VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection

VoxelNet: 端到端学习的基于点云的 3D 目标检测

论文引用：

Zhou Y, Tuzel O. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR). 2018: 4490-4499.

ABSTRACT

Accurate detection of objects in 3D point clouds is a central problem in many applications, such as autonomous navigation, housekeeping robots, and augmented/virtual reality. To interface a highly sparse LiDAR point cloud with a region proposal network (RPN), most existing efforts have focused on hand-crafted feature representations, for example, a bird's eye view projection. In this work, we remove the need of manual feature engineering for 3D point clouds and propose VoxelNet, a generic 3D detection network that unifies feature extraction and bounding box prediction into a single stage, end-to-end trainable deep network. Specifically, VoxelNet divides a point cloud into equally spaced 3D voxels and transforms a group of points within each voxel into a unified feature representation through the newly introduced voxel feature encoding (VFE) layer. In this way, the point cloud is encoded as a descriptive volumetric representation, which is then connected to a RPN to generate detections. Experiments on the KITTI car detection benchmark show that VoxelNet outperforms the state-of-the-art LiDAR based 3D detection methods by a large margin. Furthermore, our network learns an effective discriminative representation of objects with various geometries, leading to encouraging results in 3D detection of pedestrians and cyclists, based on only LiDAR.

摘要

准确检测 3D 点云中的目标是许多应用中的核心问题，例如自主导航，管家机器人和增强/虚拟现实。为了将高度稀疏的 LiDAR 点云与区域提议网络（RPN）连接起来，大多数现有的努力都集中在手工制作的特征表示上，例如鸟瞰视图投影。在这项工作中，我们不再需要对 3D 点云进行手动特征工程，并提出了 VoxelNet，这是一种通用的 3D 检测网络，可将特征提取和边界框预测统一到单个阶段的端到端可训练深度网络中。具体而言，VoxelNet 将点云划分为等间距的 3D 体素，并通过新引入的体素特征编码（VFE）层将每个体素内的一组点转换为统一的特征表示。通过这种方式，点云被编码为描述性体积表示，然后将其连接到 RPN 以生成检测。KITTI 汽车检测基准测试的实验表明，VoxelNet 在很大程度上优于最先进的基于 LiDAR 的 3D 检测方法。此外，我们的网络学习了具有各种几何形状的物体的有效辨别表示，从而在仅基于 LiDAR 的行人和骑车者的 3D 检测中产生令人鼓舞的结果。

1. Introduction

Point cloud based 3D object detection is an important component of a variety of real-world applications, such as autonomous navigation [11], [14], housekeeping robots [28], and augmented/virtual reality [29]. Compared to image-based detection, LiDAR provides reliable depth information that can be used to accurately localize objects and characterize their shapes [21], [5]. However, unlike images, LiDAR point clouds are sparse and have highly variable point density, due to factors such as non-uniform sampling of the 3D space, effective range of the sensors, occlusion, and the relative pose. To handle these challenges, many approaches manually crafted feature represen- tations for point clouds that are tuned for 3D object detection. Several methods project point clouds into a perspective view and apply image-based feature extraction techniques [30],

1. 引文

基于点云的三维物体检测是各种现实世界应用的重要组成部分，如自主导航[11]，[14]，家政机器人[28]和增强/虚拟现实[29]。与基于图像的检测相比，LiDAR 提供了可靠的深度信息，可用于精确定位对象并表征其形状[21]，[5]。然而，与图像不同，由于诸如 3D 空间的非均匀采样，传感器的有效范围，遮挡和相对姿势等因素，LiDAR 点云是稀疏的并且具有高度可变的点密度。为了应对这些挑战，许多方法手动制作了针对 3D 对象检测进行调整的点云的特征表示。有几种方法将点云投影到透视图中并应用基于图像的特征提取技术[30]，[15]，[22]。其他方法将点云光栅化为 3D 体素网格，并使用手工制作的特征对每个体素进行编码[43]，[9]，[39]，[40]，[21]，[5]。然而，这些手动设计选择引入了信息瓶颈，阻止这些方法有效地利用 3D 形状信息和检测任务所需的不变性。图像识别[20]和检测[13]任务的重大突破是由

[15], [22]. Other approaches rasterize point clouds into a 3D voxel grid and encode each voxel with handcrafted features [43], [9], [39], [40], [21], [5]. However, these manual design choices introduce an information bottleneck that prevents these approaches from effectively exploiting 3D shape information and the required invariances for the detection task. A major breakthrough in recognition [20] and detection [13] tasks on images was due to moving from hand-crafted features to machine-learned features.

于从手工制作的特征转变为机器学习特征。

Recently, Qi et al. [31] proposed PointNet, an end-to-end deep neural network that learns point-wise features directly from point clouds. This approach demonstrated impressive results on 3D object recognition, 3D object part segmentation, and point-wise semantic segmentation tasks. In [32], an improved version of PointNet was introduced which enabled the network to learn local structures at different scales. To achieve satisfactory results, these two approaches trained feature transformer networks on all input points (~1k points). Since typical point clouds obtained using LiDARs contain ~100k points, training the architectures as in [31], [32] results in high computational and memory requirements. Scaling up 3D feature learning networks to orders of magnitude more points and to 3D detection tasks are the main challenges that we address in this paper.

最近，Qi 等人[31]提出了 PointNet，一种端到端深度神经网络，可直接从点云中学习逐点特征。该方法在 3D 对象识别，3D 对象部分分割和逐点语义分割任务方面表现出令人印象深刻的结果。在[32]中，引入了改进版的 PointNet，使网络能够学习不同规模的局部结构。为了获得满意的结果，这两种方法在所有输入点（~1k 点）上训练了特征变压器网络。由于使用 LiDAR 获得的典型点云包含~100k 点，因此在[31]，[32]中训练架构会导致高计算和内存要求。将 3D 特征学习网络扩展到数量级更多的点和 3D 检测任务是我们在本文中提出的主要挑战。

Region proposal network (RPN) [34] is a highly optimized algorithm for efficient object detection [17], [5], [33], [24]. However, this approach requires data to be dense and organized in a tensor structure (e.g. image, video) which is not the case for typical LiDAR point clouds. In this paper, we close the gap between point set feature learning and RPN for 3D detection task.

区域提议网络（RPN）[34]是一种高度优化的有效物体检测算法[17]，[5]，[33]，[24]。然而，这种方法要求数据密集并且以张量结构（例如图像，视频）组织，这不是典型的 LiDAR 点云的情况。在本文中，我们缩小了点集特征学习与用于 3D 检测任务的 RPN 之间的差距。

We present VoxelNet, a generic 3D detection framework that simultaneously learns a discriminative feature representation from point clouds and predicts accurate 3D bounding boxes, in an end-to-end fashion, as shown in Figure 2. We design a novel voxel feature encoding (VFE) layer, which enables inter-point interaction within a voxel, by combining point-wise features with a locally aggregated feature.

我们提出了 VoxelNet，这是一种通用的 3D 检测框架，可以同时从点云中学习判别特征表示，并以端到端的方式预测精确的 3D 边界框，如图 2 所示。我们设计了一种新颖的体素特征编码（VFE））层，通过将逐点特征与局部聚合特征相结合，实现体素内的点间交互。

Stacking multiple VFE layers allows learning complex features for characterizing local 3D shape information. Specifically, VoxelNet divides the point cloud into equally spaced 3D voxels, encodes each voxel via stacked VFE layers, and then 3D convolution further aggregates local

堆叠多个 VFE 层允许学习用于表征局部 3D 形状信息的复杂特征。具体而言，VoxelNet 将点云划分为等间距的 3D 体素，通过堆叠的 VFE 层对每个体素进行编码，然后 3D 卷积进一步聚合局部体素特征，将点云转换为高维体积表示。最后，RPN 使用体积表示并产生检测结果。

voxel features, transforming the point cloud into a high-dimensional volumetric representation. Finally, a RPN consumes the volumetric representation and yields the detection result. This efficient algorithm benefits both from the sparse point structure and efficient parallel processing on the voxel grid.

这种有效的算法有益于稀疏点结构和体素网格上的有效并行处理。

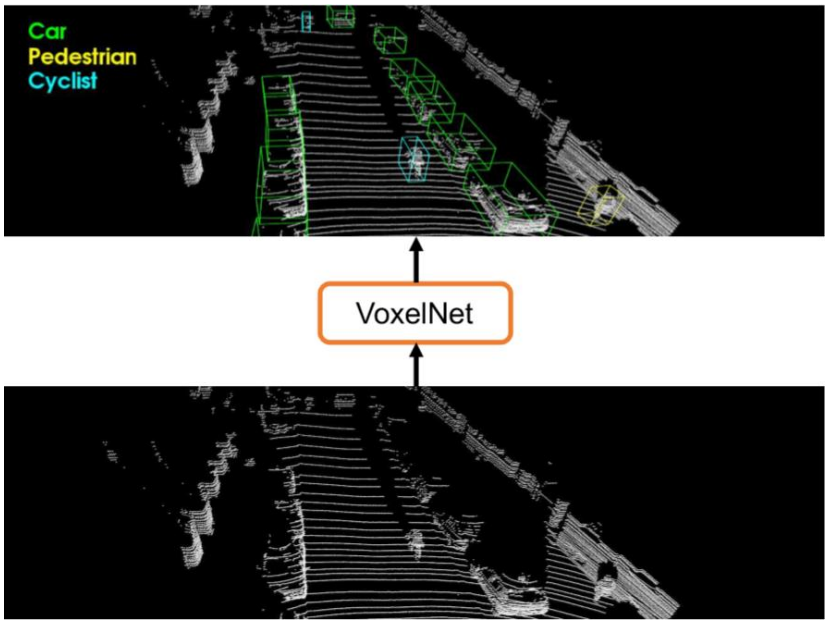

Figure 1. Voxelnet directly operates on the raw point cloud (no need for feature engineering) and produces the 3D detection results using a single end-to-end trainable network.

图1. Voxelnet 直接在原始点云上运行（无需特征工程），并使用单个端到端可训练网络生成 3D 检测结果。

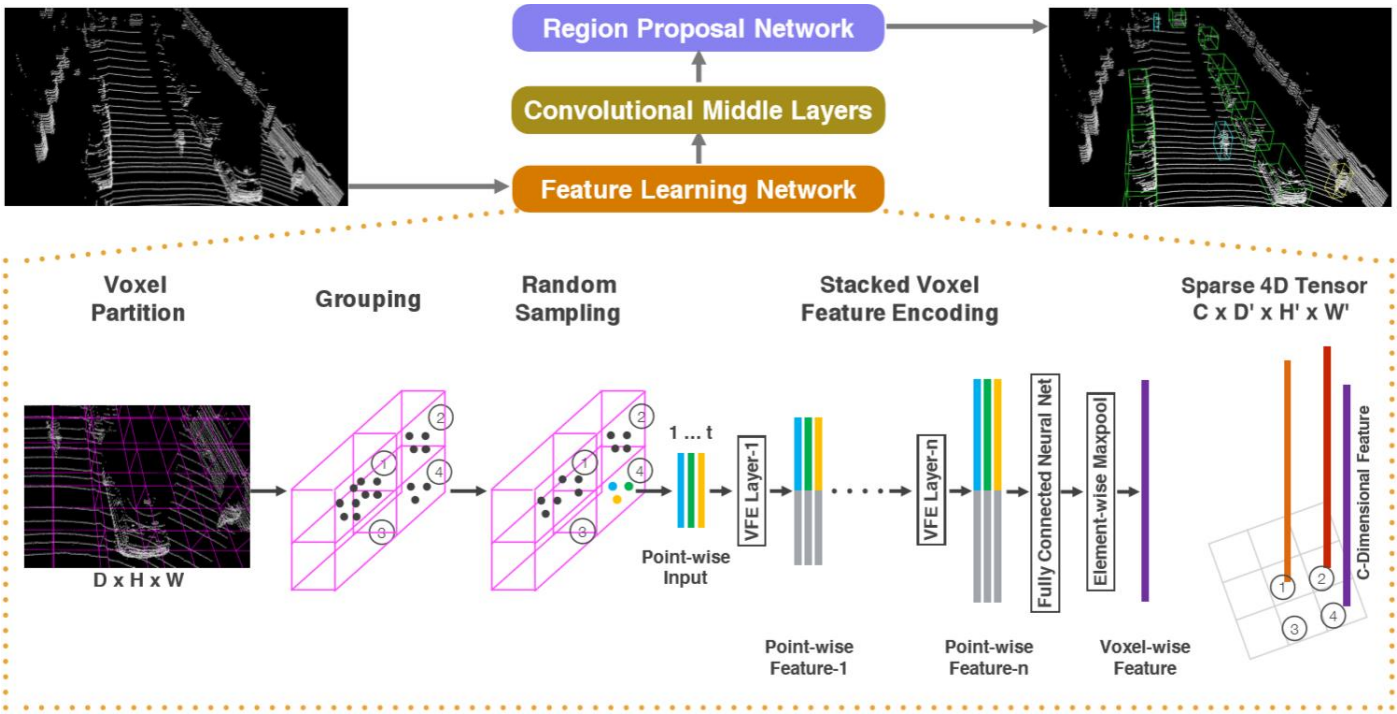

Figure 2. Voxelnet architecture. The feature learning network takes a raw point cloud as input, partitions the space into voxels, and transforms points within each voxel to a vector representation characterizing the shape information. The space is represented as a sparse 4D tensor. The convolutional middle layers processes the 4D

图2. Voxelnet 架构。特征学习网络将原始点云作为输入，将空间划分为体素，并将每个体素内的点变换为表征形状信息的矢量表示。该空间表示为稀疏的 4D 张量。卷积中间层处理 4D 张量以聚合空间上下文。最后，RPN 生成 3D 检测。

tensor to aggregate spatial context. Finally, a RPN generates the 3D detection.

We evaluate VoxelNet on the bird's eye view detection and the full 3D detection tasks, provided by the KITTI benchmark [11]. Experimental results show that VoxelNet outperforms the state-of-the-art LiDAR based 3D detection methods by a large margin. We also demonstrate that Voxel-Net achieves highly encouraging results in detecting pedestrians and cyclists from LiDAR point cloud.

我们根据 KITTI 基准[11]提供的鸟瞰图检测和全 3D 检测任务评估 VoxelNet。实验结果表明，VoxelNet 在很大程度上优于最先进的基于 LiDAR 的 3D 检测方法。我们还证明了 Voxel-Net 在检测 LiDAR 点云的行人和骑车者方面取得了非常令人鼓舞的成果。

1.1 Related Work

Rapid development of 3D sensor technology has motivated researchers to develop efficient representations to detect and localize objects in point clouds. Some of the earlier methods for feature representation are [41], [8], [7], [19], [42], [35], [6], [27], [1], [36], [2], [25], [26]. These hand-crafted features yield satisfactory results when rich and detailed 3D shape information is available. However their inability to adapt to more complex shapes and scenes, and learn required invariances from data resulted in limited success for uncontrolled scenarios such as autonomous navigation.

1.1 相关工作

3D 传感器技术的快速发展促使研究人员开发出有效的表示来检测和定位点云中的物体。一些早期的特征表示方法是[41]，[8]，[7]，[19]，[42]，[35]，[6]，[27]，[1]，[36]，[2]，[25]，[26]。当提供丰富细致的 3D 形状信息时，这些手工制作的特征可以产生令人满意的效果。然而，它们无法适应更复杂的形状和场景，并且从数据中学习所需的不变性导致了诸如自主导航之类的不受控制的场景的有限成功。

Given that images provide detailed texture information, many algorithms infered the 3D bounding boxes from 2D images [4], [3], [44], [45], [46], [38]. However, the accuracy of image-based 3D detection approaches are bounded by the accuracy of the depth estimation.

鉴于图像提供了详细的纹理信息，许多算法从 2D 图像中提取 3D 边界框[4]，[3]，[44]，[45]，[46]，[38]。然而，基于图像的 3D 检测方法的准确性受到深度估计的准确性的限制。

Several LIDAR based 3D object detection techniques utilize a voxel grid representation. [43], [9] encode each nonempty voxel with 6 statistical quantities that are derived from all the points contained within the voxel. [39] fuses multiple local statistics to represent each voxel. [40] computes the truncated signed distance on the voxel grid. [21] uses binary encoding for the 3D voxel grid. [5] introduces a multi-view representation for a LiDAR point cloud by computing a multi-channel feature map in the bird's eye view and the cylindral coordinates in the frontal view. Several other studies project point clouds onto a perspective view and then use image-based feature encoding schemes [30], [15], [22].

几种基于 LIDAR 的 3D 对象检测技术利用体素网格表示。[43]，[9]编码每个非空体素，其中 6 个统计量来源于体素内包含的所有点。[39]融合多个本地统计数据来表示每个体素。[40]计算体素网格上的截断符号距离。[21]对 3D 体素网格使用二进制编码。[5]通过计算鸟瞰图中的多通道特征图和正面视图中的圆柱坐标，介绍了 LiDAR 点云的多视图表示。其他几项研究将点云投射到透视图上，然后使用基于图像的特征编码方案[30]，[15]，[22]。

There are also several multi-modal fusion methods that combine images and LiDAR to improve detection accuracy [10], [16], [5]. These methods provide improved performance compared to LiDAR-only 3D detection, particularly for small objects (pedestrians, cyclists) or when the objects are far, since cameras provide an order of magnitude more measurements than LiDAR. However the need for an additional camera that is time synchronized and

还有几种多模态融合方法将图像和 LiDAR 结合起来以提高检测精度[10]，[16]，[5]。与仅使用 LiDAR 的 3D 检测相比，这些方法提供了改进的性能，特别是对于小物体（行人，骑自行车者）或当物体很远时，因为相机提供比 LiDAR 多一个数量级的测量。然而，需要使用 LiDAR 进行时间同步和校准的附加摄像机限制了它们的使用，并使解决方案对传感器故障模式更加敏感。在这项工作中，我们专注于仅限 LiDAR 的检测。

calibrated with the LiDAR restricts their use and makes the solution more sensitive to sensor failure modes. In this work we focus on LiDAR-only detection.

1.2 Contributions

- We propose a novel end-to-end trainable deep architecture for point-cloud-based 3D detection, VoxelNet, that directly operates on sparse 3D points and avoids information bottlenecks introduced by manual feature engineering.
- We present an efficient method to implement VoxelNet which benefits both from the sparse point structure and efficient parallel processing on the voxel grid.
- We conduct experiments on KITTI benchmark and show that VoxelNet produces state-of-the-art results in LiDAR -based car, pedestrian, and cyclist detection benchmarks.

1.2 贡献

- 我们提出了一种新颖的端到端可训练深度架构，用于基于点云的 3D 检测 VoxelNet，可直接在稀疏 3D 点上运行，并避免手动特征工程引入的信息瓶颈。
- 我们提出了一种实现 VoxelNet 的有效方法，它有利于稀疏点结构和体素网格上的高效并行处理。
- 我们对 KITTI 基准进行了实验，并表明 VoxelNet 在基于 LiDAR 的汽车，行人和骑车人检测基准测试中产生了最先进的结果。

2. VoxelNet

In this section we explain the architecture of VoxelNet, the loss function used for training, and an efficient algorithm to implement the network.

2. VoxelNet

在本节中，我们将介绍 VoxelNet 的体系结构，用于训练的损失函数以及实现网络的有效算法。

2.1 Voxelnet Architecture

The proposed VoxelNet consists of three functional blocks: (1) Feature learning network, (2) Convolutional middle layers, and (3) Region proposal network [34], as illustrated in Figure 2. We provide a detailed introduction of VoxelNet in the following sections.

2.1 Voxelnet 网络框架

提出的 VoxelNet 由三个功能块组成：（1）特征学习网络，（2）卷积中间层，以及（3）区域提议网络[34]，如图 2 所示。我们在下面详细介绍 VoxelNet 部分。

2.1.1 Feature Learning Network

Voxel Partition: Given a point cloud, we subdivide the 3D space into equally spaced voxels as shown in Figure 2. Suppose the point cloud encompasses 3D space with range D,H,W along the Z, Y, X axes respectively. We define each voxel of size v_D,v_H, and v_W accordingly. The resulting 3D voxel grid is of size $D'=D/v_D$, $H'=H/v_H$, $W'=W/v_W$. Here, for simplicity, we assume D,H,W are a multiple of vD,vH,vW.

2.1.1 特征学习网络

体素分区：给定点云，我们将 3D 空间细分为等间距的体素，如图 2 所示。假设点云包含分别沿 Z，Y，X 轴的范围 D,H,W 的 3D 空间。我们相应地定义大小为 vD，vH 和 vW 的每个体素。得到的 3D 体素网格的大小为 $D'=D/v_D$, $H'=H/v_H$, $W'=W/v_W$。这里，为简单起见，我们假设 D,H,W 是 v_D,v_H,V_W 的倍数。

Grouping： We group the points according to the voxel they reside in. Due to factors such as distance, occlusion, object's relative pose, and non-uniform sampling, the LiDAR point cloud is sparse and has highly variable point density throughout the space. Therefore, after grouping, a voxel will contain a variable number of points. An illustration is shown in Figure 2, where Voxel-1 has significantly more points than Voxel-2 and Voxel-4, while Voxel-3 contains no point.

分组：我们根据它们所处的体素对点进行分组。由于距离，遮挡，物体的相对姿势和非均匀采样等因素，LiDAR 点云稀疏并且在整个空间中具有高度可变的点密度。因此，在分组之后，体素将包含可变数量的点。图 2 中示出了图示，其中 Voxel-1 具有比 Voxel-2 和 Voxel-4 多得多的点，而 Voxel-3 不包含点。

Random Sampling: Typically a high-definition LiDAR point cloud is composed of ~100k points. Directly processing all the points not only imposes increased memory/efficiency burdens on the computing platform, but also highly variable point density throughout the space might bias the detection. To this end, we randomly sample a fixed number, T of points from those voxels containing more than T points. This sampling strategy has two purposes, (1) computational savings (see Section 2.3 for details); and (2) decreases the imbalance of points between the voxels which reduces the sampling bias, and adds more variation to training.

随机采样：通常，高清晰度 LiDAR 点云由~100k 点组成。直接处理所有点不仅会增加计算平台上的内存/效率负担，而且整个空间中高度可变的点密度可能会使检测偏差。为此，我们从包含超过 T 点的那些体素中随机采样固定数量的 T 点。这种抽样策略有两个目的，(1)计算节省（详见 2.3 节）; (2)减少体素之间点的不平衡，减少采样偏差，并增加训练的变化。

Stacked Voxel Feature Encoding: The key innovation is the chain of VFE layers. For simplicity, Figure 2 illustrates the hierarchical feature encoding process for one voxel. Without loss of generality, we use VFE Layer-1 to describe the details in the following paragraph. Figure 3 shows the architecture for VFE Layer-1.

堆叠体素特征编码: 关键创新是 VFE 层链。为简单起见，图 2 说明了一个体素的分层特征编码过程。在不失一般性的情况下，我们使用 VFE Layer-1 来描述以下段落中的细节。图 3 显示了 VFE Layer-1 的体系结构。

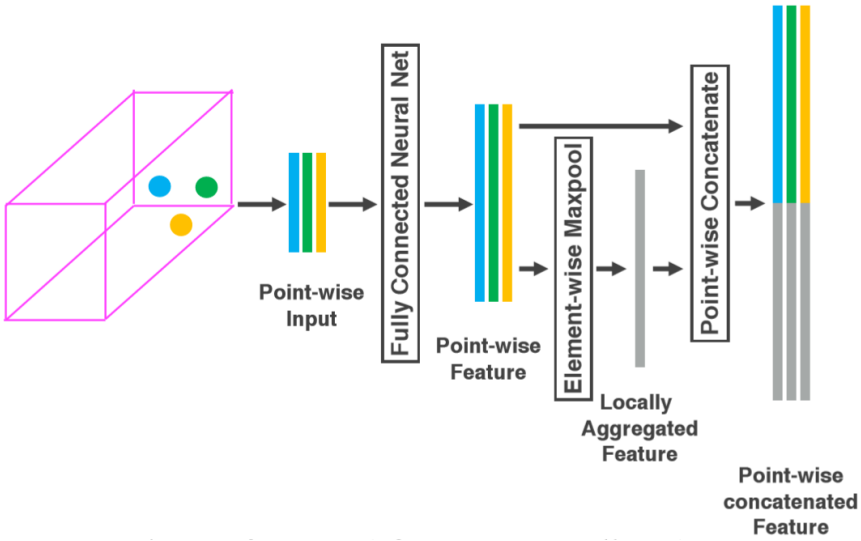

Figure 3. Voxel feature encoding layer.

图 3. Voxel 特征编码层。

Denote $V=\{p_i=[x_i,y_i,z_i,r_i]^T\in\mathbb{R}^4\}_{i=1\ldots t}$ as a non-empty voxel containing $t\le T$ LiDAR points, where p_i contains XYZ coordinates for the $i-th$ point and r_i is the received reflectance. We first compute the local mean as the centroid of all the points in V, denoted as (v_x,v_y,v_z) . Then we augment each point p_i with the relative offset w.r.t. the centroid and obtain the input feature set $V_{in}=\{p_i=[x_i,y_i,z_i,r_i,x_i-v_x,y_i-v_y,z_i-v_z]^T\in\mathbb{R}^7\}_{i=1\ldots t}$ Next, each p_i is transformed through the fully connected

将 $V=\{p_i=[x_i,y_i,z_i,r_i]^T\in\mathbb{R}^4\}_{i=1\ldots t}$ 表示为包含 $t\le T$ LiDAR 点的非空体素，其中 p_i 包含第 i 个点的 XYZ 坐标，r_i 是接收的反射率。我们首先将局部均值计算为 V 中所有点的质心，表示为 (v_x,v_y,v_z)。然后我们用相对质心偏移量 w.r.t 来扩充每个点 p_i 并获得输入特征集 $V_{in}=\{p_i=[x_i,y_i,z_i,r_i,x_i-v_x,y_i-v_y,z_i-v_z]^T\in\mathbb{R}^7\}_{i=1\ldots t}$ 接下来，每个 p_i 通过完全连接的网络（FCN）转换

network (FCN) into a feature space, where we can aggregate information from the point features $f_i \in \mathbb{R}^m$ to encode the shape of the surface contained within the voxel. The FCN is composed of a linear layer, a batch normalization (BN) layer, and a rectified linear unit (ReLU) layer. After obtaining point-wise feature representations, we use element-wise MaxPooling across all f_i associated to V to get the locally aggregated feature $f \in R^m$ for V. Finally, we augment each f_i with f to form the point-wise concatenated feature as $f_i^{out} = [f_i^T, \tilde{f}^T]^T \in R^{2m}$ .Thus we obtain the output feature set $V_{out} = \{f_i^{out}\}_{i...t}$. All non-empty voxels are encoded in the same way and they share the same set of parameters in FCN.

为特征空间，我们可以从点特征 $f_i \in \mathbb{R}^m$ 聚合信息，以编码体素中包含的表面的形状。FCN 由线性层，批量归一化（BN）层和整流线性单元（ReLU）层组成。在获得逐点特征表示之后，我们在与 V 相关联的所有 f_i 上使用逐元素 MaxPooling 来获得 V 的局部聚合特征 $f \in R^m$ 。最后，我们用 f 增加每个 f_i 以形成逐点连接特征，如 $f_i^{out} = [f_i^T, \tilde{f}^T]^T \in R^{2m}$ 。因此我们得到输出特征集 $V_{out} = \{f_i^{out}\}_{i...t}$ 。所有非空体素都以相同的方式编码，它们在 FCN 中共享同一组参数。

We use $VFE - i(c_{in}, c_{out})$ to represent the $i-th$ VFE layer that transforms input features of dimension c_{in} into output features of dimension c_{out}. The linear layer learns a matrix of size $c_{in} \times (c_{out} / 2)$, and the point-wise concatenation yields the output of dimension c_{out}.

我们使用 $VFE - i(c_{in}, c_{out})$ 来表示第 $i-th$ 个 VFE 层，它将尺寸 c_{in} 的输入特征转换为尺寸 c_{out} 的输出特征。线性层学习大小为 $c_{in} \times (c_{out} / 2)$ 的矩阵，并且逐点连接产生维度 c_{out} 的输出。

Because the output feature combines both point-wise features and locally aggregated feature, stacking VFE layers encodes point interactions within a voxel and enables the final feature representation to learn descriptive shape information. The voxel-wise feature is obtained by transforming the output of $VFE - n$ into $\mathbb{R}^C$ via FCN and applying element-wise Maxpool where C is the dimension of the voxel- wise feature, as shown in Figure 2.

由于输出要素结合了逐点特征和局部聚合特征，因此堆叠 VFE 层可对体素内的点交互进行编码，并使最终要素表示能够学习描述性形状信息。通过 FCN 将 $VFE - n$ 的输出转换为 $\mathbb{R}^C$ 并应用逐元素 Maxpool，其中 C 是体素特征的维度，获得体素方式特征，如图 2 所示。

Sparse Tensor Representation: By processing only the non-empty voxels, we obtain a list of voxel features, each uniquely associated to the spatial coordinates of a particular non-empty voxel. The obtained list of voxel-wise features can be represented as a sparse 4 D tensor, of size C×D′×H′×W′ as shown in Figure 2. Although the point cloud contains ~100k points, more than 90% of voxels typically are empty. Representing non-empty voxel features as a sparse tensor greatly reduces the memory usage and computation cost during

稀疏张量表示: 通过仅处理非空体素，我们获得体素特征列表，每个体素特征与特定非空体素的空间坐标唯一地相关联。获得的体素特征列表可以表示为稀疏的 4 D 张量，大小为 C×D'×H'×W'，如图 2 所示。虽然点云包含~100k 点，但超过 90％体素通常是空的。将非空体素特征表示为稀疏张量大大降低了反向传播期间的内存使用和计算成本，并且是我们有效实现中的关键步骤。

backpropagation, and it is a critical step in our efficient implementation.

2.1.2 Convolutional Middle Layers

We use $ConvMD(c_{in}, c_{out}, \mathbf{k}, \mathbf{s}, \mathbf{p})$ to represent an M−dimensional convolution operator where c_{in} and c_{out} are the number of input and output channels, **k**, **s**, and **p** are the M -dimensional vectors corresponding to kernel size, stride size and padding size respectively. When the size across the M -dimensions are the same, we use a scalar to represent the size e.g. k for **k** = (k, k, k).

Each convolutional middle layer applies 3D convolution, BN layer, and ReLU layer sequentially. The convolutional middle layers aggregate voxel-wise features within a progressively expanding receptive field, adding more context to the shape description. The detailed sizes of the filters in the convolutional middle layers are explained in Section 3.

2.1.3 Region Proposal Network

Recently, region proposal networks [34] have become an important building block of top-performing object detection frameworks [40], [5], [23]. In this work, we make several key modifications to the RPN architecture proposed in [34], and combine it with the feature learning network and convolutional middle layers to form an end-to-end trainable pipeline.

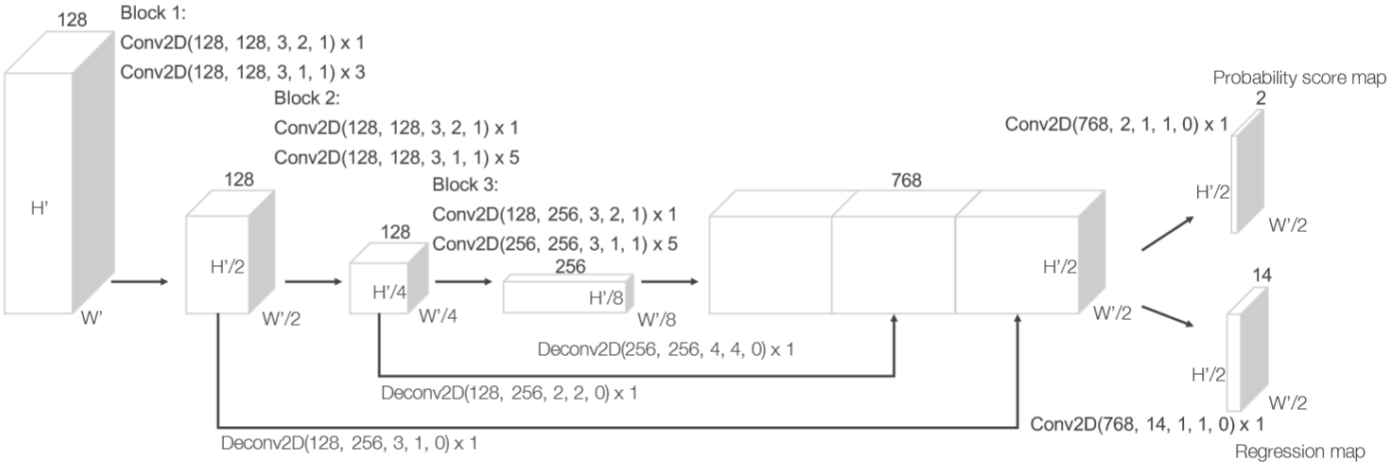

Figure 4. Region proposal network architecture.

The input to our RPN is the feature map provided by the convolutional middle layers. The architecture of this network is illustrated in Figure 4. The network has three blocks of fully convolutional layers. The first layer of each block downsamples the feature map by half via a convolution with a stride size of 2, followed by a sequence of convolutions of stride 1 (xq means q applications of the filter). After each

2.1.2 中间卷积层

我们使用$ConvMD(c_{in}, c_{out}, \mathbf{k}, \mathbf{s}, \mathbf{p})$来表示 M 维卷积算子，其中$c_{in}$和$c_{out}$是输入和输出通道的数量，**k**,**s** 和 **p** 是对应于内核的 M 维向量尺寸，步幅大小和填充尺寸。当 M 维度的大小相同时，我们使用标量来表示大小，例如 k 代表 **k** = (k, k, k)。

每个卷积中间层顺序地应用 3D 卷积，BN 层和 ReLU 层。卷积中间层在逐渐扩展的感受域内聚集体素方向特征，为形状描述添加更多上下文。卷积中间层中滤波器的详细尺寸将在第 3 节中介绍。

2.1.3 区域提案网络

最近，区域提案网络[34]已成为表现最佳的物体检测框架的重要组成部分[40]，[5]，[23]。在这项工作中，我们对[34]中提出的 RPN 架构进行了几次关键修改，并将其与特征学习网络和卷积中间层相结合，形成端到端的可训练流程。

图 4. 区域提案网络架构。

我们的 RPN 的输入是卷积中间层提供的特征映射。该网络的体系结构如图 4 所示。该网络有三个完全卷积层的块。每个块的第一层通过步幅大小为 2 的卷积将特征图缩减一半，然后是步幅 1 的卷积序列(xq 表示滤波器的 q 个应用)。在每个卷积层之后，BN 和 ReLU 操作被应用。然后，我们将每个块的输出上采样到固定大小并连接以构建高分辨率特征映射。最后，将该

convolution layer, BN and ReLU operations are applied. We then upsample the output of every block to a fixed size and concatanate to construct the high resolution feature map. Finally, this feature map is mapped to the desired learning targets: (1) a probability score map and (2) a regression map.

特征图映射到期望的学习目标：(1)概率分数图和(2)回归图。

2.2 Loss Function

Let $\{\alpha_i^{pos}\}_{i=1...N_{pos}}$ be the set of N_{pos} positive anchors and $\{\alpha_j^{neg}\}_{j=1...N_{neg}}$ be the set of N_{neg} negative anchors. We parameterize a 3D ground truth box as $(x_c^g, y_c^g, z_c^g, l^g, w^g, h^g, \theta^g)$ where x_c^g, y_c^g, z_c^g represent the center location l^g, w^g, h^g are length, width, height of the box, and θ^g is the yaw rotation around Z-axis. To retrieve the ground truth box from a matching positive anchor parameterized as $(x_c^a, y_c^a, z_c^a, l^a, w^a, h^a, \theta^a)$ we define the residual vector $u^* \in \mathbb{R}^7$ containing the 7 regression targets corresponding to center location Δx, Δy, Δz, three dimensions Δl, Δw, Δh, and the rotation Δθ , which are computed as:

2.2 损失函数

设$\{\alpha_i^{pos}\}_{i=1...N_{pos}}$是$N_{pos}$正锚的集合，$\{\alpha_j^{neg}\}_{j=1...N_{neg}}$是$N_{neg}$负锚的集合。我们将 3D 真实框参数化为$(x_c^g, y_c^g, z_c^g, l^g, w^g, h^g, \theta^g)$，其中$x_c^g, y_c^g, z_c^g$表示中心位置$l^g, w^g, h^g$是长度，宽度，框的高度和$\theta^g$是围绕 Z 轴的偏转旋转。为了从参数化为$(x_c^a, y_c^a, z_c^a, l^a, w^a, h^a, \theta^a)$的匹配正锚点检索真实框，我们定义了包含对应于中心位置 Δx, Δy, Δz 的 7 个回归目标的残差向量$u^* \in \mathbb{R}^7$，三个维度 Δl, Δw, Δh 和旋转 Δθ 计算如下：

$$\begin{aligned} &\Delta x = \frac{x_c^g - x_c^a}{d^a}, \Delta y = \frac{y_c^g - y_c^a}{d^a}, \Delta z = \frac{z_c^g - z_c^a}{h^a}, \\ &\Delta l = \log(\frac{l^g}{l^a}), \Delta w = \log(\frac{w^g}{w^a}), \Delta h = \log(\frac{h^g}{h^a}), \\ &\Delta\theta = \theta^g - \theta^a \end{aligned} \quad (1)$$

where $d^a = \sqrt{(l^a)^2 + (w^a)^2}$ is the diagonal of the base of the anchor box. Here, we aim to directly estimate the oriented 3D box and normalize Δx and Δy homogeneously with the diagonal da, which is different from [34], [40], [22], [21], [4], [3], [5]. We define the loss function as follows:

其中$d^a = \sqrt{(l^a)^2 + (w^a)^2}$是锚箱底部的对角线。在这里，我们的目标是直接估计定向的 3D 框并用对角线 da 均匀化 Δx 和 Δy，这与[34], [40], [22], [21], [4], [3], [5]不同。我们将损失函数定义如下：

$$\begin{aligned} L &= \alpha \frac{1}{N_{\text{pos}}} \sum_i L_{\text{cls}}(p_i^{\text{pos}}, 1) + \beta \frac{1}{N_{\text{neg}}} \sum_j L_{\text{cls}}(p_j^{\text{neg}}, 0) \\ &+ \frac{1}{N_{\text{pos}}} \sum_i L_{\text{reg}}(\mathbf{u}_i, \mathbf{u}_i^*) \end{aligned} \quad (2)$$

where p_i^{pos} and p_j^{neg} represent the softmax output for positive anchor α_i^{pos} and negative anchor α_j^{neg}

其中p_i^{pos}和p_j^{neg}分别代表正锚定α_i^{pos}和负锚定α_j^{neg}的 softmax 输出，而$u_i \in \mathbb{R}^7$和$u_i^* \in \mathbb{R}^7$分别是正锚定

respectively, while $u_i \in \mathbb{R}^7$ and $u_i^* \in \mathbb{R}^7$ are the regression output and ground truth for positive anchor α_i^{pos}. The first two terms are the normalized classification loss for $\{\alpha_i^{pos}\}_{i=1...N_{pos}}$ and $\{\alpha_j^{neg}\}_{j=1...N_{neg}}$, where the L_{cls} stands for binary cross entropy loss and α, β are postive constants balancing the relative importance. The last term L_{reg} is the regression loss, where we use the SmoothL1 function [12], [34].

伴随的回归输出和基本事实。前两个项是 $\{\alpha_i^{pos}\}_{i=1...N_{pos}}$ 和 $\{\alpha_j^{neg}\}_{j=1...N_{neg}}$ 的归一化分类损失，其中 L_{cls} 代表二元交叉熵损失，α, β 是平衡相对重要性的正定常数。最后一个术语 L_{reg} 是回归损失，我们使用 SmoothL1 函数[12]，[34]。

2.3 Efficient Implementation

GPUs are optimized for processing dense tensor structures. The problem with working directly with the point cloud is that the points are sparsely distributed across space and each voxel has a variable number of points. We devised a method that converts the point cloud into a dense tensor structure where stacked VFE operations can be processed in parallel across points and voxels.

2.3 有效实施

GPU 针对处理密集张量结构进行了优化。直接使用点云的问题在于，这些点稀疏地分布在空间中，并且每个体素具有可变数量的点。我们设计了一种方法，将点云转换为密集张量结构，其中堆叠的 VFE 操作可以跨点和体素并行处理。

The method is summarized in Figure 5. We initialize a K×T×7 dimensional tensor structure to store the voxel input feature buffer where K is the maximum number of non-empty voxels, T is the maximum number of points per voxel, and 7 is the input encoding dimension for each point. The points are randomized before processing. For each point in the point cloud, we check if the corresponding voxel already exists. This lookup operation is done efficiently in O (1) using a hash table where the voxel coordinate is used as the hash key. If the voxel is already initialized we insert the point to the voxel location if there are less than T points, otherwise the point is ignored. If the voxel is not initialized, we initialize a new voxel, store its coordinate in the voxel coordinate buffer, and insert the point to this voxel location. The voxel input feature and coordinate buffers can be constructed via a single pass over the point list, therefore its complexity is O(n). To further improve the memory/compute efficiency it is possible to only store a limited number of voxels (K) and ignore points coming from voxels with few points.

该方法总结在图 5 中。我们初始化 K×T×7 维张量结构以存储体素输入特征缓冲器，其中 K 是非空体素的最大数量，T 是每个体素的最大点数，并且 7 是每个点的输入编码维度。这些点在处理之前是随机的。对于点云中的每个点，我们检查相应的体素是否已存在。使用散列表在 O(1)中有效地完成该查找操作，其中体素坐标被用作散列密钥。如果体素已经初始化，如果小于 T 点，我们将该点插入体素位置，否则忽略该点。如果未初始化体素，我们初始化新体素，将其坐标存储在体素坐标缓冲区中，并将该点插入此体素位置。体素输入特征和坐标缓冲区可以通过点列表上的单次传递来构造，因此其复杂度为 O(n)。为了进一步提高存储器/计算效率，可以仅存储有限数量的体素(K)并忽略来自具有少量点的体素的点。

After the voxel input buffer is constructed, the stacked VFE only involves point level and voxellevel dense operations which can be computed on a GPU in parallel. Note that, after concatenation operations in VFE, we reset the features

在构造体素输入缓冲器之后，堆叠的 VFE 仅涉及可以在 GPU 上并行计算的点水平和体素密集操作。请注意，在 VFE 中进行连接操作后，我们将与空点对应的要素重置为零，以便它们不会影响计算的体素要素。最后，

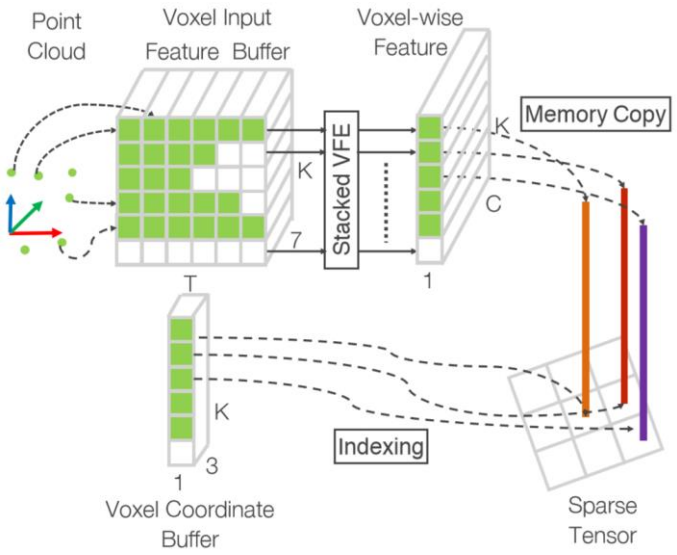

Figure 5. Illustration of efficient implementation.

图5. 高效实施的说明。

corresponding to empty points to zero such that they do not affect the computed voxel features. Finally, using the stored coordinate buffer we reorganize the computed sparse voxel-wise structures to the dense voxel grid. The following convolutional middle layers and RPN operations work on a dense voxel grid which can be efficiently implemented on a GPU.

使用存储的坐标缓冲区，我们将计算的稀疏体素结构重新组织到密集体素网格。以下卷积中间层和 RPN 操作在密集体素网格上工作，其可以在 GPU 上有效地实现。

3. Training Details

In this section, we explain the implementation details of the VoxelNet and the training procedure.

3.1 Network Details

Our experimental setup is based on the LiDAR specifications of the KITTI dataset [11].

Car Detection: For this task, we consider point clouds within the range of $[-3,1]\times[-40,40]\times[0,70.4]$ meters along Z, Y, X axis respectively. Points that are projected outside of image boundaries are removed [5]. We choose a voxel size of vD=0.4,vH=0.2,vW=0.2 meters, which leads to $D'=10$. $H'=400$. $W'=352$ We set T=35 as the maximum number of randomly sampled points in each non-empty voxel. We use two VFE layers VFE-l(7, 32) and VFE-2(32, 128). The final FCN maps VFE-2 output to R128 Thus our feature learning net generates a sparse tensor of shape $128 \times 10 \times 400 \times 352$. To aggregate voxel-wise features, we employ three convolution middle layers sequentially as Conv3D(128, 64, 3, (2,1,1), (1,1,1)), Conv3D(64, 64, 3, (1,1,1), (0,1,1)), and Conv3D(64, 64, 3, (2,1,1), (1,1,1)), which yields a 4D tensor of size $64 \times 2 \times 400 \times 352$. After reshaping, the input to RPN is a feature map of size $128 \times$

3. 训练细节

在本节中，我们将解释 VoxelNet 的实现细节和训练过程。

3.1 网络细节

我们的实验设置基于 KITTI 数据集的 LiDAR 规范[11]。

汽车检测：对于此任务，我们分别考虑沿 Z，Y，X 轴在 $[-3,1]\times[-40,40]\times[0.70.4]$米范围内的点云。投射到图像边界之外的点被删除[5]。我们选择体素大小 vD = 0.4，vH = 0.2，vW = 0.2 米，这导致 D'= 10。H'= 400。W'= 352 我们将 T = 35 设置为每个非空体素中的最大随机采样点数。我们使用两个 VFE 层 VFE-1（7,32）和 VFE-2（32,128）。最终的 FCN 将 VFE-2 输出映射到 R128 因此，我们的特征学习网生成一个 $128\times10\times400\times352$ 的稀疏张量。为了聚合体素方面的特征，我们采用三个卷积中间层顺序为 Conv3D(128, 64, 3, (2,1,1), (1,1,1)), Conv3D(64, 64, 3, (1,1,1), (0,1,1))和 Conv3D(64, 64, 3, (2,1,1), (1,1,1))，产生尺寸为 $64 \times 2 \times 400 \times 352$ 的 4D 张量重塑后，RPN 的输入是尺寸为 $128\times400\times352$ 的特征图，其中尺寸对应于 3D 张量的通道，高度和宽度。图 4 显示了此任务的详细网络体系结构。与[5]不同，我们

400 × 352, where the dimensions correspond to channel, height, and width of the 3D tensor. Figure 4 illustrates the detailed network architecture for this task. Unlike [5], we use only one anchor size, $l^a = 3.9, w^a = 1.6, h^a = 1.56$ meters, centered at $z_c^a = -1.0$ meters with two rotations, 0 and 90 degrees. Our anchor matching criteria is as follows: An anchor is considered as positive if it has the highest Intersection over Union (IoU) with a ground truth or its IoU with ground truth is above 0.6 (in bird's eye view). An anchor is considered as negative if the IoU between it and all ground truth boxes is less than 0.45. We treat anchors as don't care if they have 0.45 ≤ IoU < 0.6 with any ground truth. We set α=1.5 and β = 1 in Eqn.2.

只使用一个锚尺寸，$l^a = 3.9, w^a = 1.6, h^a = 1.56$ 米，以 $z_c^a = -1.0$ 米为中心，两次旋转，0 度和 90 度。我们的锚点匹配标准如下：如果锚点具有最高的交叉点（IoU）与地面实况或其地面实况的 IoU 高于 0.6（在鸟瞰图中），则锚点被视为正数。如果锚点与所有地面实况框之间的 IoU 小于 0.45，则认为该锚点为负。我们认为锚点如果它们有 0.45≤IOU<0.6 且没有任何基本事实就不在乎。我们在 Eqn.2 中设置 α= 1.5 和 β= 1。

Pedestrian and Cyclist Detection: The input range is [−3, 1] × [−20,20] × [0, 48] meters along Z, Y, Xaxis respectively. We use the same voxel size as for car detection, which yields D=10,H=200,W=240 We set T=45 in order to obtain more LiDAR points for better capturing shape information. The feature learning network and convolutional middle layers are identical to the networks used in the car detection task. For the RPN, we make one modification to block 1 in Figure 4 by changing the stride size in the first 2D convolution from 2 to 1. This allows finer resolution in anchor matching, which is necessary for detecting pedestrians and cyclists. We use anchor size $l^a = 0.8, w^a = 0.6, h^a = 1.73$ meters centered at $z_c^a = -0.6$ meters with 0 and 90 degrees rotation for pedestrian detection and use anchor size $l^a = 1.76, w^a = 0.6, h^a = 1.73$ meters centered at $z_c^a = -0.6$ with 0 and 90 degrees rotation for cyclist detection. The specific anchor matching criteria is as follows: We assign an anchor as postive if it has the highest IoU with a ground truth, or its IoU with ground truth is above 0.5. An anchor is considered as negative if its IoU with every ground truth is less than 0.35. For anchors having 0.35 ≤ IoU ≤ 0.5 with any ground truth, we treat them as don't care.

行人和骑车人检测：输入范围分别沿 Z，Y，X 轴为[-3,1]×[-20,20]×[0,48]米。我们使用与汽车检测相同的体素尺寸，其产生 D = 10，H = 200，W = 240 我们设置 T = 45 以获得更多的 LiDAR 点以更好地捕获形状信息。特征学习网络和卷积中间层与汽车检测任务中使用的网络相同。对于 RPN，我们通过将第一个 2D 卷积中的步幅大小从 2 改为 1 来对图 4 中的块 1 进行一次修改。这允许锚点匹配中更精细的分辨率，这对于检测行人和骑车者是必要的。我们使用锚尺寸 $l^a = 0.8, w^a = 0.6, h^a = 1.73$ 米，以 $z_c^a = -0.6$ 米为中心，0 度和 90 度旋转用于行人检测并使用锚尺寸 $l^a = 1.76, w^a = 0.6, h^a = 1.73$ 米为中心 $z_c^a = -0.6$ 米，旋转 0 度和 90 度，用于骑车人检测。具体的锚匹配标准如下：如果锚具有最高的具有基础事实的 IoU，或者其具有基本事实的 IoU 高于 0.5，则将锚分配为 postive。如果每个基本事实的 IoU 小于 0.35，则锚被认为是否定的。对于具有 0.35≤IOU≤0.5 且具有任何基本事实的锚，我们将它们视为无关紧要。

During training, we use stochastic gradient descent (SGD) with learning rate 0.01 for the first 150 epochs and decrease the learning rate to 0.001 for the last 10 epochs. We use a batchsize of 16 point clouds.

在训练期间，我们使用随机梯度下降（SGD），前 150 个时期的学习率为 0.01，并且在过去的 10 个时期内将学习率降低到 0.001。我们使用 16 帧点云的批量大小。

3.2 Data Augmentation

With less than 4000 training point clouds, training our network from scratch will inevitably suffer from overfitting. To reduce this issue, we introduce three different forms of data augmentation. The augmented training data are generated on-the-fly without the need to be stored on disk [20].

Define set $\mathbf{M} = \{\boldsymbol{p}_i = [x_i, y_i, z_i, r_i]^T \in \mathbb{R}^4\}_{i=1\ldots N}$ as the whole point cloud, consisting of *N* points. We parameterize a 3D bounding box $\boldsymbol{b}_i$ as $(\mathrm{x}_c, \mathrm{y}_c, \mathrm{z}_c, \mathrm{l}, \mathrm{w}, \mathrm{h}, \theta)$,where $\mathrm{x}_c, \mathrm{y}_c, \mathrm{z}_c$ are center locations, $\mathrm{l}, \mathrm{w}, \mathrm{h}$ are length, width, height, and θ is the yaw rotation around Z-axis. We define $\Omega_i = \{\boldsymbol{p} | x \in [\mathrm{x}_c - l/2, \mathrm{x}_c + l/2], y \in [\mathrm{y}_c - w/2, \mathrm{y}_c + w/2], z \in [\mathrm{z}_c - h/2, \mathrm{z}_c + h/2], \boldsymbol{p} \in \boldsymbol{M}\}$ as the set containing all LiDAR points within $\boldsymbol{b}_i$, where $\mathbf{p} = [\mathrm{x}, \mathrm{y}, \mathrm{z}, \mathrm{r}]$ denotes a particular LiDAR point in the whole set **M**.

The first form of data augmentation applies perturbation independently to each ground truth 3D bounding box together with those LiDAR points within the box. Specifically, around Z-axis we rotate $\boldsymbol{b}_i$ and the associated Ωi with respect to (xc, yc, zc) by a uniformally distributed random variable $\Delta\theta \in [-\pi/10, +\pi/10]$. then we add a translation (Δx, Δy, Δz) to the XYZ components of $\boldsymbol{b}_i$ and to each point in Ωi, where Δx, Δy, Δz are drawn independently from a Gaussian distribution with mean zero and standard deviation 1.0. To avoid physically impossible outcomes, we perform a collision test between any two boxes after the perturbation and revert to the original if a collision is detected. Since the perturbation is applied to each ground truth box and the associated LiDAR points independently, the network is able to learn from substantially more variations than from the original training data.

Secondly, we apply global scaling to all ground truth boxes $\boldsymbol{b}_i$ and to the whole point cloud M. Specifically, we multiply the XYZ coordinates and the three dimensions of each $\boldsymbol{b}_i$, and the XYZ coordinates of all points in M with a random variable drawn from uniform distribution [0.95,1.05]. Introducing global scale augmentation improves robustness of the network for detecting objects with various sizes and distances as shown in image-based classification [37], [18] and detection tasks [12], [17].

Finally, we apply global rotation to all ground truth boxes $\boldsymbol{b}_i$ and to the whole point cloud M. The rotation is applied along Z-axis and around (0,0,0). The global rotation offset

3.2 数据增强

只有不到 4000 个训练点云，从头开始训练我们的网络将不可避免地遭受过度拟合。为了减少这个问题，我们引入了三种不同形式的数据扩充。增强的训练数据是即时生成的，无需存储在磁盘上[20]。

定义集合$\mathbf{M} = \{\boldsymbol{p}_i = [x_i, y_i, z_i, r_i]^T \in \mathbb{R}^4\}_{i=1\ldots N}$作为整个点云，由 N 个点组成。我们将 3D 包围盒$\boldsymbol{b}_i$参数化为$(\mathrm{x}_c, \mathrm{y}_c, \mathrm{z}_c, \mathrm{l}, \mathrm{w}, \mathrm{h}, \theta)$，其中$\mathrm{x}_c, \mathrm{y}_c, \mathrm{z}_c$是中心位置，$\mathrm{l}, \mathrm{w}, \mathrm{h}$是长度，宽度，高度，$\theta$是偏航围绕 Z 轴旋转。我们定义$\Omega_i = \{\boldsymbol{p} | x \in [\mathrm{x}_c - l/2, \mathrm{x}_c + l/2], y \in [\mathrm{y}_c - w/2, \mathrm{y}_c + w/2], z \in [\mathrm{z}_c - h/2, \mathrm{z}_c + h/2], \boldsymbol{p} \in \boldsymbol{M}\}$作为包含$\boldsymbol{b}_i$内所有 LiDAR 点的集合，其中$\mathbf{p} = [\mathrm{x}, \mathrm{y}, \mathrm{z}, \mathrm{r}]$表示整个集合 **M** 中的特定 LiDAR 点。

第一种形式的数据增强将扰动独立地应用于每个真实 3D 边界框以及框内的那些 LiDAR 点。具体地，在 Z 轴周围，我们通过均匀分布的随机变量 $\Delta\theta \in [-\pi/10, +\pi/10]$将$\boldsymbol{b}_i$和相关的 Ωi 相对于(xc, yc, zc)旋转。然后我们向$\boldsymbol{b}_i$的 XYZ 分量和 Ωi 中的每个点添加平移(Δx, Δy, Δz)，其中 Δx，Δy，Δz 独立于具有平均零和标准偏差 1.0 的高斯分布。为了避免物理上不可能的结果，我们在扰动之后在任意两个盒子之间执行碰撞测试，并且如果检测到碰撞则恢复到原始盒子。由于扰动被独立地应用于每个地面实况框和相关联的 LiDAR 点，因此网络能够从比原始训练数据显着更多的变化中学习。

其次，我们将全局缩放应用于所有真值框$\boldsymbol{b}_i$和整个点云 M.具体来说，我们将 XYZ 坐标和每个$\boldsymbol{b}_i$的三个维度相乘，并将 M 中所有点的 XYZ 坐标与从中抽取的随机变量相乘均匀分布[0.95,1.05]。引入全局比例增强提高了网络的稳健性，用于检测具有各种大小和距离的对象，如基于图像的分类[37]，[18]和检测任务[12]，[17]所示。

最后，我们将全局旋转应用于所有地面实况框$\boldsymbol{b}_i$和整个点云 M.旋转沿 Z 轴和（0,0,0）应用。通过从均匀分布[$-\pi/4, +\pi/4$]采样确定全局旋转偏移。通过旋转整个点云，

is determined by sampling from uniform distribution $[-\pi/4,+\pi/4]$. By rotating the entire point cloud, we simulate the vehicle making a turn.

我们模拟车辆转弯。

4. Experiments

We evaluate VoxelNet on the KITTI 3D object detection benchmark [11] which contains 7,481 training images/point clouds and 7,518 test images/point clouds, covering three categories: Car, Pedestrian, and Cyclist. For each class, detection outcomes are evaluated based on three difficulty levels: easy, moderate, and hard, which are determined according to the object size, occlusion state, and truncation level. Since the ground truth for the test set is not available and the access to the test server is limited, we conduct comprehensive evaluation using the protocol described in [4], [3], [5] and subdivide the training data into a training set and a validation set, which results in 3,712 data samples for training and 3,769 data samples for validation. The split avoids samples from the same sequence being included in both the training and the validation set [3]. Finally we also present the test results using the KITTI server.

4. 实验

我们在 KITTI 3D 物体检测基准[11]上评估 VoxelNet，其中包含 7,481 个训练图像/点云和 7,518 个测试图像/点云，涵盖三类：汽车，行人和骑车人。对于每个类，检测结果基于三个难度级别来评估：简单，中等和难，其根据对象大小，遮挡状态和截断水平确定。由于测试装置的真实值不可用且测试服务器的访问受限，我们使用[4]，[3]，[5]中描述的协议进行综合评估，并将训练数据细分为训练集和验证集，产生 3,712 个用于训练的数据样本和 3,769 个用于验证的数据样本。分割避免了来自同一序列的样本被包含在训练集和验证集中[3]。最后，我们还使用 KITTI 服务器呈现测试结果。

Method	Modality	Car			Pedestrian			Cyclist		
		Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
Mono3D [3]	Mono	5.22	5.19	4.13	N/A	N/A	N/A	N/A	N/A	N/A
3DOP [4]	Stereo	12.63	9.49	7.59	N/A	N/A	N/A	N/A	N/A	N/A
VeloFCN [22]	LiDAR	40.14	32.08	30.47	N/A	N/A	N/A	N/A	N/A	N/A
MV (BV+FV) [5]	LiDAR	86.18	77.32	76.33	N/A	N/A	N/A	N/A	N/A	N/A
MV (BV+FV+RGB) [5]	LiDAR+Mono	86.55	78.10	76.67	N/A	N/A	N/A	N/A	N/A	N/A
HC-baseline	LiDAR	88.26	78.42	77.66	58.96	53.79	51.47	63.63	42.75	41.06
VoxelNet	LiDAR	**89.60**	**84.81**	**78.57**	**65.95**	**61.05**	**56.98**	**74.41**	**52.18**	**50.49**

Table 1. Performance comparison in bird's eye view detection: average precision (in %) on KITTI validation set.

Method	Modality	Car			Pedestrian			Cyclist		
		Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
Mono3D [3]	Mono	2.53	2.31	2.31	N/A	N/A	N/A	N/A	N/A	N/A
3DOP [4]	Stereo	6.55	5.07	4.10	N/A	N/A	N/A	N/A	N/A	N/A
VeloFCN [22]	LiDAR	15.20	13.66	15.98	N/A	N/A	N/A	N/A	N/A	N/A
MV (BV+FV) [5]	LiDAR	71.19	56.60	55.30	N/A	N/A	N/A	N/A	N/A	N/A
MV (BV+FV+RGB) [5]	LiDAR+Mono	71.29	62.68	56.56	N/A	N/A	N/A	N/A	N/A	N/A
HC-baseline	LiDAR	71.73	59.75	55.69	43.95	40.18	37.48	55.35	36.07	34.15
VoxelNet	LiDAR	**81.97**	**65.46**	**62.85**	**57.86**	**53.42**	**48.87**	**67.17**	**47.65**	**45.11**

Table 2. Performance comparison in 3D detection: average precision (in %) on KITTI validation set.

For the Car category, we compare the proposed method with several top-performing algorithms, including image based approaches: Mono3D [3] and 3DOP [4]; LiDAR based approaches: VeloFCN [22] and 3D-FCN [21]; and a multi-modal approach MV [5]. Mono3D [3], 3DOP [4] and MV [5] use a pre-trained model for initialization whereas we train VoxelNet from scratch using only the LiDAR data provided in KITTI.

对于汽车类别，我们将提出的方法与几种表现最佳的算法进行比较，包括基于图像的方法：Mono3D [3]和 3DOP [4];基于 LiDAR 的方法：VeloFCN [22]和 3D-FCN [21]; 和多模态方法 MV [5]。Mono3D [3]，3DOP [4]和 MV [5]使用预先训练的模型进行初始化，而我们仅使用 KITTI 中提供的 LiDAR 数据从头开始训练 VoxelNet。

To analyze the importance of end-to-end learning, we implement a strong baseline that is derived from the Vox-

为了分析端到端学习的重要性，我们实施了一个强大的基线，该基线源自 Vox-elNet 架构，但使用手工制作的

elNet architecture but uses hand-crafted features instead of the proposed feature learning network. We call this model the hand-crafted baseline (HC-baseline). HC-baseline uses the bird's eye view features described in [5] which are computed at 0.lm resolution. Different from [5], we increase the number of height channels from 4 to 16 to capture more detailed shape information- further increasing the number of height channels did not lead to performance improvement. We replace the convolutional middle layers of VoxelNet with similar size 2D convolutional layers, which are Conv2D(16, 32, 3, 1, 1), Conv2D(32, 64, 3, 2, 1), Conv2D(64, 128, 3, 1, 1). Finally RPN is identical in VoxelNet and HC-baseline. The total number of parameters in HC-baseline and VoxelNet are very similar. We train the HC-baseline using the same training procedure and data augmentation described in Section 3.

功能而不是建议的功能学习网络。我们将此模型称为手工制作的基线（HC 基线）。HC-基线使用[5]中描述的鸟瞰图特征，其以 0.1l 分辨率计算。与[5]不同，我们将高度通道的数量从 4 增加到 16 以捕获更详细的形状信息 - 进一步增加高度通道的数量不会导致性能提高。我们将 VoxelNet 的卷积中间层替换为类似大小的 2D 卷积层，即 Conv2D（16,32,3,1,1），Conv2D（32,64,3,2,1），Conv2D（64,128,3），1,1）。最后，RPN 在 VoxelNet 和 HC-baseline 中是相同的。HC-baseline 和 VoxelNet 中的参数总数非常相似。我们使用第 3 节中描述的相同训练程序和数据增强训练 HC 基线。

4.1 Evaluation on Kitti Validation Set

Metrics：We follow the official KITTI evaluation protocol, where the IoU threshold is 0.7 for class Car and is 0.5 for class Pedestrian and Cyclist. the IoU threshold is the same for both bird's eye view and full 3D evaluation. We compare the methods using the average precision (AP) metric.

4.1 Kitti 验证集上的评估

度量标准：我们遵循官方的 KITTI 评估协议，其中 Car 类别的 IoU 阈值为 0.7， Pedestrian and Cyclist 类别的 IoU 阈值为 0.5。鸟瞰图和全 3D 评估的 IoU 阈值相同。我们使用平均精度(AP)度量来比较这些方法。

Evaluation in Bird's Eye View：The evaluation result is presented in Table 1. VoxelNet consistently outperforms all the competing approaches across all three difficulty levels. HC-baseline also achieves satisfactory performance compared to the state-of-the-art [5], which shows that our base region proposal network (RPN) is effective. For Pedestrian and Cyclist detection tasks in bird's eye view, we compare the proposed VoxelNet with HC-baseline. VoxelNet yields substantially higher AP than the HC-baseline for these more challenging categories, which shows that end-to-end learning is essential for point-cloud based detection.

鸟瞰图评估：评估结果如表 1 所示。VoxelNet 在所有三个难度级别上始终优于所有竞争方法。与现有技术相比，HC-基线也达到了令人满意的性能[5]，这表明我们的基地提案网络（RPN）是有效的。对于鸟瞰图中的行人和骑车人检测任务，我们将建议的 VoxelNet 与 HC 基线进行比较。对于这些更具挑战性的类别，VoxelNet 产生的 AP 比 HC 基线高得多，这表明端到端学习对于基于点云的检测至关重要。

Evaluation in 3D：Compared to the bird's eye view detection, which requires only accurate localization of objects in the 2D plane, 3D detection is a more challenging task as it requires finer localization of shapes in 3D space. Table 2 summarizes the comparison. For the class Car, VoxelNet significantly outperforms all other approaches in AP across all difficulty levels. Specifically, using only LiDAR, VoxelNet significantly outperforms the state-of-the-art method MV (BV+FV+RGB) [5] based on LiDAR+RGB, by 10.68%, 2.78% and 6.29% in easy, moderate, and hard levels respectively. HC-baseline achieves similar accuracy to the MV [5] method.

3D 评估：与只需要在 2D 平面中精确定位物体的鸟瞰图检测相比，3D 检测是一项更具挑战性的任务，因为它需要在 3D 空间中更精细地定位形状。表 2 总结了比较。对于 Car 类，VoxelNet 在所有难度级别上明显优于 AP 中的所有其他方法。具体来说，仅使用 LiDAR，VoxelNet 明显优于基于 LiDAR + RGB 的最先进方法 MV（BV + FV + RGB）[5]，简单，中等，以及 10.68％，2.78％和 6.29％。硬水平分别。HC-基线达到与 MV [5]方法类似的精确度。

As in the bird's eye view evaluation, we also compare VoxelNet with HC-baseline on 3D Pedestrian and Cyclist detection. Due to the high variation in 3D poses and shapes, successful detection of these two categories requires better 3D shape representation. As shown in Table 2 the improved performance of VoxelNet is emphasized for more challenging 3D detection tasks (from ~8% improvement in bird's eye view to ~12% improvement on 3D detection) which suggests that VoxelN et is more effective in capturing 3D shape information than hand-crafted features.

在鸟瞰视图评估中，我们还将 VoxelNet 与行人和骑行者检测的 HC 基线进行比较。由于 3D 姿势和形状的高度变化，成功检测这两个类别需要更好的 3D 形状表示。如表 2 所示，VoxelNet 的改进性能强调了更具挑战性的 3D 检测任务（鸟瞰图从~8％提高到 3D 检测提高约 12％），这表明 VoxelN 等在捕获 3D 形状信息方面更有效比手工制作的功能。

4.2 Evaluation on Kitti Test Set

We evaluated VoxelNet on the KITTI test set by submitting detection results to the official server. The results are summarized in Table 3. VoxelNet, significantly outperforms the previously published state-of-the-art [5] in all the tasks (bird‘s eye view and 3D detection) and all difficulties. We would like to note that many of the other leading methods listed in KITTI benchmark use both RGB images and LiDAR point clouds whereas VoxelNet uses only LiDAR.

4.1 Kitti 测试集上的评估

我们通过向官方服务器提交检测结果来评估 KITTI 测试集上的 VoxelNet。结果总结在表 3 中.VoxelNet 在所有任务（鸟瞰图和 3D 检测）和所有困难中明显优于先前发布的最新技术[5]。我们想要注意的是，KITTI 基准测试中列出的许多其他主要方法都使用 RGB 图像和 LiDAR 点云，而 VoxelNet 仅使用 LiDAR。

Benchmark	Easy	Moderate	Hard
Car (3D Detection)	77.47	65.11	57.73
Car (Bird's Eye View)	89.35	79.26	77.39
Pedestrian (3D Detection)	39.48	33.69	31.51
Pedestrian (Bird's Eye View)	46.13	40.74	38.11
Cyclist (3D Detection)	61.22	48.36	44.37
Cyclist (Bird's Eye View)	66.70	54.76	50.55

Table 3. Performance evaluation on KITTI test set.

Figure 6. Qualitative results. For better visualization 3D boxes detected using lidar are projected on to the RGB images.

图 6. 定性结果。为了更好的可视化，使用激光雷达检测到的 3D 盒子被投射到 RGB 图像上。

We present several 3D detection examples in Figure 6. For better visualization 3D boxes detected using LiDAR are projected on to the RGB images. As shown, VoxelNet

我们在图 6 中提供了几个 3D 检测示例。为了更好地可视化，使用 LiDAR 检测到的 3D 盒子被投影到 RGB 图像上。如图所示，VoxelNet 在所有类别中提供高度精确

provides highly accurate 3D bounding boxes in all categories.

的 3D 边界框。

The inference time for VoxelNet is 33ms: the voxel input feature computation takes 5ms, feature learning network takes 16ms, convolutional middle layers take 1ms, and region proposal network takes 11ms on a TitanX GPU and 1.7Ghz CPU.

VoxelNet 的推理时间为 33ms：体素输入特征计算需要 5ms，特征学习网络需要 16ms，卷积中间层需要 1ms，区域提议网络需要 11ms 才能使用 TitanX GPU 和 1.7Ghz CPU。

5. Conclusion

Most existing methods in LiDAR-based 3D detection rely on hand-crafted feature representations, for example, a bird's eye view projection. In this paper, we remove the bottleneck of manual feature engineering and propose VoxelNet, a novel end-to-end trainable deep architecture for point cloud based 3D detection. Our approach can operate directly on sparse 3D points and capture 3D shape information effectively. We also present an efficient implementation of VoxelNet that benefits from point cloud sparsity and parallel processing on a voxel grid. Our experiments on the KITTI car detection task show that VoxelNet outperforms state-of-the-art LiDAR based 3D detection methods by a large margin. On more challenging tasks, such as 3D detection of pedestrians and cyclists, VoxelNet also demonstrates encouraging results showing that it provides a better 3D representation. Future work includes extending Voxel-Net for joint LiDAR and image based end-to-end 3D detection to further improve detection and localization accuracy.

5. 结论

基于 LiDAR 的 3D 检测中的大多数现有方法依赖于手工制作的特征表示，例如鸟瞰视图投影。在本文中，我们消除了手动特征工程的瓶颈，并提出了 Vox-elNet，这是一种基于点云的 3D 检测的新型端到端可训练深度架构。我们的方法可以直接在稀疏的 3D 点上操作，并有效地捕获 3D 形状信息。我们还提供了一个有效的 VoxelNet 实现，它受益于点云稀疏性和体素网格上的并行处理。我们对 KITTI 汽车检测任务的实验表明，VoxelNet 在很大程度上优于最先进的基于 LiDAR 的 3D 检测方法。在更具挑战性的任务中，例如行人和骑自行车者的 3D 检测，VoxelNet 也展示了令人鼓舞的结果，表明它提供了更好的 3D 表示。未来的工作包括扩展 Voxel-Net 用于联合 LiDAR 和基于图像的端到端 3D 检测，以进一步提高检测和定位精度。

6. References

[1] Bariya P, Nishino K. Scale-hierarchical 3d object recognition in cluttered scenes[C]//2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2010: 1657-1664.

[2] Bo L, Ren X, Fox D. Depth kernel descriptors for object recognition[C]//2011 IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS). IEEE, 2011: 821-826.

[3] Chen X, Kundu K, Zhang Z, et al. Monocular 3d object detection for autonomous driving[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2016: 2147-2156.

[4] Chen X, Kundu K, Zhu Y, et al. 3d object proposals for accurate object class detection[C]//Advances in Neural Information Processing Systems (NIPS). 2015: 424-432.

[5] Chen X, Ma H, Wan J, et al. Multi-view 3d object detection network for autonomous driving[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2017: 1907-1915.

[6] Choi C, Taguchi Y, Tuzel O, et al. Voting-based pose estimation for robotic assembly using a 3D sensor[C]//2012 IEEE International Conference on Robotics and Automation(ICRA). IEEE, 2012: 1724-1731.

[7] Chua C S , Jarvis R . Point Signatures: A New Representation for 3D Object Recognition[J]. International Journal of Computer Vision, 1997, 25(1):63-85.

[8] Dorai C , Jain A K . COSMOS-A representation scheme for 3D free-form objects[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1997, 19(10):0-1130.

[9] Engelcke M, Rao D, Wang DZ, et al. Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks[C]//2017 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2017: 1355-1361.
[10] Enzweiler M , Gavrila D M . A multi-level mixture-of-experts framework for pedestrian classification[J]. Image Processing IEEE Transactions on, 2011, 20(10):2967-2979.
[11] Geiger A, Lenz P, Urtasun R. Are we ready for autonomous driving? the kitti vision benchmark suite[C]//2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2012: 3354-3361.
[12] Girshick R. Fast r-cnn[C]//Proceedings of the 2015 IEEE International Conference on Computer Vision(ICCV). IEEE, 2015: 1440-1448.
[13] Girshick R , Donahue J , Darrelland T , et al. Rich feature hierarchies for object detection and semantic segmentation[C]//2014 IEEE Conference on Computer Vision and Pattern Recognition(CVPR). IEEE, 2014: 580-587.
[14] Gomez-Ojeda R, Briales J, Gonzalez-Jimenez J. Plsvo: Semi-direct monocular visual odometry by combining points and line segments[C]// 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2016: 4211–4216.
[15] Gonzalez A , Villalonga G , Xu J , et al. Multiview random forest of local experts combining RGB and LIDAR data for pedestrian detection[C]// 2015 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2015: 356-361.
[16] Gonzalez A , Vazquez D , Lopez A M , et al. On-Board Object Detection: Multicue, Multimodal, and Multiview Random Forest of Local Experts[J]. IEEE Transactions on Cybernetics, 2016, 47(11):1-11.
[17] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition[C]// Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition(CVPR). IEEE, 2016: 770-778.
[18] A. G. Howard, "Some improvements on deep convolutional neural network based image classification", CoRR abs/1312.5402, 2013.
[19] A. E. Johnson, M. Hebert, "Using spin images for efficient object recognition in cluttered 3d scenes", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 21, no. 5, pp. 433-449, 1999.
[20] A. Krizhevsky, I. Sutskever, G. E. Hinton, F. Pereira, C. J. C. Burges, L. Bottou, K. Q. Weinberger, "Imagenet classification with deep convolutional neural networks", Advances in Neural Information Processing Systems 25, pp. 1097-1105, 2012.
[21] B. Li. 3d fully convolutional network for vehicle detection in point cloud. In IROS, 2017.
[22] B. Li, T. Zhang, and T. Xia. Vehicle detection from 3d lidar using fully convolutional network. In Robotics: Science and Systems, 2016.
[23] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Doll´ar. Focal loss for dense object detection. IEEE ICCV, 2017.
[24] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In ECCV, pages 21–37, 2016.
[25] D. Maturana and S. Scherer. 3D Convolutional Neural Networks for Landing Zone Detection from LiDAR. In ICRA, 2015.
[26] D. Maturana and S. Scherer. VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition. In IROS, 2015.
[27] A. Mian, M. Bennamoun, and R. Owens. On the repeatability and quality of keypoints for local feature-based 3d object retrieval from cluttered scenes. International Journal of Computer Vision, 89(2):348–361, Sep 2010.
[28] Y.-J. Oh and Y. Watanabe. Development of small robot for home floor cleaning. In Proceedings of the 41st SICE Annual Conference. SICE 2002., volume 5, pages 3222–3223 vol.5, Aug 2002.
[29] Y. Park, V. Lepetit, and W. Woo. Multiple 3d object tracking for augmented reality. In 2008 7th IEEE/ACM International Symposium on Mixed and Augmented Reality, pages 117– 120, Sept 2008.
[30] C. Premebida, J. Carreira, J. Batista, and U. Nunes. Pedestrian detection combining RGB and dense LIDAR data. In IROS, pages 0–1. IEEE, Sep 2014.
[31] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. Proc. Computer Vision and Pattern Recognition (CVPR), IEEE, 2017.
[32] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. arXiv preprint arXiv:1706.02413, 2017.
[33] J. Redmon and A. Farhadi. YOLO9000: better, faster, stronger. In IEEE Conference on Computer Vision and Pattern

Recognition (CVPR), 2017.
[34] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In Advances in Neural Information Processing Systems 28, pages 91–99. 2015.
[35] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (fpfh) for 3d registration. In 2009 IEEE International Conference on Robotics and Automation, pages 3212– 3217, 2009.
[36] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In CVPR 2011, pages 1297–1304, 2011.
[37] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. CoRR, abs/1409.1556, 2014.
[38] S. Song and M. Chandraker. Joint sfm and detection cues for monocular 3d localization in road scenes. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3734–3742, June 2015.
[39] S. Song and J. Xiao. Sliding shapes for 3d object detection in depth images. In European Conference on Computer Vision, Proceedings, pages 634–651, Cham, 2014. Springer International Publishing.
[40] S. Song and J. Xiao. Deep Sliding Shapes for amodal 3D object detection in RGB-D images. In CVPR, 2016.
[41] F. Stein and G. Medioni. Structural indexing: efficient 3-d object recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 14(2):125–145, 1992.
[42] O. Tuzel, M.-Y. Liu, Y. Taguchi, and A. Raghunathan. Learning to rank 3d features. In 13th European Conference on Computer Vision, Proceedings, Part I, pages 520–535, 2014.
[43] D. Z. Wang and I. Posner. Voting for voting in online point cloud object detection. In Proceedings of Robotics: Science and Systems, Rome, Italy, July 2015.
[44] Y. Xiang, W. Choi, Y. Lin, and S. Savarese. Data-driven 3d voxel patterns for object category recognition. In Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, 2015.
[45] M. Z. Zia, M. Stark, B. Schiele, and K. Schindler. Detailed 3d representations for object recognition and modeling. IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(11):2608–2623, 2013.
[46] M. Z. Zia, M. Stark, and K. Schindler. Are cars just 3d boxes? jointly estimating the 3d shape of multiple objects. In 2014 IEEE Conference on Computer Vision and Pattern Recognition, pages 3678–3685, June 2014.