间信息提取能力。
- 本文提出一个多尺度图像信息结合的网络块 Mini-HRNet block，能有效地进行多层次特征融合。

4.4 实验数据介绍

本文实验是基于经典 3D 目标检测数据集 KITTI[27] 进行的。KITTI 数据集包含 7481 个训练点云数据和 7518 个测试点云数据，一共包含 80256 个标记物体。KITTI 数据集的点云的 x,y,z 的范围为：[(0,69.12),(-39.68,39.68),(-3,1)]。本文将训练数据集进行分割，其中 3712 个训练数据作为模型的训练集，3769 个训练数据作为模型的验证集。而对于 KITTI 数据集中的目标识别难度，官方将其分成三个等级：

- Easy：最小的检测框高度为 40 像素，不会出现遮挡情况，最大检测框重叠比例为 15 %；
- Moderate：最小检测框高度为 25 像素，最多出现部分遮挡现象，最大检测框重叠比例为 30%；
- Hard：最小检测框高度为 25 像素，最多出现完全遮挡现象，最大检测框重叠比例为 50%。

4.5 参数设置

4.5.1 基本参数设置

本文设置每个网络单元格 xy 的长度为 0.16m，z 轴上每个细粒度柱子的长度为 0.8m，每一个点云的最大柱数为 12000，每个细粒度体素块的最大点为 100 个。

4.5.2 类别基本参数设置

KITTI 目标检测数据集一共包含三个类别，分别为：Car（汽车）、Pederstrain（行人）和 Cyclist（自行车手）。对于这三种不同类别的目标对象，本文对其基本参数设置如下：

(1) Car：汽车的检测框的长宽高分别为 3.9，1.6 和 1.56。并且在得到预测检测框后，当预测框与真实检测框的 IOU 值小于 0.45 时，则将该框分类为负样本；否则当预测框与真实检测框的 IOU 值大于 0.6 时，则将该框分类为正样本；当 IOU 值介于 0.45 到 0.6 之间时，则不对该框进行分类。

(2) Pedestrian：行人的检测框的长宽高分别为 0.8，0.6 和 1.73。并且在得到预测检测框后，当预测框与真实检测框的 IOU 值小于 0.35 时，则将该框分类为负样本；否则当预测框与真实检测框的 IOU 值大于 0.5 时，则将该框分类为正样本；当 IOU 值介于 0.35 到 0.5 之间时，则不对该框进行分类。

(3) Cyclists：自行车手的检测框的长宽高分别为 1.76，0.6 和 1.73。并且在得到预测检测框后，当预测框与真实检测框的 IOU 值小于 0.35 时，则将该框分类为负样本；否则当预测框与真实检测框的 IOU 值大于 0.5 时，则将该框分类为正样本；当 IOU 值介于 0.35 到 0.5 之间时，则不对该框进行分类。

5 实验结果分析

本文依照 KITTI 数据集的测试指标，对不同类别的 IOU 进行定义。对于 Car 类别，则当 IOU>0.7 时，则定义为检测正确，当 IOU 低于 0.7 时则检测错误；同理对于 Pedestrian 和 Cyclist 类别，当 IOU>0.5 时，则定义为检测正确，当 IOU 低于 0.5 时则判定为检测错误。

5.1 算法性能比较

表 1: 各个算法 BEV 指标性能比较

模型	FPS	Car			Pedestrain			Cyclist		
		Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
PointPillars	**63**	89.76	**87.37**	**85.38**	59.09	54.71	50.80	83.18	66.98	62.98
PiFHNet	48	**89.85**	87.21	85.12	**62.88**	**56.40**	**52.78**	**90.31**	**72.68**	**69.34**

表 2: 各个算法 3D 指标性能比较

模型	FPS	Car			Pedestrain			Cyclist		
		Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
PointPillars	**63**	87.22	77.14	75.62	54.66	49.45	46.19	80.43	63.20	59.40
PiFHNet	48	**88.04**	**77.61**	75.94	**58.01**	**51.99**	**48.53**	**89.55**	**70.48**	**65.68**

如表 1 和表 2 所示为各个模型在 KITTI 目标检测数据集上的表现。首先进行横向分析，对于原始 PointPillars 算法，PiFHNet 尽管在 Car 类的性能与原始算法性能相似，然而在 Pedestrian 和 Cyclist 类表现却比原始算法性能更加佳。对于 BEV 指标，PiFHNet 在 Pedestrian 的 AP 指标上提升了大约 3 个百分点，而在 Cyclists 上的 AP 指标上提升了大约 6 个百分点，虽然本文改良算法在检测效率（48fps）上低于原始 PointPillars 检测性能（63fps），而在这性能提升的基础上是可接受的。

5.2 消融实验

如表 3 和表 4 为对本文提出的算法架构进行的消融实验对比。对比 PointPillars 和细粒度化柱化算法，可以看出虽然 Car 类的提升不是很明显，然而 Pedestrians 和 Cyclist 类别提升是较大的，最大提升可达 4%（Cyclist 的 hard 指标），而这正是得益于改良的细粒度柱化效果。由于原始 PointPillars 算法直接在 z 轴进行柱化操作，很大程度上对目标的空间信息进行了破坏，造成网络无法很好地接收到目标检测物体的空间信息。因此对于 Pedestrian 和 Cyclist 带有极强的空间信息特征的目标对象，无法很好地进行空间信息的提取是无疑是丢失了目标检测物体的一个显著的特征。引入的细粒度化方法在 z 轴上进行多粒度划分，将 z 轴空间分为多个区域，每个区域再进行局部的柱化。因此相比于原先算法可以从 z 轴方向上获取关于目标物体的空间架构信息，使得目标检测更加准确。

对于 PointPillars 和加入 Mini-HRNet block 骨干网络算法，从表中可以看到虽然在 Car 类和 Pedestrian 类上的性能与 PointPillars 的性能差距不大，而在 Cyclist 类别上的性能上却能高于原始 baseline 算法大约 4%。说明对转换的 2D 伪图像进行进一步的特征提取对于目标检测任务来说还是有益的。对于整体识别性能速率而言，虽然对比于 PointPillars 的 63fps 是稍稍逊色的，但 52fps 和 58fps 的识别速率换取识别性能的较大增长还是可以接受的。

表 3: 各个算法 BEV 指标性能比较

模型	FPS	Car			Pedestrain			Cyclist		
		Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
PointPillars	**63**	89.76	87.37	85.38	59.09	54.71	50.80	83.18	66.98	62.98
细粒度化	52	**90.09**	**88.09**	**86.51**	62.60	**56.67**	53.13	85.77	69.02	65.91
Mini-HRNet	58	89.67	87.11	84.51	59.49	54.24	51.14	88.29	69.24	64.62
PiFHNet	48	89.85	87.21	85.12	**62.88**	56.40	**52.78**	**90.31**	**72.68**	**69.34**

表 4: 各个算法 3D 指标性能比较

模型	FPS	Car			Pedestrain			Cyclist		
		Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
PointPillars	**63**	87.22	77.14	75.62	54.66	49.45	46.19	80.43	63.20	59.40
细粒度化	52	87.55	**78.53**	**77.07**	56.43	51.79	48.00	83.95	65.95	63.71
Mini-HRNet	58	87.12	76.71	74.15	53.62	48.75	45.33	84.77	65.54	61.86
PiFHNet	48	**88.04**	77.61	75.94	**58.01**	**51.99**	**48.53**	**89.55**	**70.48**	**65.68**

6 总结与展望

本文基于工业界常用目标检测网络框架 PointPillars，对柱化进行改进，提出细粒度化方法获取目标检测对象空间特征；再者针对骨干网络过于简单，对于点云特征信息提取不充分问题，本文提出 Mini-HRNet block，将其穿插进原先骨干网络中，提高了网络对于点云特征的提取。通过 KITTI 数据集的性能分析表明，尽管在识别速度有所下降，然而在识别性能上出现了不同程度的性能上浮；而对

比于现阶段流行的三维目标检测算法而言，在某些检测对象性能是较低的，但在部分类上的检测性能是较为不俗的，识别速度更是快了三倍。

对于本文面向无人车的三维目标识别算法，还存在以下可以改进与拓展的空间：首先是多视角融合技术，从柱化的算法思想不难看出，其实是采用了鸟瞰图的思想，将三维点云进行投影处理化，只是处理方式相对于以往的方式较优，因此可以同样采用前视图的思想，将三维点云进行前视图投影化并与柱化得到的特征信息进行融合，应能进一步提升本算法对目标特征的信息提取；其次是对点云柱化的方式，本文对于点云柱化采用的是传统的体素化方式，由于对于每个体素的点云的数量和体素块的数量存在阈值，容易丢失带有特征信息的点云数据，因此可以根据 lidar 扫描数据特性，进行扇形的体素空间划分 [28]，并且对体素内容进行动态体素化方法 [29]，即有则处理，无则丢弃，对点云数据完全进行利用，不进行填充 0 占用计算空间；最后是添加 FPN 网络架构 [30]，利用特征金字塔的 top-down 结构更充分利用三维数据的多尺度信息，提升识别精确率。

参考文献

[1] SHI S, GUO C, JIANG L, et al. PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection [C]//2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2020.

[2] CHEN X, MA H, WAN J, et al. Multi-View 3D Object Detection Network for Autonomous Driving[C]//2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017.

[3] QI C R, WEI L, WU C, et al. Frustum PointNets for 3D Object Detection from RGB-D Data[Z]. 2017.

[4] SHI S, WANG X, LI H. PointRCNN: 3D Object Proposal Generation and Detection From Point Cloud [C]//2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2019.

[5] ZHOU Y, TUZEL O. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection[C]//2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2018.

[6] LANG A H, VORA S, CAESAR H, et al. PointPillars: Fast Encoders for Object Detection From Point Clouds[C]//2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2019.

[7] WANG J, SUN K, CHENG T, et al. Deep High-Resolution Representation Learning for Visual Recognition[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2020, PP(99): 1-1.

[8] LIN T Y, GOYAL P, GIRSHICK R, et al. Focal Loss for Dense Object Detection[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2017, PP(99): 2999-3007.