

3D Object Detection(LiDar-based)

LiDar Point Cloud 的特点是：

1. 分布在三维空间中，且具有 稀疏性、无序性、无纹理 等特点，因此无法直接使用传统的卷积进行特征提取和检测，通过何种方式组织点云使其可被深度算法提取特征是一个重要课题（如 point-based method, voxel-base method, multiview-based）。
2. 点云可以提供 目标物的准确位置信息，同时 不受光照条件影响，无论从提供 free space 或者 目标检测 角度，都是自动驾驶车辆目标检测重要的补充。
3. 不同品牌激光雷达得到的点云，不同距离的点云（注 1）分布都不同，因此对于算法而言需要更高的 鲁棒性 。
4. Camera 与 Lidar 数据是非常不同 异源数据，如何将数据融合在一起也是一个重要课题。

在算法研究中需要注意的问题：

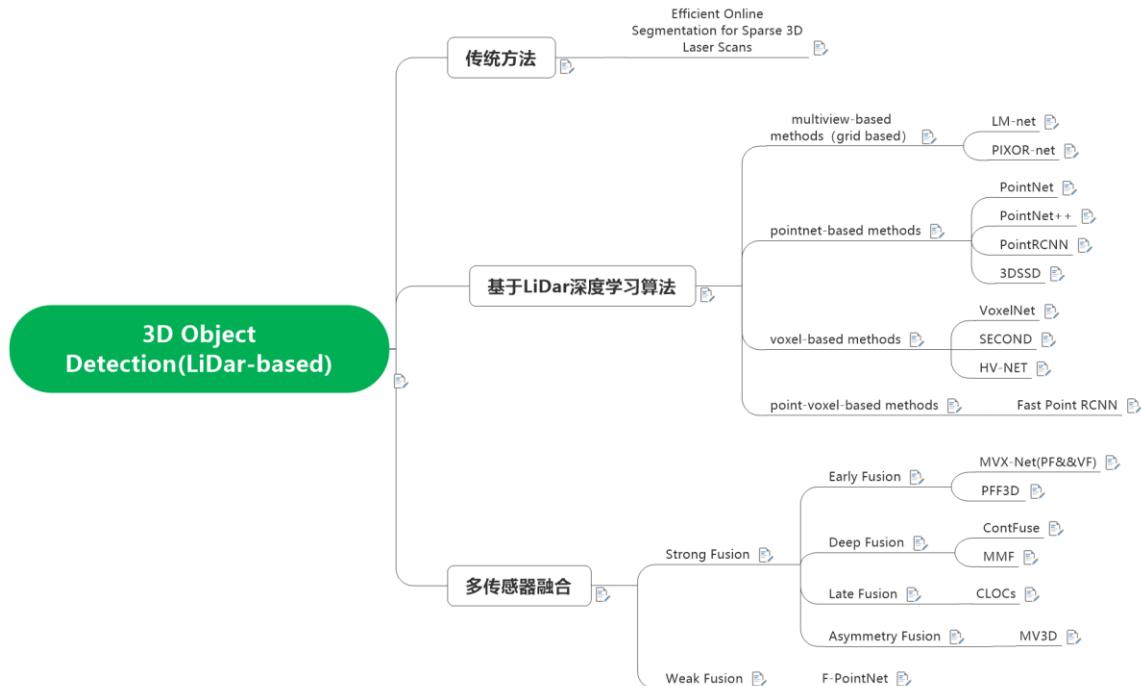
1. 算法是通过何种方式组织点云从而使神经网络可提取特征的。如 point-base method、voxel-based method, multiview-based 的方法。
2. 各种方法的发展脉络如何，该类算法的有缺点如何，该方法相对此前方法有何改进，又可能会存在何种问题。

工程中需要注意的问题：

1. 该方法对于 各种目标（大车、小车、行人、骑行）的目标检测准确率\角度预测准确率 分别是多少。很多论文中只选择一种目标如 car 作为 baseline，但是工程实践中，需要考虑所有目标的性能。
2. 需要考虑算法的实时性。一般地说，车载处理器的性能比桌面端要差很多，过于复杂的算法无法保证实时性。因此必须在检测准确性和算法复杂度上找到一个平衡。
3. 需要知道算法的性能瓶颈，如知道算法在不同距离下的准确性能，进行后续的工程上的处理。

注 1：

一般来说，在点云空间中，近距离目标包含的点云会比远距离目标多很多，因此虽然都是同种类的目标但是数据分布是完全不同的。虽然目前还没有在讨论这个问题，但是在具体的工程实践中是一个非常值得注意的问题。



1.	传统方法	3
1.1.	Efficient Online Segmentation for Sparse 3D Laser Scans	3
2.	基于 LiDar 深度学习算法	6
2.1.	multiview-based methods (grid based)	7
2.1.1.	LM-net.....	8
2.1.2.	PIXOR-net.....	9
2.2.	pointnet-based methods	10
2.2.1.	PointNet.....	11
2.2.2.	PointNet++.....	12
2.2.3.	PointRCNN.....	14
2.2.4.	3DSSD.....	15
2.3.	voxel-based methods	18
2.3.1.	VoxelNet.....	19
2.3.2.	SECOND.....	21
2.3.3.	HV-NET.....	22
2.4.	point-voxel-based methods	24
2.4.1.	Fast Point RCNN.....	25
3.	多传感器融合	27
3.1.	Strong Fusion	28
3.1.1.	Early Fusion.....	30
	• MVX-Net (PF&&VF)	31
	• PFF3D.....	32
3.1.2.	Deep Fusion.....	32
	• ContFuse.....	33
	• MMF.....	34
3.1.3.	Late Fusion.....	36
	• CLOCs.....	36
3.1.4.	Asymmetry Fusion.....	39

• MV3D.....	39
3.2. Weak Fusion.....	40
3.2.1. F-PointNet.....	40

1. 传统方法

Apollo 2.0

基于传统方法的 LiDar 目标检测，基本流程如下：

1. 对点云数据进行预处理，如降采样等。
2. 选择感兴趣区域。如通过道路检测选择道路区域为感兴趣区域，通过高精地图进行感兴趣区域的分割。
3. 对感兴趣区域的点进行聚类处理，并手工提取点簇的特征，如高度、密度、高程差等信息。
4. 使用传统机器学习方法对聚类后的点簇进行分类或者边框回归。（SVM，随机森林）。

总结与思考

优势：

1. 计算速度非常快，消耗的计算资源很少。
2. 如果只是说要提供一个 freespace，传统方法不失为一个不错的选择（仅仅通过聚类方法进行可行驶区域的提取）。

劣势：

1. 分类的准确性不高，边框回归的结构不准确。面对点云具有随机性、无序性，传统的聚类和分类方法没有良好的鲁棒性。（**数据分布广泛，而模型过于简单**）
2. 算法流程比较长，因此依赖也很多（如感兴趣区域的提取的准确性），每一步都会对算法的最终结果形成扰动。

思考

传统方法存在很大的短板，无法应用在对安全性能很高的三维目标检测中。但假如在技术方案中中需要激光雷达进行可行驶区域的提取，也是一个可行方案。

1.1. Efficient Online Segmentation for Sparse 3D Laser Scans

Efficient Online Segmentation for Sparse 3D Laser Scans

采用了提供了一种能够快速对点云进行分割的传统方法，在移动端处理器上 64 线激光雷达的分割效果可达 70hz，基本步骤是： 1. 基于点云生成的 range image 进行地面去除。

2. 基于 range image 进行快速点云分割。



Fig. 1: Left: Segmentation of objects such as people, cars, and trees generated from sparse 3D range data recorded with Velodyne VLP-16 scanner. Colours correspond to different segments.
Right: Clearpath Husky robot used for the experiments.

基于 range image 的地面去除

由于 RANSAC-based plane fitting 的地面去除方法，消耗的计算时间比较长，因此作者采用了一种基于几何角度的点云分割方法。

该方法的前提是三个假设：假设传感器是大致安装在水平基座上的；假设地面的曲率比较低；在点云生成的 range image 上，至少在最下面几行的像素上能够观测到地平面。

1. 首先将点云投射到 range image 上。
2. Savitsky–Golay 滤波算法对 range image 每一个列进行处理，去除异常点，平滑角度曲线。
3. 首先选择 range image 最下方一行的 pixel 作为基准 pixel，对 pixel 的 N4 邻域求夹角，若夹角 alpha 小于 5 度，则标记为地面。alpha 角如下图右上所示。

角度的计算公式如下：

$$\begin{aligned}\alpha &= \text{atan}2(\|BC\|, \|AC\|) = \text{atan}2(\Delta z, \Delta x) \\ \Delta z &= |R_{r-1,c} \sin \xi_a - R_{r,c} \sin \xi_b| \\ \Delta x &= |R_{r-1,c} \cos \xi_a - R_{r,c} \cos \xi_b|\end{aligned}$$

ξ 表示 LiDar 的线束垂直角，R 表示深度值，r 表示那一行。

伪代码实现如下：

Algorithm 1 Ground Labelling

```

1: procedure LABELGROUND( $R$ )
2:    $M \leftarrow [\alpha_{r-1,c}^r]$ , matrix of angles  $\alpha$  computed with Eq. (1).
3:   for  $c = 1 \dots R_{cols}$  do
4:     if  $M(0, c)$  not labelled then
5:       LabelGroundBFS( $0, c$ );
6: procedure LABELGROUNDBFS( $r, c$ )
7:   queue.push( $\{r, c\}$ )
8:   while queue is not empty do
9:      $\{r, c\} \leftarrow \text{queue.pop}()$ 
10:     $\{r, c\} \leftarrow \text{labelled as ground}$ 
11:    for  $\{r_n, c_n\} \in \text{neighbourhood}\{r, c\}$  do
12:      if  $|M(r, c) - M(r_n, c_n)| < 5^\circ$  then
13:        queue.push( $\{r_n, c_n\}$ )
14:    queue.pop()
```

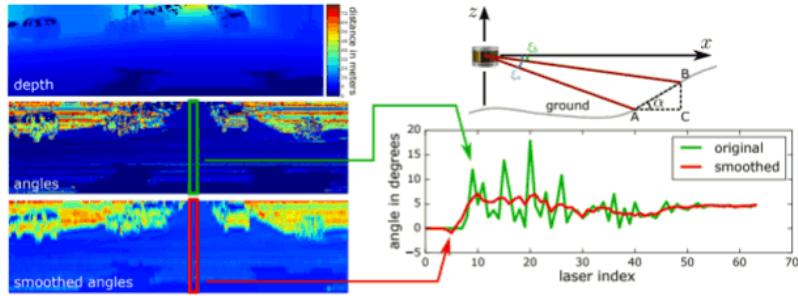


Fig. 2: Top left: part of a range image. Middle left: an image generated by showing α angles. Bottom left: angles after applying the Savitsky-Golay smoothing. Top right: an illustration of α angle. Bottom right: illustration of the smoothing for a column of α angles as marked in the left image.

基于 range image 的高效实例分割

作者同样利用 range image 上的几何角度关系进行点云的实例分割。

基本步骤如下：

1. 选取已经去除地面的点云。
2. 对所有非地面 pixel，进行聚类分割。

点 0 为 LiDar 位置，通过 N4 邻域进行搜索，OA\OB 为 LiDar 到点云的连线，计算夹角 beta，如果 beta 大于一个阈值则分类为一个目标，如果小于一个阈值则分为不同类。

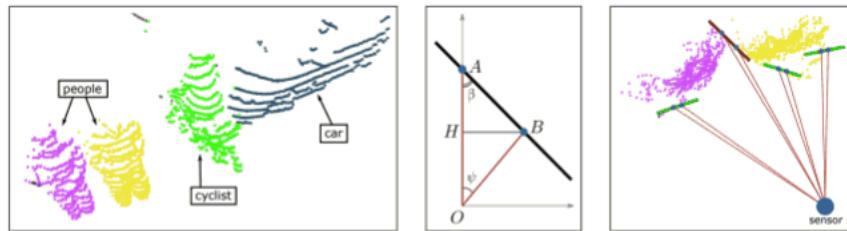


Fig. 5: Left: example scene with two pedestrians, a cyclist and a car. Middle: Given that the sensor is in O and the lines OA and OB represent two laser beams, the points A and B spawn a line that estimates the surface of an object should they both belong to the same object. We make the decision about this fact based on the angle β . If $\beta > \theta$, where θ is a predefined threshold, we consider the points to represent one object. Right: a top view on the pedestrians from the example scene. The green lines represent points with $\beta > \theta$ while the red one shows an angle that falls under the threshold and thus labels objects as different.

beta 的计算公式

$$\beta = \text{atan}2(\|BH\|, \|HA\|) = \text{atan}2(d_2 \sin \psi, d_1 - d_2 \cos \psi),$$

伪代码，通过 BFS 进行领域搜索

Algorithm 2 Range Image Labelling

```

1: procedure LABELRANGEIMAGE( $R$ )
2:   Label  $\leftarrow 1, L \leftarrow zeros(R_{rows} \times R_{cols})$ 
3:   for  $r = 1 \dots R_{rows}$  do
4:     for  $c = 1 \dots R_{cols}$  do
5:       if  $L(r, c) = 0$  then
6:         LabelComponentBFS( $r, c, Label$ );
7:         Label  $\leftarrow Label + 1$ ;
8: procedure LABELCOMPONENTBFS( $r, c, Label$ )
9:   queue.push( $\{r, c\}$ )
10:  while queue is not empty do
11:     $\{r, c\} \leftarrow queue.top()$ 
12:     $L(r, c) \leftarrow Label$ 
13:    for  $\{r_n, c_n\} \in Neighbourhood\{r, c\}$  do
14:       $d_1 \leftarrow max(R(r, c), R(r_n, c_n))$ 
15:       $d_2 \leftarrow min(R(r, c), R(r_n, c_n))$ 
16:      if  $atan2\frac{d_2 \sin \psi}{d_1 - d_2 \cos \psi} > \theta$  then
17:        queue.push( $\{r_n, c_n\}$ )
18:   queue.pop()

```

分割结果如下：

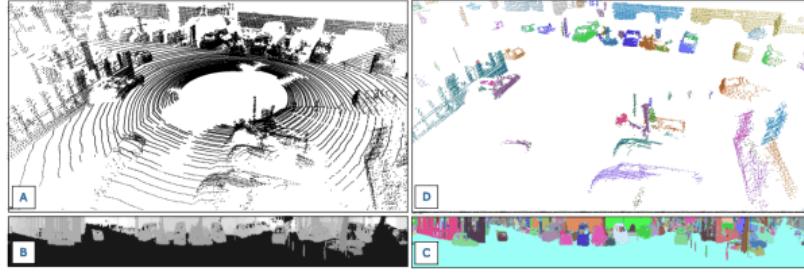


Fig. 4: Illustration of our method. (A) Point cloud from Velodyne, which is shown for illustration reasons only. (B) We build up a range image not considering points lying on the ground plane and (C) perform the segmentation in the range image directly. (D) This allows us to provide individual small point clouds for the different segments. The different objects are shown with random colours. Range and label images are scaled for better visibility.

总结与思考：

1. 该算法将点云投射到 range image 上，然后通过几何角度对点云进行地面去除，和实例分割。
2. 只用于点云的实例分割，并不进行点云的分类和检测。
3. 运行速度非常快。

2. 基于 LiDar 深度学习算法

该小节将介绍基于深度学习 LiDar 目标检测算法。

根据对点云数据的组织方式不同，可以分为四类深度学习方法，分别是 multiview-based method\voxel-based method\pointnet-based method\point-voxel-based method。

1. multiview-based method

该类方法是指通过 手工提取特征 将点云编码成前视图或者俯视图，然后通过 二维卷积 进行特征提取及目标检测的方法。该类方法 具有检测速度快，但是过程中 丢失大量空间信息，存在 准确率瓶颈。

2. voxel-based method

该类方法是指将 点云体素化，再通过 VEF 初步提取特征，然后出入 3d 卷积进行特征提取及目标检测的算法。该类算法保留了很多点云的空间信息，改进版的算法也具有良好的检测性能和实时性。

3. pointnet-based method

该类方法是指直接使用 pointnet 算法直接提取点云的特征和目标检测的算法。在该类算法可以达到一个不错的检测准确率，但是由于需要对大量目标点进行

4. pointnet-voxel-based method

该类方法是指 使用 voxel-based 进行 region proposal ，然后使用 point-net 对该区域的点云进行目标判断和边框 的回归。

优缺点：

每类方法都有自己的优缺点，子树中将一一论述。

2. 1. multiview-based methods (grid based)

multiview-based method

该类方法是指通过 手工提取特征 将点云编码成前视图或者俯视图，然后通过 二维卷积 进行特征提取及目标检测的方法。

基本流程：

1. 投影。将点云投射到 brideye-view 或者 front-view。
2. 手工构建特征图。网格化并对网格中的点云 手工提取特征 ，如手工提取最大高度、点云密度和高程差等信息。
3. 深度学习。使用 二维卷积 对预处理之后的数据进行特征提取和目标检测。

优点：

1. 计算资源消耗少，检测速度快。 预处理部分使用 cuda 加速，时间控制在 1ms 以内；深度学习部分使用 2d 卷积进行处理，相对其他方法是非常高效的方法，具体处理时间和硬件条件\感知区域大小\网络层数相关。

缺点：

1. 丢失空间信息。 通过手工提取特征的方法会丢失大量空间信息，以至于在目标检测准确率和边框回归准确率上存在比较大的挑战，且在边框回归任务上挑战更加大。因为缺少点云的空间信息，边框的预测常常会出现预测角度不准和边框大小预测困难的案例，影响自动驾驶的安全性。
2. 对于行人和骑行两类而言，检测准确性存在挑战。 可能存在的原因如下：在现实中，行人在点云中的体积一般是 $0.5 \times 0.5 \times 1.7$ (米)，骑行在点云中的体积一般是 $0.8 \times 2 \times 1.8$ (m)，且在仅被打到一个面的情况下，体积更小。因此在网格化和手工特征提取时，能够表达的信息较少，因此准确性存在很大挑战。

代表算法：

brideye-view based:

PIXOR-net, YOLO3D

front-view based:
LM-net

2.1.1. LM-net

front-view 这类方法的检测性能并不是很高，一笔带过吧。

LM-net 的算法流程也很简单，如下：

1. 将点云投射到前向视角，得到一个类似图像的点云前视图。
 2. 手工提取特征得到特征图，然后通过二维卷积进行目标检测。
- 前向特征图编码方式：

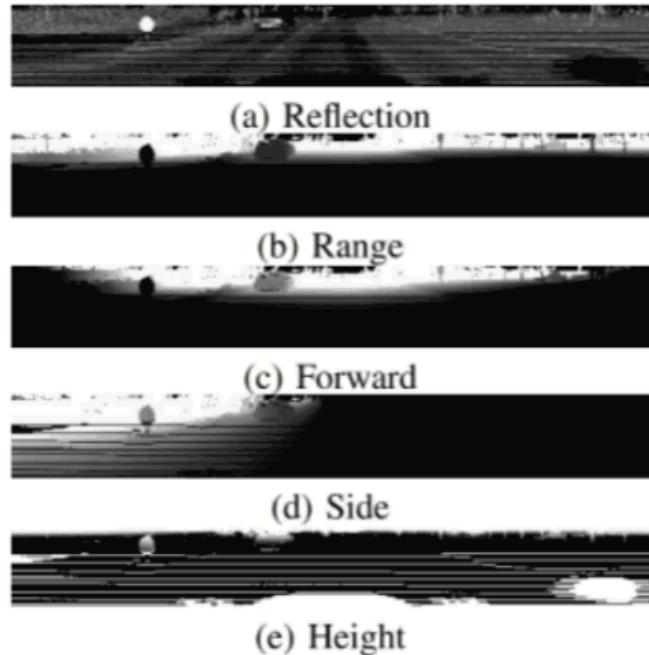


Fig. 2. Encoded input point cloud

算法流程图：



Fig. 1. LMNet architecture

总结与思考

优点：

1. 引入了空洞卷积，检测速度非常快。

缺点：

1. 将点云投射到前向视角丢失了大量三维信息，且有遮挡问题，精度较低，着实不是一个很好的解决方案。

2.1.2. PIXOR-net

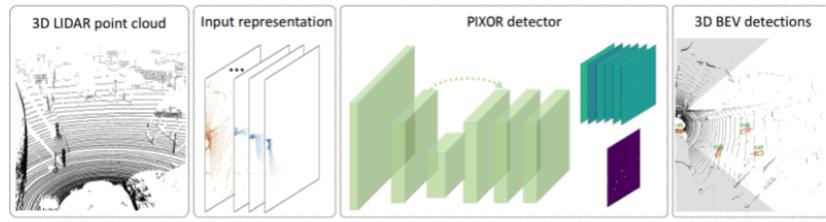


Figure 1. Overview of the proposed 3D object detector from Bird's Eye View (BEV) of LIDAR point cloud.

图1 算法流程

算法流程比较好理解，如下：

1. 网格化。在 bird eye view 视角下，以 $dL \times dW \times dH$ 的分辨率，将点云网格化。
2. 手工特征图构建。如果网格中包含点云，则将该位置的数值置为 1（占有率）。
3. 特征提取及目标检测。使用 2d 卷积框架提取特征，并检测出目标。

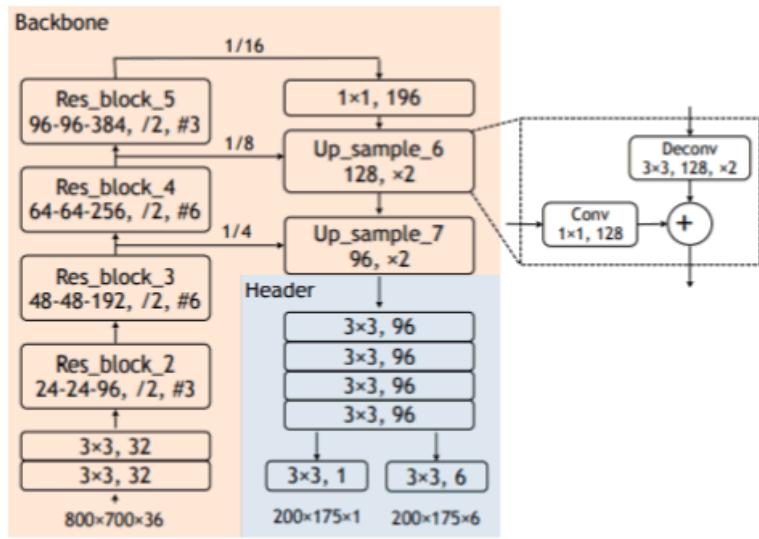


Figure 2. The network architecture of PIXOR.

图2 网络结构

边框回归：

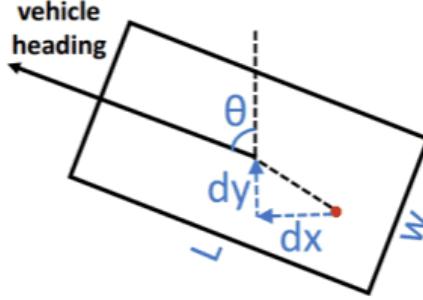


Figure 3. The geometry output parameterization for one positive sample (the red pixel). The learning target is $\{\cos(\theta), \sin(\theta), dx, dy, \log(w), \log(l)\}$, which is normalized before-hand over the training set to have zero mean and unit variance.

损失函数：

1. 由于背景点远远多余前景点，pixor 使用了 focal_loss 来平衡这个样本不均衡问题。
2. 为了能够较为平滑得进行边框回归的学习，pixor 为边框回归任务设计了 smooth loss。

$$Loss = \text{focal_loss}(p, y_{cls}) + \text{smooth}_{L_1}(q - y_{reg})$$

$$\text{focal_loss}(p, y) = \begin{cases} -\alpha(1-p)^\gamma \log(p) & \text{if } y = 1 \\ -(1-\alpha)p^\gamma \log(1-p) & \text{otherwise,} \end{cases}$$

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases}$$

总结与思考：

1. pixor 在当时是一个不错的尝试，具有不错的检测准确率（也曾一度是 kitti 的榜一，ap 达到 0.77），也有良好的检测速度（35fps）。
2. 但由于这类方法会丢失大量特征，无法解决小目标检测不准的问题，在航向角预测中也常常会出现明显的偏移。
3. pixor 对于损失函数的设计是相当不错的，我们今天仍然可以借鉴。

2. 2. pointnet-based methods

pointnet-based method 是基于 PointNet++ 和 PointNet 所提出的一种可直接在原始点云上提取特征的 LiDar 目标检测算法。

基本流程是：

1. Set Abstraction 提取原始点云特征，通过 SA 的叠加为算法提高感知范围。pointnet++ 包含的基本操作是 FPS（最远点采样 D-FPS, F-FPS）、grouping（不同层次结构的好几种）、pointnet（提取特征）。

2. region proposal。初步预测出目标的位置和大小。（3DSSD 算法省去了这一步）
3. refined。对初步提取候选框，进行精炼操作。预测出目标种类，修正边框大小。

总结与思考

优点：

1. 直接在点云数据中提取特征，保留更多的点云原始信息。有助于提供边框回归的效果。

缺点：

1. 时间效率不高。PointNet++本身的时间计算效率不高，其原因是需要对很多个点簇进行特征提取操作。
2. 非规则访问，访问耗时比较长。点云是并不是规则存储的，包括 grouping 和 sampling 都是对点云的非规则访问。为了避免重复的非规则访问，算法会对数据进行多份的拷贝，消耗大量存储空间。

2.2.1. PointNet

PointNet 和 PointNet++是最早用于点云的分割和对完整点云的分类，但却是 pointnet-based 检测的基石。想要读懂 pointnet-based 方法，首先需要理解 PointNet 和 PointNet++。

PointNet：

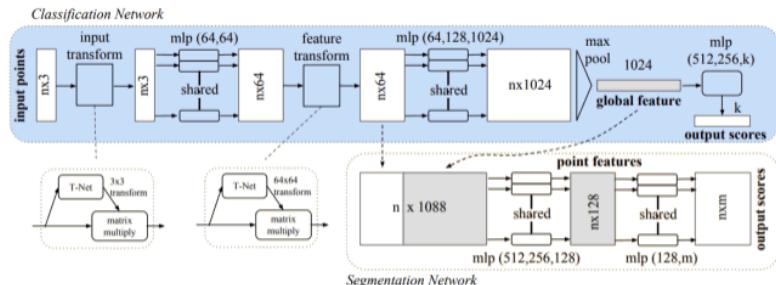


Figure 2. **PointNet Architecture.** The classification network takes n points as input, applies input and feature transformations, and then aggregates point features by max pooling. The output is classification scores for k classes. The segmentation network is an extension to the classification net. It concatenates global and local features and outputs per point scores. “mlp” stands for multi-layer perceptron, numbers in bracket are layer sizes. Batchnorm is used for all layers with ReLU. Dropout layers are used for the last mlp in classification net.

算法流程如图：

- 1、特征提取。将一帧点云中的数据全部输入算法；输入数据先通过 T-Net 来对齐（注 1），保证了模型的对特定空间转换的不变性；然后通过权重共享的 mlp 对各点云数据进行特征提取，之后再使用用一个 T-Net 对特征进行对齐。
- 2、获取全局特征。在特征的各个维度上进行 maxpooling 操作来得到最终的全局特征。
- 3、输出结果。对分类任务，将全局特征通过 mlp 来预测最后的分类分数；对分割任务，将全局特征和之前学习到的各点云的局部特征进行 concatenate 操作，再通过 mlp 得到每个数据点的分类结果。

优点：

1. 开创性。PointNet 在点云处理领域是一篇具有开创性地文章。首先提出了一种可直接作用于点云的特征提取器。

缺点:

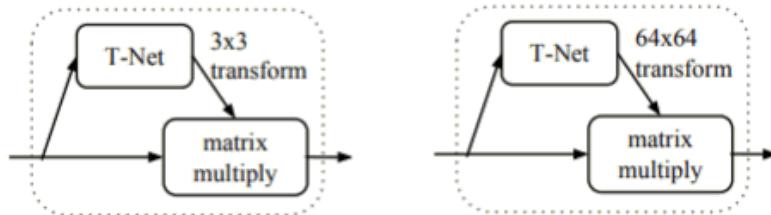
- 无法提取局部特征信息。在算法中，缺乏点与点之间的交互模块，因此无法提取局部信息，因此仅适用简单场景。

总结:

PointNet 在点云处理领域是一篇具有开创性地文章。首先提出了一种可直接作用于点云的特征提取器。但也存在缺陷，除了 maxpooling 这一步操作，点与点几乎是孤立的，因此算法缺乏提取局部特征的能力，无法获得良好的泛化能力有限。

注:

- T-NET 是为了保证点云数据不变性的一个小型网络。点云数据所代表的目标对某些空间转换应该具有不变性，如旋转和平移等刚体变换。为了保证点云的空间不变性，作者需要对点云进行对其操作。通过训练一个小型的网络来得到一个变换矩阵，将变换矩阵和输入点云相乘以实现点云的对齐。



PointNet 中使用了两个 T-NET, 第一个 T-NET 输出一个 3×3 的矩阵，第二个 T-NET 输出一个 64×64 的矩阵。作者发现第二个 T-NET 在训练中比较难优化，但如果输出的 64×64 矩阵是一个正交矩阵，那么优化就方便地多。因此引入了一个额外地损失函数。

$$L_{reg} = \|I - AA^T\|_F^2, \quad (2)$$

2.2.2. PointNet++

PointNet++是对 PointNet 的重要改进，解决了 PointNet 无法提取出局部特征的问题。

PointNet++也是一篇具有开创性意义的文章，通过引入 Set Abstraction 来提供局部信息感知的能力。

Set Abstraction:

Set Abstraction 由 Sampling layer\Grouping layer\Pointnet 构成，多个 SA 层叠加可达到类似 2 维卷积的层级特征提取效果。其中 Sample Layer 和 Grouping Layer 是作者开创性提出的，PointNet 是延用前一篇论文 PointNet。

Sampling layer 解决的是点云过多带来地冗余问题。一帧点云往往具有非常多的点，对每个点都做局部特征提取，将会造成巨大的冗余，消耗过多的计算资源。Sample layer 通过对有代表性的点云进行采样，减少冗余计算。PointNet++的采样算法是最远点采样(farthest point sampling algorithm, FPS)。算法流程如下(参考 [最远点采样介绍及 CUDA 实现分析](#))

[cbw052 - 博客园 \(cnblogs.com\)](#) , 时间复杂度约为 $K \times N$, K 为需要采样的点数, N 为点云包含点的总数:

1. FPS逻辑描述

假设有个点, 要从中按FPS算法, 采样出 k 个点 ($k \leq n$)。那么, 可以将所有点归类到两个集合 A, B 中, A 集合表示选中的点形成的集合, B 集合表示未选中的点构成的集合。FPS所做的事情就是每次从集合 B 中选取一个到集合 A 里的点距离最大的点。

初始情况: 集合 A 为空, 集合 B 包含所有点

选第一个点: 对所有点进行shuffle后, 选取第一个点, 将其加入集合 A 中。此时 $\text{size}(A) = 1, \text{size}(B) = n - 1$

选第二个点: 分别计算出集合 B 里面每个点到集合 A 中一个点的距离, 选取距离最大的点, 加入集合 A 。此时, $\text{size}(A) = 2, \text{size}(B) = n - 1$,

选第三个点及后续的点: 设此时集合 A 中有 m 个点 ($m \geq 2$), 集合 B 中有 $n - m$ 个点。此时需要定义集合 B 中的点 p_B 到集合 A 中点的距离, 注意到集合 A 中不止一个点, 这是FPS的核心。对 p_B 到集合 A 的距离 d_B 定义如下

$$d_B = d(p_B, A) = \min(d(p_B, p_A^j)) \quad (p_B \in B, p_A^j \in A)$$

1. 分别计算出 p_B 到集合 A 中每个点的距离, 当集合 A 中有 m 个点时, 可以计算出 m 个距离值

2. 从计算出来的 k 个距离值中, 取最小的距离值, 作为点 p_B 到集合 A 的距离

对于集合 B 中的 $n - m$ 个点, 每个点都可以计算出一个距离值: $\{d_B^1, d_B^2, \dots, d_B^{n-m}\}$ 。选出最大距离值对应的点, 记为第 $m + 1$ 个点, 将其加入集合 A , 此时集合 A 包含 $m + 1$ 个点, 集合 B 包含 $n - (m + 1)$ 个点。

重复上述步骤3, 直至选出 k 个点为止。

Grouping layer 解决的是选取哪些点作为已采样点的领域点。PointNet++ 采用的是 Ball Query 的方法, 即以 FPS 得到的每个点作为球心, 以 R 为半径, 在球中随机采样 C 个点。如果球内不足 C 个点, 则以 0 补全; 如果球内超过 C 个点, 则随机剔除其余点。考虑到点云的分布是不均匀的, 在密集区域学习到的特征是不适用于稀疏区域的。因此作者在 Ball Query 的基础上提出了 MSG 和 MRG 两种采样方式。

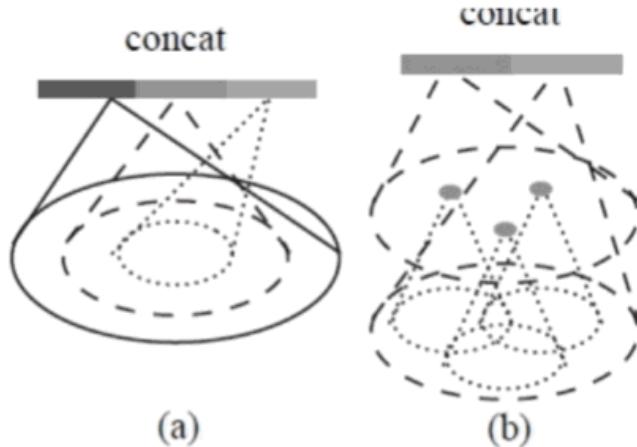


Figure 3: (a) Multi-scale grouping (MSG); (b) Multi-resolution grouping (MRG).

Segmentation: 在分割任务中, 作者采用反向插值和 skip-connection 的方法。反向插值是用较少的点把更多的点的特征插出来, 实现上采样。

skip-connection 类似于 u-net 的一种网络结构, 将上层特征与插值后的特征进行链接。

算法的流程如下:

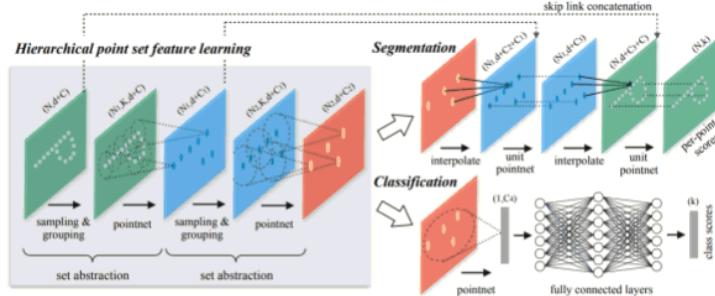


Figure 2: Illustration of our hierarchical feature learning architecture and its application for set segmentation and classification using points in 2D Euclidean space as an example. Single scale point grouping is visualized here. For details on density adaptive grouping, see Fig. 3

总结与思考:

PointNet++是PointNet的重要补充，提供了局部感知的能力。本质上PointNet++是通过SA的方法将一张大点云分成很多个小点云簇，然后通过PointNet进行分割\预测\分类。

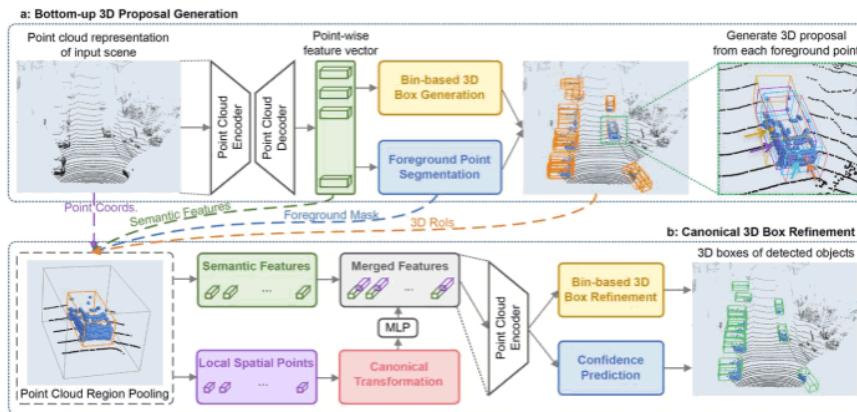
2.2.3. PointRCNN

PointRCNN是基于PointNet++提出的LiDar目标检测算法，首次将PointNet++应用到点云目标检测领域。

PointRCNN也可以看成Faster-RCNN在LiDar目标检测中的应用，和Faster-RCNN一样PointRCNN也是一种two stage的目标检测算法。

一阶段：region proposal。通过PointNet++进行前景与背景的分割，对于每个前景点都会预测一个3d Box信息(bin-based 3d bbox，置信度信息+朝向信息，共7维度)。然后进行NMS去冗余操作，最终保留Top T个proposal

二阶段：refined。对于保留的每个proposal提出的3d bbox(略微增大一部分尺度)中的点云进行采样；而这些筛选的点云映射到局部坐标系，提取局部特征(SA)；将局部特征和一阶段的全局特征进行融合；最后输出精细定位的检测结构。



思考与总结:

优点：

- 首次将 PointNet++ 应用到 LiDar 目标检测任务中，可直接对点云进行特征提取，可保留大量的原始三维空间信息。
 - 对于边框的预测更加准确。由于保留了大量点云的空间信息，且采用了 Bin-Based Box Generation 方法（将边框回归任务分解成分类任务和回归任务），边框预测更加准确。
- 缺点：
- 最大的缺点便是计算效率不高。原因主要有两个：two stage 的检测算法，这本身就是一种牺牲效率，提高精度的方法；PointNet++本身就比较消耗计算资源。

2.2.4. 3DSSD

3DSSD 也是基于 PointNet++ 的 LiDar 目标检测算法，但相比于 PointNet++，拥有更高的计算效率。主要原因有以下两点：1). 优化了最远点采样，引入了基于特征的最远点采样 F-FPS。2) 采用 one-stage 的 header，减少大量计算消耗。但也采用 anchor free 的方法提出更多的 proposal 以提高感知的准确性。

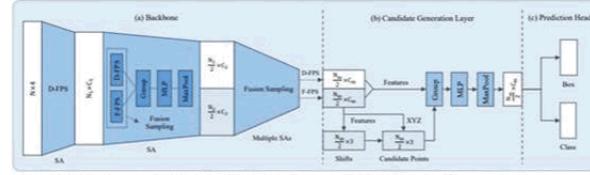


Figure 1. Illustration of the 3DSSD framework. On the whole, it is composed of backbone and box prediction network including a candidate generation layer and an anchor-free prediction head. (a) Backbone network. It takes the raw point cloud (x, y, z, r) as input, and generates global features for all representative points through several SA layers with fusion sampling (FS) strategy. (b) Candidate generation layer (CG). It downsamples, shifts and extracts features for representative points after SA layers. (c) Anchor-free prediction head.

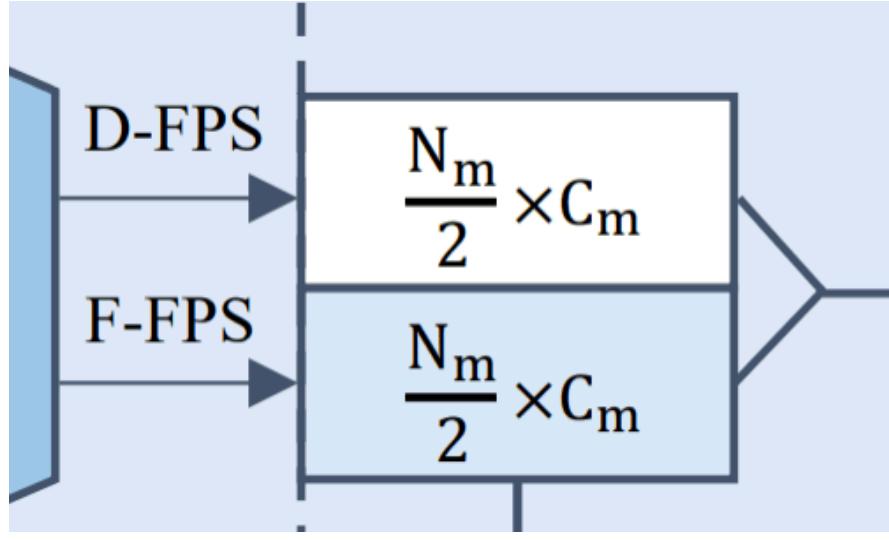
Feature-FPS

更高的采样效率。原始 PointNet++ 中 Distance-FPS 是基于欧氏距离进行采样，目标是采样点尽可能分布在整个三维空间中，因此会采样到大量的背景点（背景点分步比前景点多很多），采样效率不高（点占有率）。于是作者提出了能够尽可能采样到更多前景点的采样方法 F-FPS，该方法兼顾特征距离和欧式距离（只考虑特征距离，容易出现采样点都是同一个目标的情况）：

$$C(A, B) = \lambda L_d(A, B) + L_f(A, B), \quad (1)$$

F-FPS+D-FPS

只通过 F-FPS 进行采样，导致选出来的点很多都是 Positive 的，而 Negative 的点很少。这就引起 样本不均衡的问题，最终导致分类效果不佳。故作者提出了一种融合采样策略，通过 F-FPS 采样一半特征点，通过 D-FPS 采样一部分特征点。



Candidate Generation Layer

作者通过 F-FPS 选取 candidate points(经过偏移的), 再对 candidate points 附近的点进行 grouping, 输入 MLP, 最终进行目标检测。

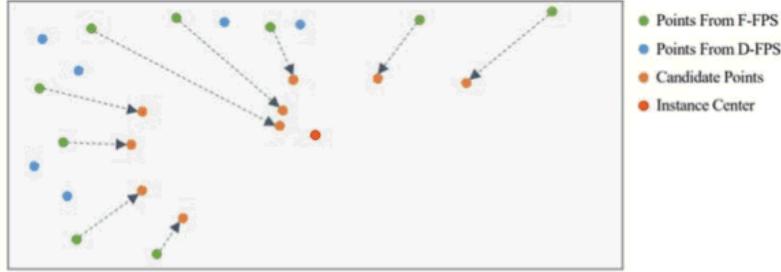


Figure 2. Illustration of the shifting operation in CG layer. The gray rectangle represents an instance with all positive representative points from F-FPS (green) and D-FPS (blue). The red dot represents instance center. We only shift points from F-FPS under the supervision of their distances to the center of an instance.

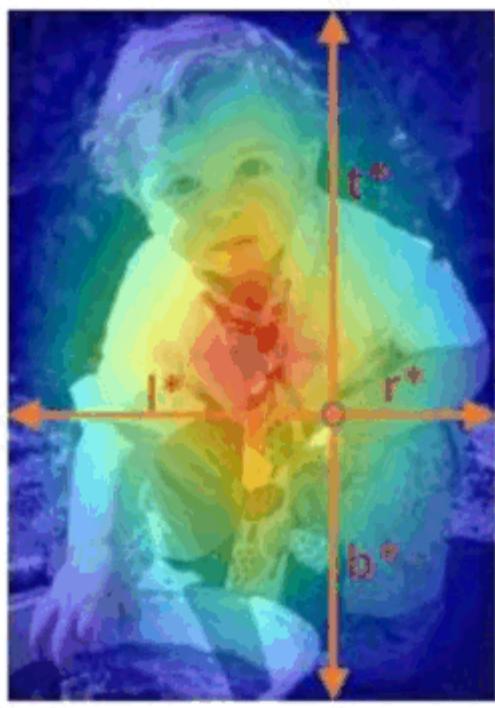
Anchor-free Head

通过引入 F-FPS+D-FPS 的融合采样与 CG Layer, 可安全地移除耗时的 FP 层和精炼层。为了更好地进行候选框位置回归, 作者采用了更加先进的 anchor-free 方法。

对于每个候选点, 需要回归的目标是:

1. 每个候选点到其对应的实例 Gt bbox 中心点的距离
2. 每个候选点对应的实例 Gt bbox 的尺度信息
3. 每个候选点对应的实例 Gt bbox 的转角信息, 引入了 bin+res 的形式进行回归。

Center-ness



为了提升检测框的质量，作者借鉴了 Anchor-free 2D 检测模型 FCOS 中的中心度来设置最终模型要学习的类别标签信息。中心度反应了某一个点到实例 gt bbox 中心的（标准化）距离，如上图所示，越靠近实例中心，该中心度越大（热力图越红），越远离小（热力图越蓝）。

$$l_{ctrness} = \sqrt[3]{\frac{\min(f, b)}{\max(f, b)} \times \frac{\min(l, r)}{\max(l, r)} \times \frac{\min(t, d)}{\max(t, d)}}, \quad (2)$$

优点：

1. 效率更高。通过引入 F-FPS 与 Candidate Generation Layer 安全移除了 feature propagation 层和精炼层，相比于 PointRCNN 效率提升了将近一倍（从 13fps 到 25fps）。

缺点：

1. 仍然不适用于车载系统，车载处理器的性能比桌面处理差很多。3DSSD 仍然过于重了。

2.3. voxel-based methods

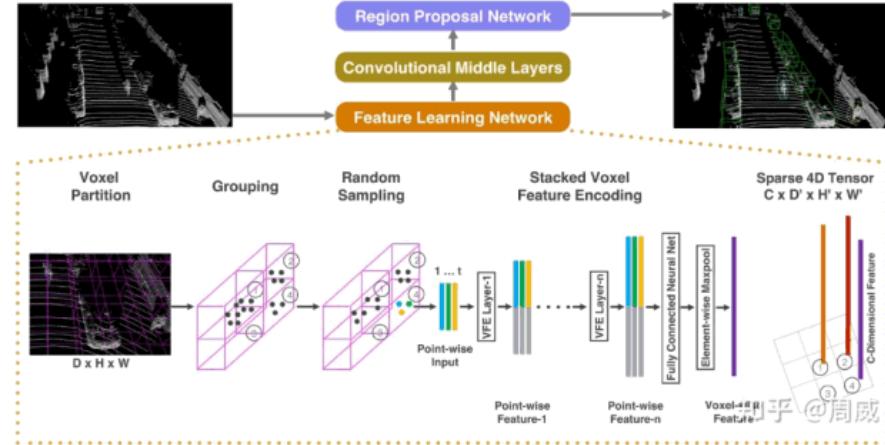


图1

图1 voxel-net

voxel-based method 指的是通过将点云体素化，然后通过 VEF、三维卷积进行特征提取及目标检测的方法。

基本流程：

1. 将点云体素化，根据体素将点云分组。
2. 对体素中的点云进行随机采样（选取 t 个点，如不足则补全），编码体素中的点云特征。

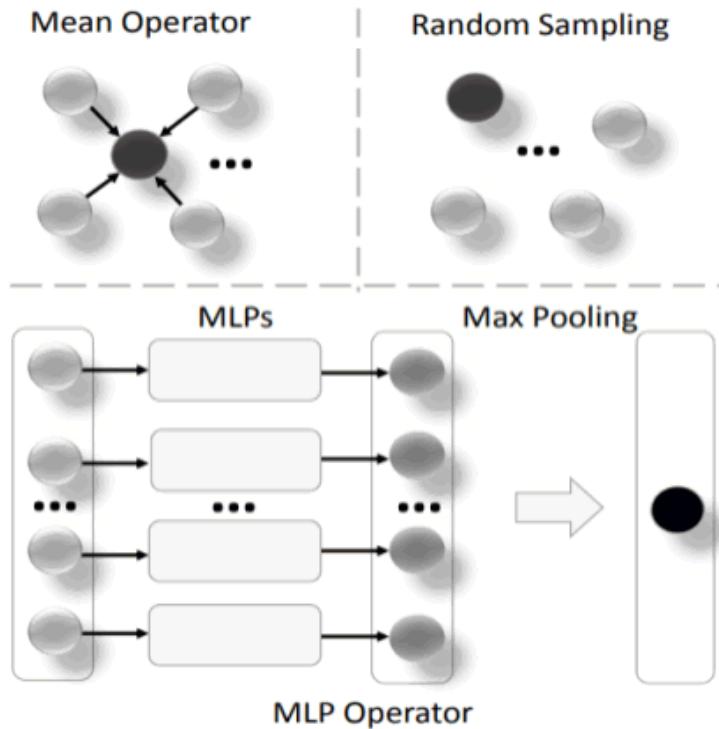


图2 点云特征编码方法 3. 通过 3d 卷积 进行特征提取。（一些算法优化了 3D 卷积，如 SECOND，一些算法巧妙地绕开了 3D CNN 如 HVNet, Pointpillar）

4. 使用 2d 卷积 检测出目标，并对目标进行边框回归。

优点：

1. 在特征提取过程中，可以利用到大量的点云空间信息。
2. 可以达到比较好的检测指标，在合理构建网络的情况下也可以得到不错的检测速度（如 pointpillar 可以达到 62hz，second 可以达到 30hz）。

缺点：

1. 一般的说，3d 卷积的计算 花销 是很大的。但是可以构建更加精简的网络去避开这个弊端（如尽快将三维特征数据加工成二维特征数据）。
2. 如果 体素的分辨率过大，边框预测的准确性会存在挑战； 体素的分辨率过小，会增加很多计算量，增加一次检测完成的时间。
3. 需要 gpu 内存容量相对较高，原因是算法构建了 三维的特征结构，对于每个 voxel 都会存储其中每个点云的信息（dynamic voxelization 可改善这个问题）。

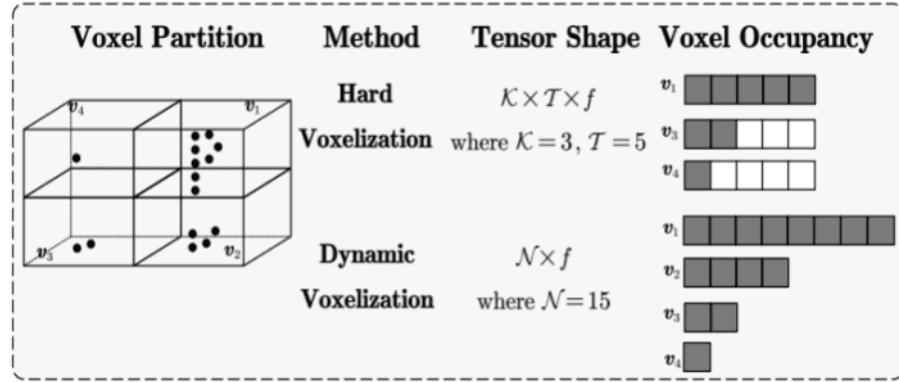


图 3 dynamic voxelization

4. 因为存在一些建立索引和建立反索引等操作，模型的压缩算法并不是很成熟，工程化时 模型压缩的难度较大。（我当时的调研是这样的，不知道现在的情况是否有改善）。

2.3.1. VoxelNet

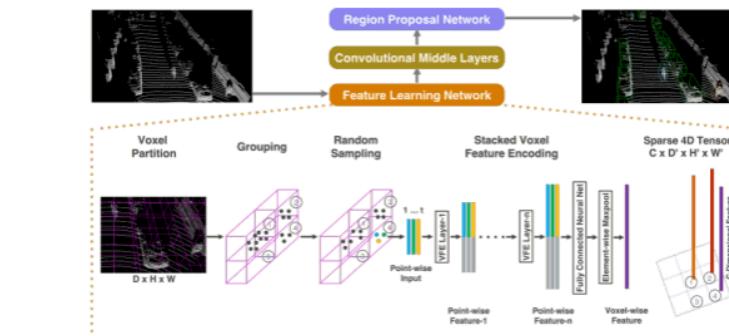


Figure 2. VoxelNet architecture. The feature learning network takes a raw point cloud as input, partitions the space into voxels, and transforms points within each voxel to a vector representation characterizing the shape information. The space is represented as a sparse 4D tensor. The convolutional middle layers process the 4D tensor to aggregate spatial context. Finally, a RPN generates the 3D detection.

VoxelNet 是将原始点云进行**体素化处理**，然后使用 2 维及 3 维卷积进行**特征提取**的一种目标检测方法。VoxelNet 的网络结构分为三部分，分别为**特征学习网络\中部卷积层\headnet**

特征学习网络

(1) Voxel Partition: 也就是将空间划分为一个个堆叠的、相同大小的 Voxel。

(2) Grouping: 上面将空间划分为一个个的 Voxel 了，Grouping 这一步的作用就是将 3D 点云数据**装进这一个个的 Voxel 中，实现分组**。

(3) Random Sampling: 3D 点云的数据量往往都是几万以上的。要是直接在这个数量级上进行特征提取，是非常消耗计算资源的。所以作者提出了随机采样方法，将点云数量超过 的 Voxel 中的**点云数量降至 T**。

(4) Stacked Voxel Feature Encoding (VFE 下面)：这一步是最重要的一步。作者在这一步提出了 VFE 层 (VFE= Voxel Feature Encoding)。

VFE 是将体素中去中心修正后的点输入权重共享的全连接神经网络中，然后利用 element-wised maxpooling 选择最具代表性的特征作为全局特征，将全局特征和输入特征进行拼接作为 VFE 的输出。

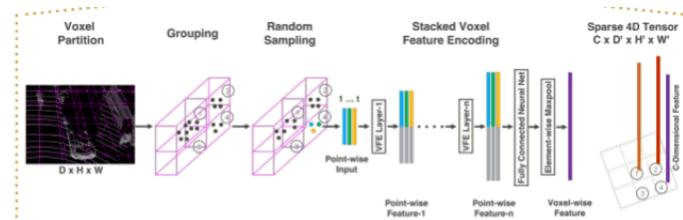


Figure 2. VoxelNet architecture. The feature learning network takes a raw point cloud as input, partitions the space into voxels, and transforms points within each voxel to a vector representation characterizing the shape information. The space is represented as a sparse 4D tensor. The convolutional middle layers processes the 4D tensor to aggregate spatial context. Finally, a RPN generates the 3D detection.

中间层卷积

特征学习网络输出的结果是一个 $W \times H \times D \times C$ 的 4D TENSOR，因此使用了一个 3D CNN 作为特征提取网络，经过多层的特征提取，最终输出一个 2D TENSOR。

3D CNN 相比于 2D CNN 参数量上了一个数量级，需要的存储空间也上了一个数量级。

headnet

headnet 与 one stage 的 2D 目标检测算法的结构非常相似，输入为 2D Tensor，输出为目标的边框、位置及类别。

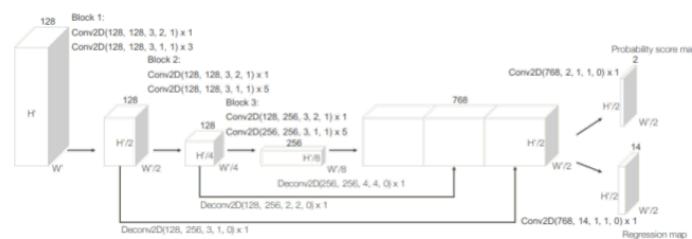


Figure 4. Region proposal network architecture.

总结与思考

优点:

1. 是 voxel-based 的开山鼻祖，具有开创性意义。
2. 该方法能够保留一部分的点云空间特征，是一种不错 LiDar 目标检测方案。

缺点：

1. 计算效率低，需占用的内存空间大。主要原因是使用了 3D CNN。后续也有很多方法改进了该缺点，如 pointpillars\HVNet 直接跳过了 3D CNN，SECOND 算法提出了 3 维稀疏卷积也有效改善计算效率和空间存储效率。

2.3.2. SECOND

考虑到 VoxelNet 通过 Feature Learning Network 后获得了稀疏的四维张量，而采用 3D 卷积直接对这四维的张量做卷积运算的话，消耗了大量计算资源。SECOND 用稀疏 3D 卷积替换了普通 3D 卷积，大幅提高了计算效率。如下图，SECOND 的算法流程与 VoxelNet 类似的，主要区别是使用稀疏 3D 卷积替代了普通 3D 卷积，从而大大得提高了计算效率。

3.1. NETWORK ARCHITECTURE

The proposed SECOND detector, depicted in Figure 1, consists of three components:
(1) a voxelwise feature extractor; (2) a sparse convolutional middle layer; and (3) an RPN.

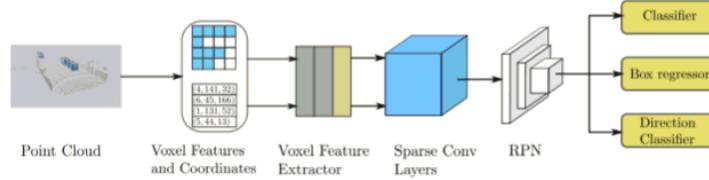


Figure 1. The structure of our proposed SECOND detector. The detector takes a raw point cloud as input, converts it to voxel features and coordinates, and applies two VFE (voxel feature encoding) layers and a linear layer. Then, a sparse CNN is applied. Finally, an RPN generates the detection.

3D 稀疏卷积（索引与反索引，减少对于空 voxel 的无效计算）：

如下图所示，可以归纳为：

1. 将稀疏的输入特征通过 `gather` 操作获得密集的 `gather` 特征；
2. 然后使用 GEMM 对密集的 `gather` 特征进行卷积操作，获得密集的输出特征；
3. 通过预先构建的输入-输出索引规则矩阵，将密集的输出特征映射到稀疏的输出特征。

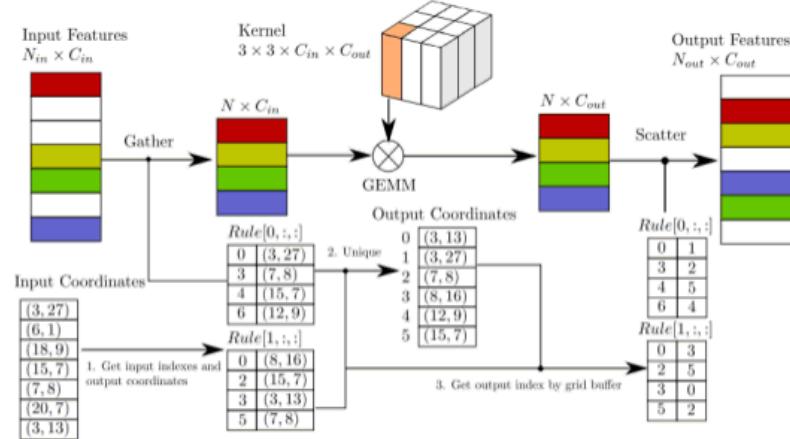
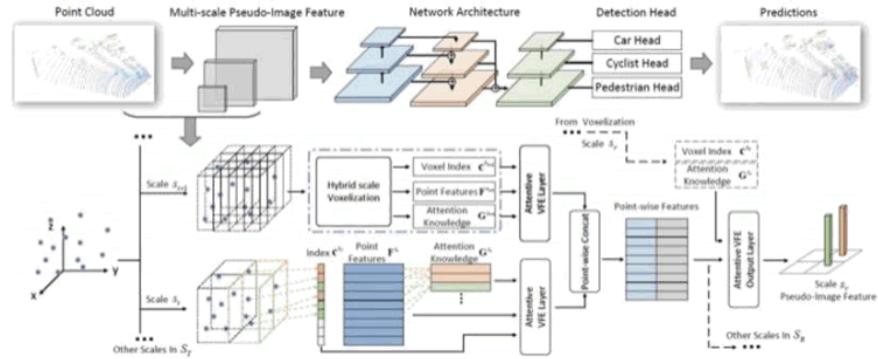


Figure 2. The sparse convolution algorithm is shown above, and the GPU rule generation algorithm is shown below. N_{in} denotes the number of input features, and N_{out} denotes the number of output features. N is the number of gathered features. $Rule$ is the rule matrix, where $Rule[i, :, :]$ is the i -th rule corresponding to the i -th kernel matrix in the convolution kernel. The boxes with colors except white indicate points with sparse data and the white boxes indicate empty points.

总结：

3D 稀疏卷积对于 VoxelNet 而言是一个重要优化，对于 Voxel-based method 方法，3d 卷积是非常消耗计算资源和内存的。而点云是非常稀疏的数据，有大量的计算资源消耗在对于空洞特征的计算上，稀疏 3D 卷积通过建立索引和反索引的方法，减少了这些冗余计算和无谓的内存消耗。

2.3.3. HV-NET



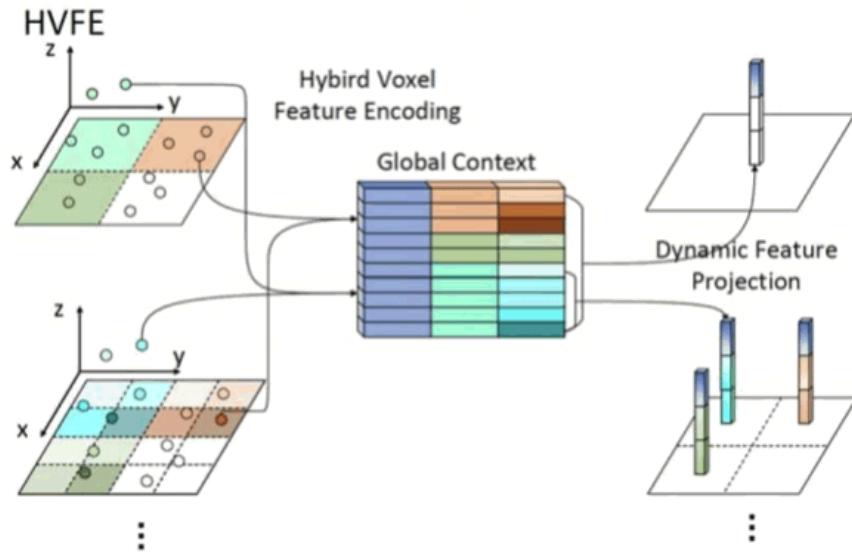
相比于 VoxelNet，HVNET 主要有以下几点不同：

1. 移除了消耗内存和计算资源最多的 3D 卷积网络，时间效率和存储效率都大大提高。
2. 通过多尺度的 VFE 特征提取，将不同细粒度的特征进行融合。
3. 分别在不同尺度的 feature 上，进行目标检测。在高分辨率的特征图上检测行人，在中分辨率的特征图上检测骑行目标，在粗分辨率的特征图上检测车辆。这是非常不错的尝试，行人和骑行这些小目标在高分辨率的特征图上，更容易检出。车辆的检出需要比较大的感受野，因此在高层次的特征图上检出。

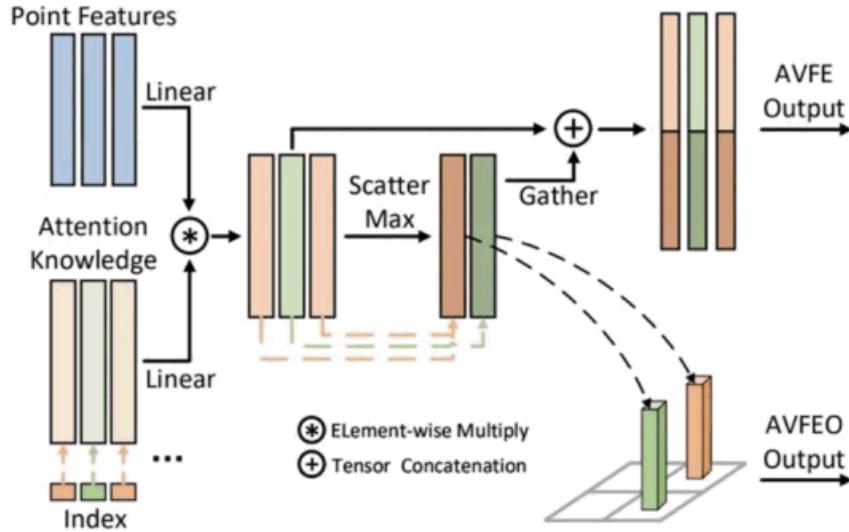
HVNET 的基本结构有 HVFE layer\2D 特征提取层\Detection Head。

HVFE layer

HVFE 的作用是提取不同尺度下点云的体素特征。如下图为了融合不同尺度下的点云的特征，提出了 attentive layer，对于不同层次结构的特征进行叠加融合。为了提高空间使用效率和查找效率，对于特征和体素都使用 index-based 索引。



在整个过程提出了 index-based 的高效操作，使得整个聚集 (scatter)，以及分散 (gather) 的操作能够充分利用 GPU 并行，相比与之前方法的操作，该方式可以有效减少信息的损失以及 GPU 显存的使用。

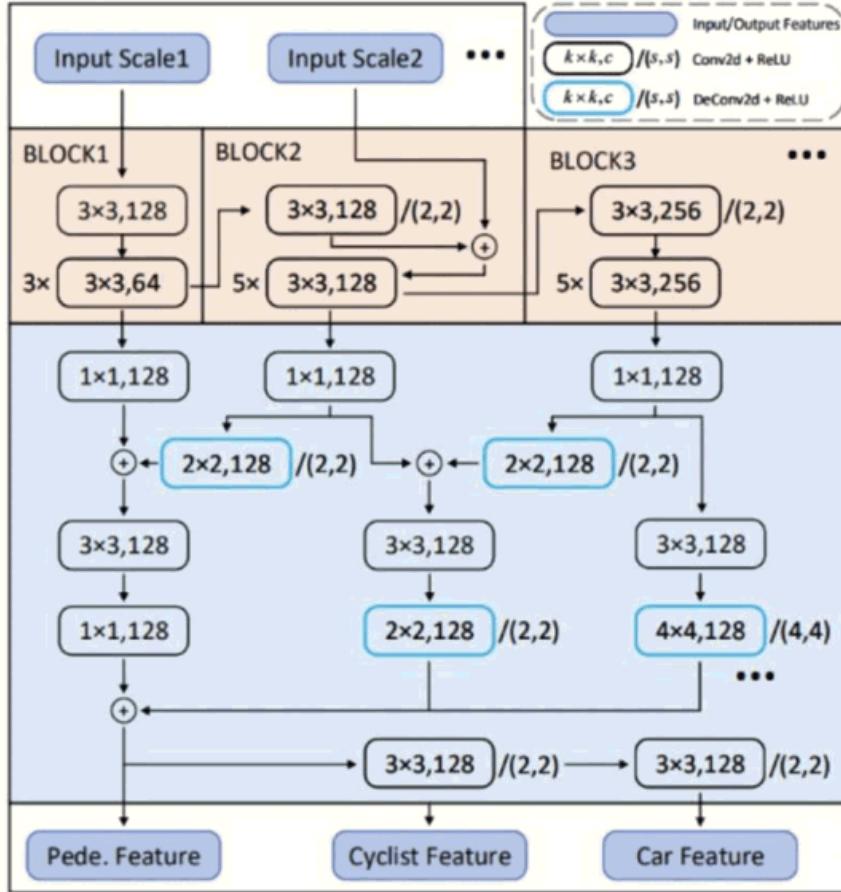


2d 卷积层&& detection head

2d 卷积层

2d 卷积的主要作用是对不同尺度下的体素特征进行融合和更进一步的特征提取。如图，通过类似于特征金字塔的结构将不同细粒度的特征进行融合 detecttion head

利用不同层的 feature map 感受野不同的特性，对不同的层设计相应的 anchor box 进行预测。具体来说，对于不同层的 feature map，在 detection head 部分只会对相对应的尺度的类别进行预测，这样的方式可以有效减少类别间的混淆。



总结与思考：

1. 去掉了 3D CNN，大幅提高了计算速度。同时又通过 HVFE layer 对原始点云进行编码，通过多尺度的 VFE 特征提取，将不同细粒度的特征进行融合，以减少去除 3D CNN，带来的特征整合不够的影响。
2. 在不同尺度上，预测不同大小的目标。在高分辨率的特征图上检测行人，在中分辨率的特征图上检测骑行目标，在粗分辨率的特征图上检测车辆。这是非常不错的尝试，行人和骑行这些小目标在高分辨率的特征图上，更容易检出。车辆的检出需要比较大的感受野，因此在高层次的特征图上检出。

2.4. point-voxel-based methods

voxel-based 方法 (brideyview-based 方法可以看成特殊地 voxel-based 方法) 和 point-based 有各自地优缺点。如下：

voxel-based 方法：

优点：

1. 更高地计算效率，更少地空间存储。（注 1）
3. 更加方便、更加高效的多尺度特征融合，更高的质量的 region proposal。

缺点：

1. 因为体素化导致的信息丢失是不可避免的，小目标\边框\位置回归存在挑战。
(体素内的点云下采样，样，高级特征提取时也可能丢失特征信息)
2. 感受野的大小比较固定，对于比较大的目标（如 bus），检测效果不太好。

pointnet-based 方法：

优点：

1. 保留更好的点云的空间位置信息，边框的回归更加准确。
2. 感受野更加灵活，容易得到更大的感受野。

缺点：

1. 计算效率比较低。（数据非结构性访问，需要进行计算的点云簇很多，有实验表明 4096 个点云簇才能覆盖 90% 的正样本）。

point voxel based method 能够充分利用 voxel-based\pointnet-based 算法的优点，避开 voxel-based\pointnet-based 算法的缺点。目前主流的组合方法是一种二阶目标检测算法。第一阶段采用粗略的候选框提议（保证较高的召回率），在第二阶段对粗候选框中的点进行特征提取，实现位置和方向上的调整，获得更精准的候选框：

1. 一阶段，region proposal。voxel-based 方法具有良好的高效特征提取且高效 region proposal 的特点，更适用于第一阶段的 region proposal。
2. 二阶段，bbox refined。refined.pointnet-based method 可以保留更多的空间信息，且感受野更加灵活。因此适用于 bbox refined。

注：

1. 稀疏 3D 卷积、dynamic voxel 等改进后的 voxel-based method 是高效计算的。如 PointPillars 与 HVNet 则去除了 3D CNN。

2.4.1. Fast Point RCNN

Faster Point RCNN 是一个二阶段的目标检测算法，一阶段是通过 Voxel-Based Method 进行高召回率（实验得到的是 95%）的 region proposal，二阶段通过将原始点云信息和第一阶段的提取的特征信息融合（point-based method）进行边框、位置的 refined。

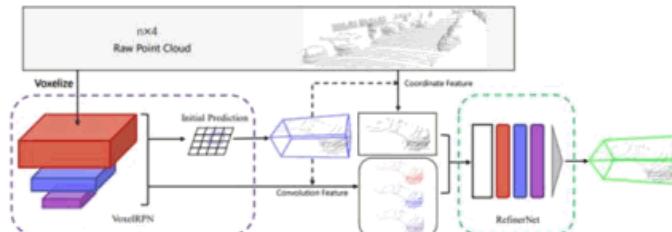


Figure 1. Overview of our two-stage framework. In the first stage, we voxelize point cloud and feed them to VoxelRPN to produce a small number of initial predictions. Then we generate the box feature for each prediction by fusing interior points' coordinates and context feature from VoxelRPN. Box features are fed to RefinerNet for further refinement.

该算法主要包含以下两部分 VoxelRPN 和 RefinerNet。

VoxelRPN

作者直接利用基于 Voxel 的数据处理方式进行点云结构化，然后利用三维卷积和二维卷积的堆叠实现 Voxel 特征的提取（注 1）。

该网络可以看成以下三部分（1）VFE 与 3D 卷积特征提取（2）二维卷积特征提取与 FPN（3）回归和分类分支。

1. VFE 与 3D 卷积特征提取。该模块接受体素化的点云数据为输入，通过 6 层三维卷积进行特征提取。

2. 二维卷积特征提取与 FPN。上面已经实现了 3 维的特征提取了。为了提高效率，作者引入了二维卷积进行特征的进一步提取与感受野提升。并且，在提取过程中，作者引出了三个不同感受野大小的特征图输出，并将这三个特征图进行缩放和堆叠，实现特征融合。

3. 回归和分类分支。上述融合的特征被进一步作为回归和分类分支的特征输入。和其他 RPN 层一样，在 VoxelRPN 层中作者也使用了这两个分支实现对候选框的初步筛选。其中，回归分支的训练采用了 Smooth L1 损失，分类损失采用了 Binary 交叉熵。

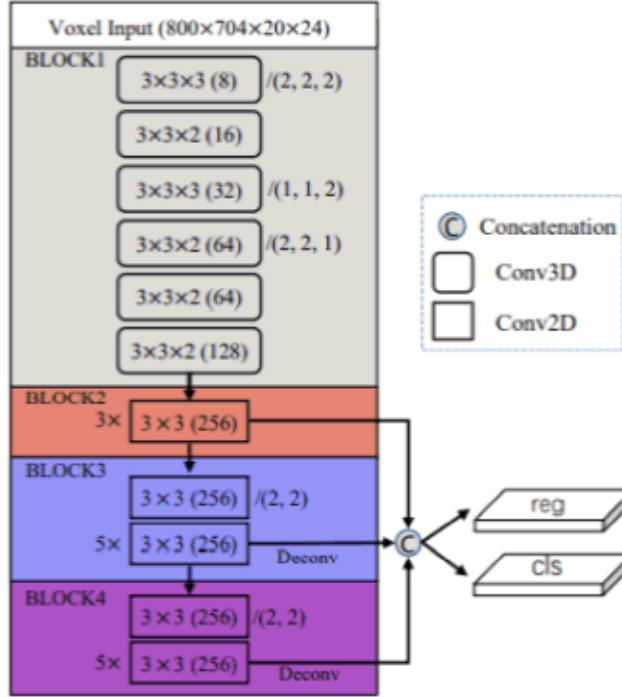


Figure 2. Network Structure of VoxelRPN. The format of layers used in the figure follows (kernel size)(channels) / (stride), i.e., $(k_x, k_y, k_z)(chn)/(s_x, s_y, s_z)$. The default stride is 1 unless otherwise specified.

RefinerNet 1. 将第一阶段检测出来的 3d-box 投影到原始点云的 BEV 上，找到 box 内的原始点云，对这些点通过坐标映射，找到对应的 box feature，得到特征 $n*C$ 。

2. 将每个找到的点云 (x, y, z, i) 输入 MLP，进行特征提取，得到 $n*128$ 特征。

3. 通过 Attention 结构对步骤 1 和步骤 2 得到的点云特征进行融合。

4. 将融合后的特征输入全连接网络输出最终检测结果。

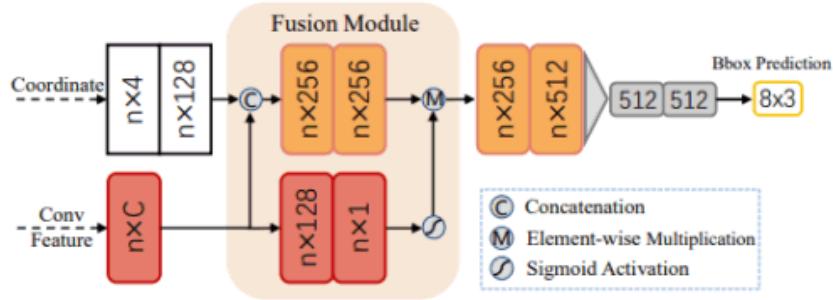


Figure 3. Network Structure of RefinerNet.

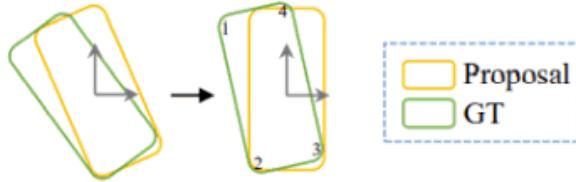


Figure 4. Canonization of a box. The number denotes the order of corner prediction in RefinerNet.

总结:

优势:

1. 可以得到更高召回率更准确的位置框。现在的趋势是结合 voxel-based 的优势和 point-based 的优势，分为 two-stage。stage-1 采用 voxel-based 的方法用类似 one-stage 的检测方法端到端训练；stage-2 采用了 pointnet 对原始点云进行回归。

注:

1. 全三维卷积可以保留 Z 轴的信息，但是效率会比较低，因为运算量大。全二维卷积直接就忽略了 Z 轴的信息了，虽然速度快，但是精度也受到了影响。所以作者先采用这种“三维卷积-二维卷积”的网络结构，先利用“三维卷积”保留 Z 轴的信息，然后为了提高效率，采用了“二维卷积”进行特征提取。实验表明，这种方式确实可以提高效率，而且精度也不会降低。

3. 多传感器融合

参考文献: Multi-modal Sensor Fusion for Auto Driving Perception: A Survey
 LiDar 和 Camera 是自动驾驶的主要传感器。Lidar 数据包含准确的空间信息，但是稀疏\无序\缺乏纹理特征。Camera 数据包含丰富的纹理特征，但是缺乏空间信息。因此如何将 Lidar 和 Camera 数据进行融合，也是自动驾驶领域的重要课题。

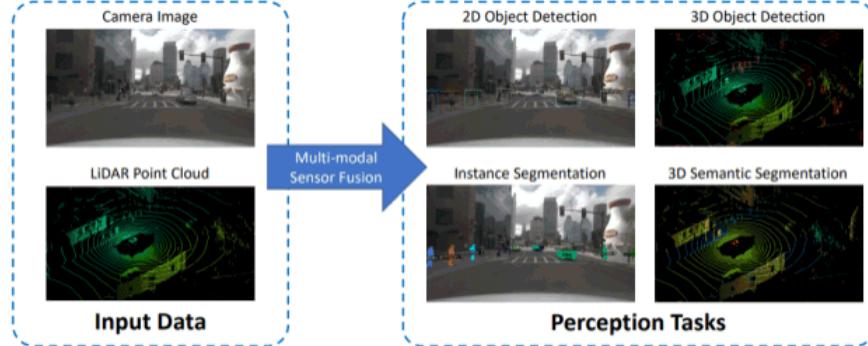


Figure 1. Perception Tasks of Autonomous Driving by Multi-modal Sensor Fusion Model.

多传感器融合可以分为以下几类。强融合是指在数据\特征\结果层面，找到 LiDar 和 Camera 的一一对应关系，然后进行融合的一种方法。而弱融合不是直接在数据\特征\结果层面进行融合，通过某种关系将 Lidar 和 Camera 联系在一起。

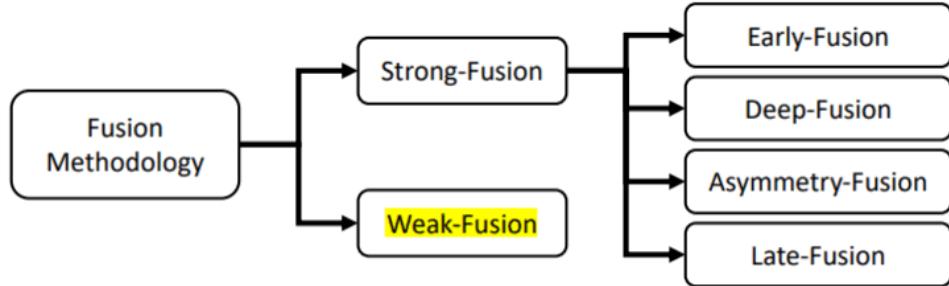


Figure 2. Fusion Methodology Overview

3.1. Strong Fusion

Table 2. Survey of BEV Task Results in KITTI [26] for Test Dataset

Method	Year	Car			Pedestrian			Cyclist		
		Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard
Early-Fusion										
PFF3D [87]	2021	89.61	85.08	80.42	48.74	40.94	38.54	72.67	55.71	49.58
Painted PointRCNN [76]	2020	92.45	88.11	83.36	58.70	49.93	46.29	83.91	71.54	62.97
PI-RCNN [90]	2020	91.44	85.81	81.00	-	-	-	-	-	-
Complexer-YOLO [68]	2019	77.24	68.96	64.95	21.42	18.26	17.06	32.00	25.43	22.88
MVX-Net(PF) [69]	2019	89.20	85.90	78.10	-	-	-	-	-	-
Deep-Fusion										
RoIFusion [9]	2021	92.88	89.03	83.94	46.21	38.08	35.97	83.13	67.71	61.70
EPNet [32]	2020	94.22	88.47	83.69	-	-	-	-	-	-
MAFF-Net [105]	2020	90.79	87.34	77.66	-	-	-	-	-	-
SemanticVoxels [23]	2020	-	-	-	58.91	49.93	47.31	-	-	-
MVAF-Net [78]	2020	91.95	87.73	85.00	-	-	-	-	-	-
3D-CVF [102]	2020	93.52	89.56	82.45	-	-	-	-	-	-
MMF [45]	2019	93.67	88.21	81.99	-	-	-	-	-	-
ConfFuse [46]	2018	94.07	85.35	75.88	-	-	-	-	-	-
SparsePool [85]	2017	-	-	-	43.33	34.15	31.78	43.55	35.24	30.15
Late-Fusion										
CLOCs [55]	2020	92.91	89.48	86.42	-	-	-	-	-	-
Asymmetry-Fusion										
VMVS [40]	2019	-	-	-	60.34	50.34	46.45	-	-	-
MLOD [17]	2019	90.25	82.68	77.97	55.09	45.40	41.42	73.03	55.06	48.21
MV3D [12]	2017	86.62	78.93	69.80	-	-	-	-	-	-
Weak-Fusion										
Faraway-Frustum [104]	2020	91.90	88.08	85.35	52.15	43.85	41.68	79.65	64.54	57.84
F-ConvNet [83]	2019	91.51	85.84	76.11	57.04	48.96	44.33	84.16	68.88	60.05
IPOD [99]	2018	86.93	83.98	77.85	60.83	51.24	45.40	77.10	58.92	51.01
F-PointNet [60]	2017	91.17	84.67	74.77	57.13	49.57	45.48	77.26	61.37	53.78

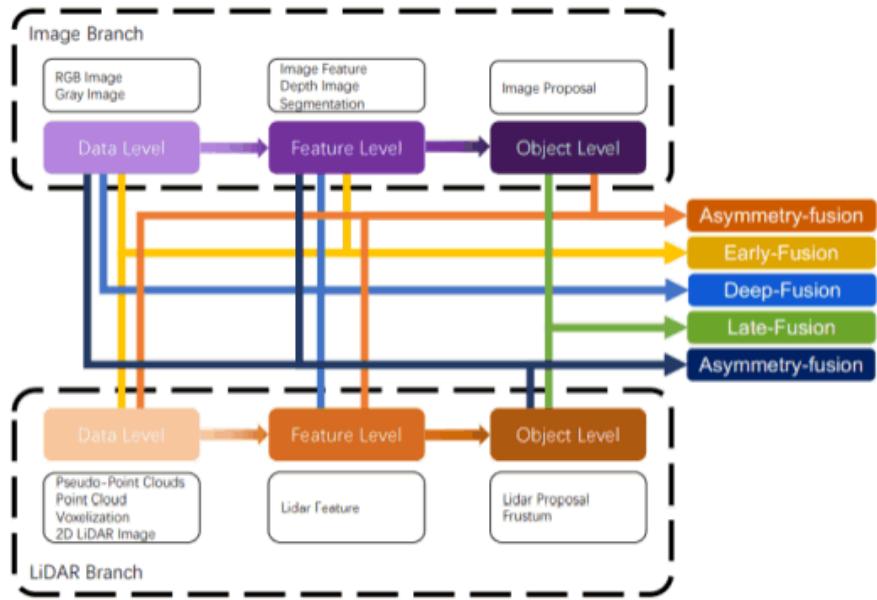


Figure 3. Strong-Fusion Overview

前融合。数据级融合是一种通过原始数据级的空间对齐和投影直接融合每个模态数据的方法，与之不同的是，前融合在数据级是融合激光雷达数据，在数据级或特征级则融合摄像头数据。

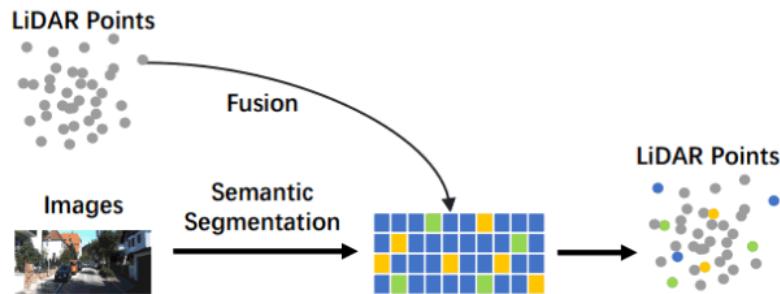


Figure 4. An Example of Early-Fusion

深度融合。深度融合方法在激光雷达分支的特征级对跨模态数据融合，但在图像分支的数据级和特征级做融合。

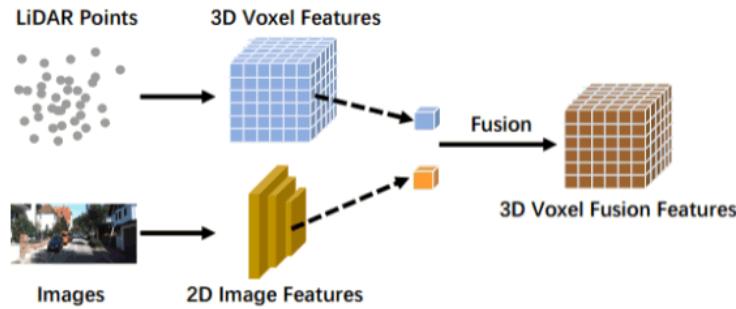


Figure 5. An Example of Deep-Fusion

后融合。后融合，也称为目标级融合，指的是融合每个传感器中 pipeline 结果的方法。

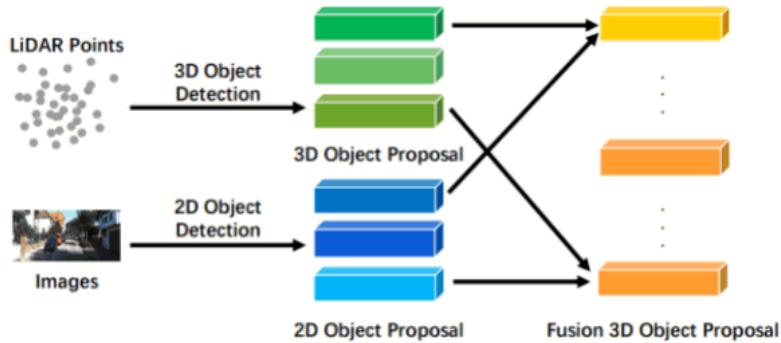


Figure 6. An Example of Late-Fusion

非对称融合。除了早融合、深度融合和后融合外，一些方法以不同的权限处理跨模态分支，因此融合一个分支的目标级信息和其他分支的数据级或特征级信息，定义为**非对称融合**。

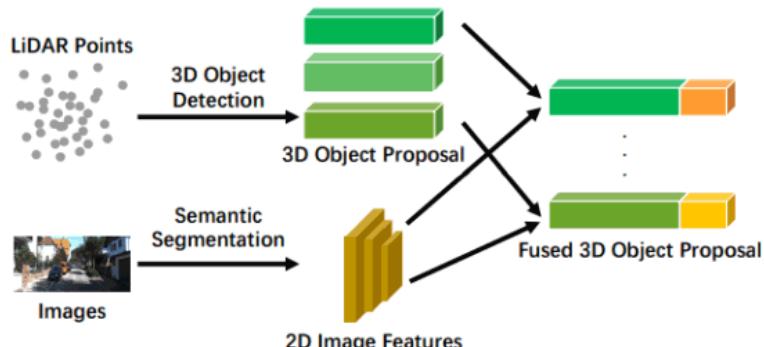


Figure 7. An Example of Asymmetry-Fusion

3.1.1. Early Fusion

前融合。数据级融合是一种通过原始数据级的空间对齐和投影直接融合每个模态数据的方法，与之不同的是，前融合在数据级是融合激光雷达数据，在数据级或特征级则融合摄像头数据。

Method	Year	Car			Pedestrian			Cyclist		
		Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard
Early-Fusion										
PFF3D [87]	2021	89.61	85.08	80.42	48.74	40.94	38.54	72.67	55.71	49.58
Painted PointRCNN [76]	2020	92.45	88.11	83.36	58.70	49.93	46.29	83.91	71.54	62.97
PI-RCNN [90]	2020	91.44	85.81	81.00	-	-	-	-	-	-
Complexer-YOLO [68]	2019	77.24	68.96	64.95	21.42	18.26	17.06	32.00	25.43	22.88
MVX-Net(PF) [69]	2019	89.20	85.90	78.10	-	-	-	-	-	-

- MVX-Net (PF&&VF)

MVX-Net 是在 Voxel-Net 的基础上，将 image 特征进行融合的一种方法。作者提出了两种框架 Point Fusion 和 Voxel Fusion，Point Fusion 属于 Early Fusion，Voxel Fusion 也属于泛意义上的 Deep Fusion。

MVX-Net (PF) 的算法可分为两个分支：

1. 使用一个预训练的图像检测的分支对图像进行特征提取。
2. 将点云体素化，而后将体素中的点云投射到图像平面中，选取对应的图像特征。
3. 将图像特征与体素中的点云进行 concat 融合。
4. 使用 Voxel-based method 进行 3d 目标检测

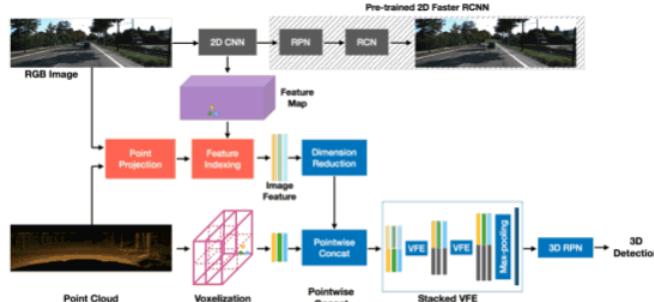


Fig. 2. Overview of the proposed MVX-Net PointFusion method. The method uses convolutional filters of pre-trained 2D faster RCNN to compute the image feature map. Note that the RPN and RCN (shown in shaded rectangle) are not part of the 3D inference pipeline. The 3D points are projected to the image using the calibration information and the corresponding image features are appended to the 3D points. The VFE layers and the 3D RPN process the aggregated data and produce the 3D detections.

相比于 PF 的融合 VF，融合的位置更晚一点。MVX-Net (Voxel Fusion) 算法流程如下：

1. 使用一个预训练的图像检测的分支对图像进行特征提取。
2. 用标定信息将非空体素投影到图像获得 ROI。每个 ROI 的特征放入池化。
3. 将图像特征与 VFE 得到的体素特征进行 concat 融合。
4. 使用 Voxel-based method 进行 3d 目标检测

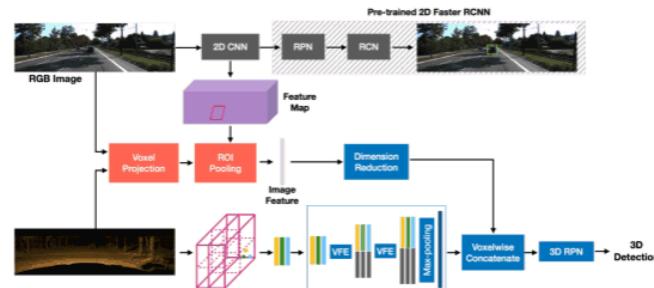


Fig. 3. Overview of the proposed MVX-Net VoxelFusion method. The method uses convolutional filters of pre-trained 2D faster RCNN to compute the image feature map. Note that the RPN and RCN (shown in shaded rectangle) are not part of the 3D inference pipeline. The non-empty voxels are projected to the image using the calibration information to obtain the ROIs. The features within each ROI are pooled and appended to the voxel features computed by VFE layers. The 3D RPN processes the aggregated data and produces the 3D detections.

PF 与 VF 对比：

1. VF 大幅优于 PF 的计算效率。PF 中 voxel 中每个点都会与 image feature 进行融合。VF 则是以非空 voxel 为单位，对 image feature 进行融合。
2. 从效果而言，Point Fusion 的效果要优于 Voxel Fusion 的结果。原因可能是特征融合的程度更加密集。
3. 相比于 VoxelNet，MVX-Net 感知能力并没有提高多少。主要原因可能是异源数据融合存在难度，camera 和 Lidar 的存储形式、表达形式非常不同。过早的进行数据级融合，可以提高的效果很一般。

TABLE II
COMPARISON OF RESULTS ON KITTI VALIDATION SET USING MEAN
AVERAGE PRECISION (MAP) WITH IOU=0.8.

Method	AP _{BEV} (IoU=0.8)			AP _{3D} (IoU=0.8)		
	Easy	Med	Hard	Easy	Med	Hard
Baseline VoxelNet (S)	72.4	62.2	56.5	32.8	28.1	24.6
MVX-Net (VF) (M)	72.2	62.3	61.0	39.5	30.8	29.8
MVX-Net (PF) (M)	74.2	64.5	61.6	43.6	33.2	31.3

- PFF3D

PFF3D (Fast and Accurate 3D Object Detection for LiDAR-camera-based Autonomous Vehicles using One Shared Voxel-based Backbone)

PFF3D 与 MVX-Net (PF) 是非常相似的，都是将图像特征与点云进行 point-wise 的融合。两者的区别：MVX-Net 是对 2D CNN 提取的 image 特征进行点云数据级的融合，而 PFF3D 是在将 image 数据进行 point transform net 后进点云数据级行融合。相比于 MVX-Net (PF)，PFF3D 运行速度更快，因为其采用更轻的图像特征提取网络。

算法流程如下：

1. 将点云投射到图像平面上，得到包含点云坐标 (x, y, z) 的 RGB+ 图像。然后将 RGB+ Image 输入 Point Transfrom Net (Spatial Transformer Networks) 进行特征提取。
2. 将 Point Transfrom Net 得到的特征映射回 3 维空间中，与点云进行 Point-wise Fusion。
3. 输入 voxel-based method 进行目标检测。

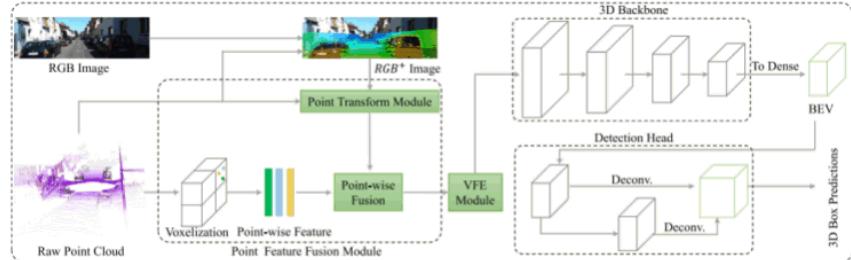


FIGURE 2. The architecture of the proposed one-stage 3D object detection network for the LiDAR and camera. It mainly includes the input data, the point feature fusion module, the 3D backbone, and the detection head. The gray box and green box represent the convolutional block and feature map, respectively.

3.1.2. Deep Fusion

深度融合。深度融合方法在激光雷达分支的特征级对跨模态数据融合，但在图像分支的数据级和特征级做融合。

		Deep-Fusion									
RoIFusion [9]	2021	92.88	89.03	83.94	46.21	38.08	35.97	83.13	67.71	61.70	-
EPNet [32]	2020	94.22	88.47	83.69	-	-	-	-	-	-	-
MAFF-Net [105]	2020	90.79	87.34	77.66	-	-	-	-	-	-	-
SemanticVoxels [23]	2020	-	-	-	58.91	49.93	47.31	-	-	-	-
MIVAF-Net [78]	2020	91.95	87.73	85.00	-	-	-	-	-	-	-
3D-CVF [102]	2020	93.52	89.56	82.45	-	-	-	-	-	-	-
MMF [45]	2019	93.67	88.21	81.99	-	-	-	-	-	-	-
ContFuse [46]	2018	94.07	85.35	75.88	-	-	-	-	-	-	-
SparsePool [85]	2017	-	-	-	43.33	34.15	31.78	43.55	35.24	30.15	-

- ContFuse

ContFuse 是 UberATG 的大作，基本思路是利用双流网络结构在多尺度下对点云特征和图像特征进行深度连续融合，从分类来说是一种 Deep Fusion 算法。ContFuse 通过 continuous fusion layer 将图像与点云进行融合。

continuous fusion layer

1. 对于 bev map 上的每个 target pixel，首先根据 K 近邻找到与之距离最近的 k 个 LiDar Point。（注意是每个，因此是最终得到的 bev map 是稠密的）
2. 将找到的 K 个点投射到 Image Feature 中，找到对应的 image feature。
3. 将 image feature 与 3D offset (k 个 point 与 target pixel 的 offset) 输入 MLP 得到新的特征。

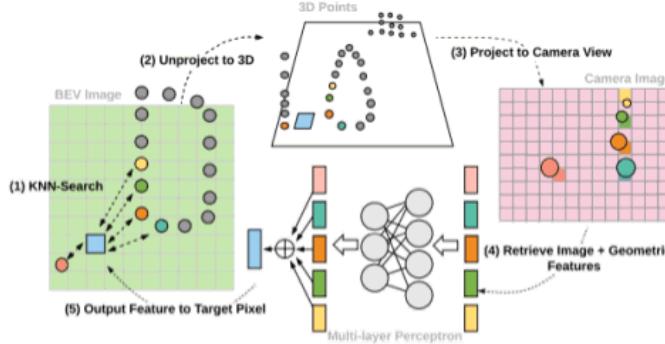


Fig. 2: Continuous fusion layer: given a target pixel on BEV image, we first extract K nearest LIDAR points (Step 1); we then project the 3D points onto the camera image plane (Step 2-Step 3); this helps retrieve corresponding image features (Step 4); finally we feed the image feature + continuous geometry offset into a MLP to generate feature for the target pixel (Step 5).

算法流程如下：

1. 使用预训练 2D CNN 提取图像特征。
2. 将点云投射到 brid eye view (grid-base method) 视角中，使用 2D CNN 提取 bev 特征。
3. 使用 continuous fusion layer 将图像特征与 bev 特征进行 concat 融合，得到稠密的 bev map。

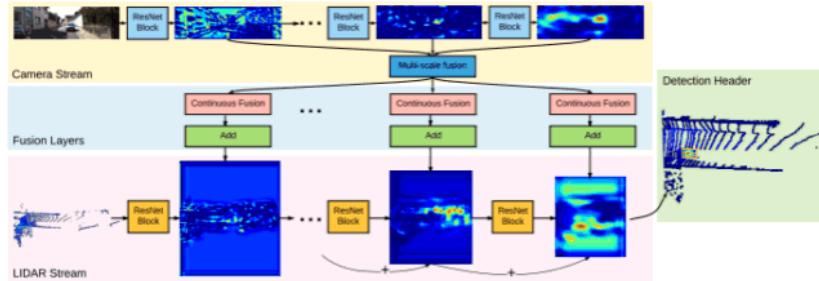


Fig. 1: Architecture of our model. There are two streams, namely the camera image stream and the BEV LIDAR stream. Continuous fusion layers are used to fuse the image features onto the BEV feature maps.

总结与思考：

1. ContFuse 通过将每个 bev 的 pixel 与 image feature 融合得到 dense bev map，一定程度上解决了点云数据过于稀疏的问题，但也增加了一定的冗余计算量。（本人做过实验，dense map 中的大部分融合信息是没有意义的）。
2. 这种基于 grid-based (multiview-based)，丢失的空间信息较多，对于小目标而言存在较大的挑战。

- MMF

MMF 与 ContFuse 一脉相承，都是 UberAGT 的作品。MMF 同时进行 4 个任务（2D detection, Ground estimation, Depth Completion, 3D detection）最终得到 3D 物体检测结果。通过同时进行多个 task 联合学习，该算法能够得到更加好的特征表示，从而进一步提升算法精度。MMF 使用了 point-wise Fusion 和 ROI Fusion 两种融合方法。

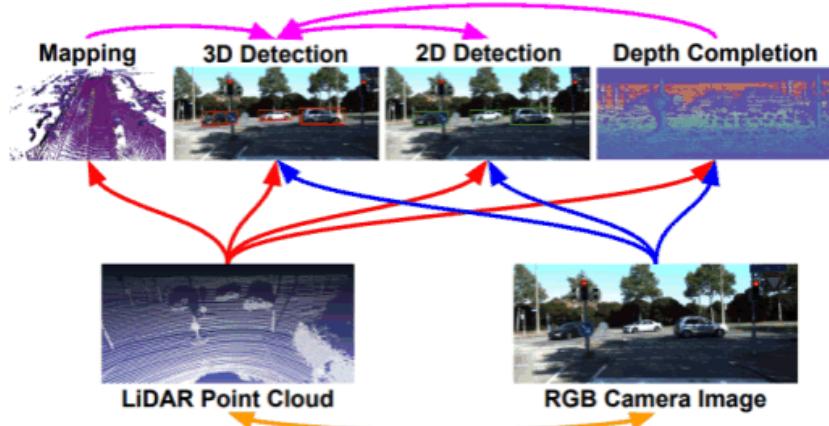


Figure 1. Different sensors (bottom) and tasks (top) are complementary to each other. We propose a joint model that reasons on two sensors and four tasks, and show that the target task - 3D object detection can benefit from multi-task learning and multi-sensor fusion.

网络结构：

1. MMF 同时进行四个任务，分别是图像 2D 目标检测、深度补全、Lidar 3D 目标检测地面估计，最终得到 3D 目标检测结果。通过同时进行多个 task 联合学习，该算法能够得到更加好的特征表示，从而进一步提升算法精度。
2. MMF 由两个分支构成，一个图像特征提取任务，一个是 grid-based (multiview-based) 的 LiDar 特征提取分支。图像特征提取任务主要完成深度补全任务和图像特征提取的任务，LiDar 特征提取分支主要完成 LiDar 特征提取任务、3D Box proposal 任务和地面估计任务。
3. 地面估计。作者把地面估计看成一个回归任务，首先将 bev map 输入一个小型 U-Net (运行时间大约 8ms)，对 bev map 的每个网格都进行高度的估计。对每个激光雷达点的 Z 轴值中减去地面高度，并生成一个新的 bev map (相对于地面)。将新的 bev map 输入后续的 3D 特征提取网络。
4. 深度补全。深度补全与 2D 特征提取任务共享主干网络。通过深度补全可以得到相对稠密点云信息 (pseudo lidar point)。
5. Dense Fusion。和 ContFuse 一样，MMF 采用了 continuous fusion layer，对特征进行 dense fusion。但是不同的是，MMF 通过通过 RGB+种 LiDar 信息反索引到 Bev 特征图上。如果 RGB+中没有真正的点云，则使用 pesudo point cloud 进行反索引。
6. ROI Fusion。由 Bev 特征提取分支提取 3D proposal，而后投射到图像特征图上提取对应的图像特征。将 3d proposal 提取的 bev 特征与图像特征进行融合，最终输出目标的 2D 检测结果和 3D 检测结果。

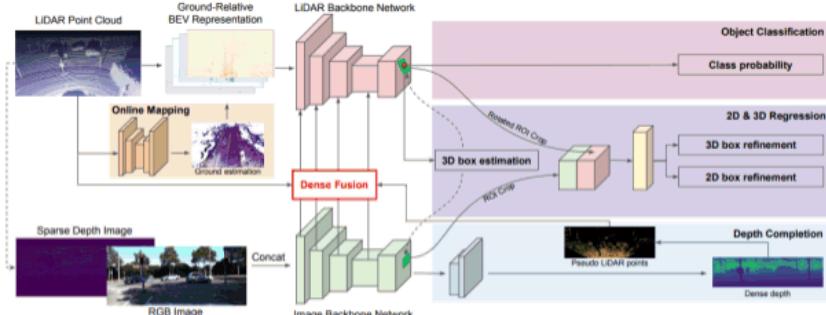


Figure 2. The architecture of the proposed multi-task multi-sensor fusion model for 2D and 3D object detection. Dashed arrows denote projection, while solid arrows denote data flow. Our model is a simplified two-stage detector with densely fused two-stream multi-sensor backbone networks. The first stage is a single-shot detector that outputs a small number of high-quality 3D detections. The second stage applies ROI feature fusion for more precise 2D and 3D box regression. Ground estimation is explored to incorporate geometric ground prior to the LiDAR point cloud. Depth completion is exploited to learn better cross-modality feature representation and achieve dense feature map fusion by transforming predicted dense depth image into dense pseudo LiDAR points. The whole model can be learned end-to-end.

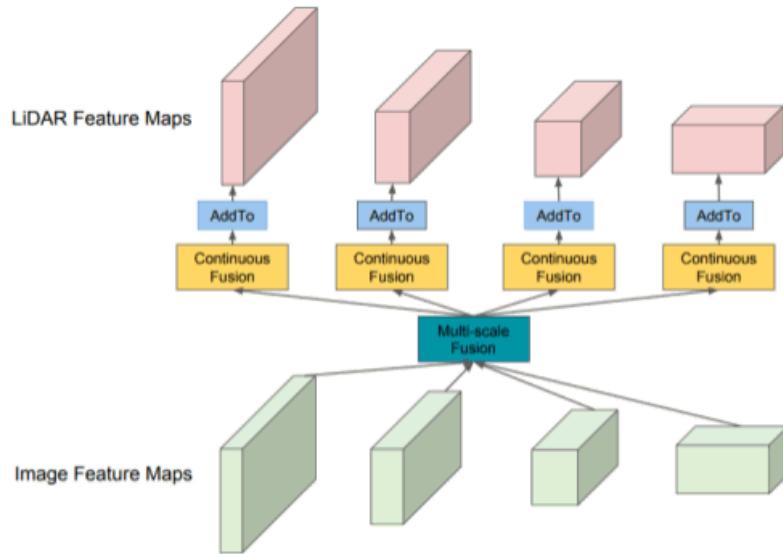


Figure 3. Point-wise feature fusion between multi-scale feature maps from LiDAR and image backbone networks.

总结与思考：

- 是 confuse 的升级版，通过同时进行多个 task 联合学习，该算法能够得到更加好的特征表示，从而进一步提升算法精度。

3.1.3. Late Fusion

Late-Fusion									
CLOCs [55]	2020	92.91	89.48	86.42	-	-	-	-	-

- CLOCs

CLOCs

目前的 fusion 普遍都是在特征层面或者数据层面进行融合，效果不如纯 lidar 的方法。原因主要是 Lidar 数据与 Camera 数据的表现形式过于不同：

- 视角不同，camera 为 FV 视角，LiDar 为 BEV 视角。
- 表征形式不同，camera 通过 RGB 表达，LiDar 则通过 xyz 表达。
- 存储形式不同，camera 规则存储，LiDar 则是无序存储。

CLOCs 可以不需要在重新训练已有模型的情况下，通过 Geometric-consistency 和 Semantic-consistency，将结果层面的 2D Object 与 3D Object 进行后融合，并可以有效提高结果的精度。

Geometric-consistency

假设图像和点云中同时存在一个目标，那么 3D 检测和 2D 检测的目标边框是基本一致的。而当出现误检，则 2D 检测和 3D 检测的结果是不能对应上的。于是作者使用 Geometric-consistency 进行多模态检测结果进行关联。

Semantic consistency

对于同一个目标，不同模态的检测器得到的类别也应当一；如果不一致，大概率不是同一个目标。于是作者通过 Semantic consistency 对多模态检测结果进行关联。

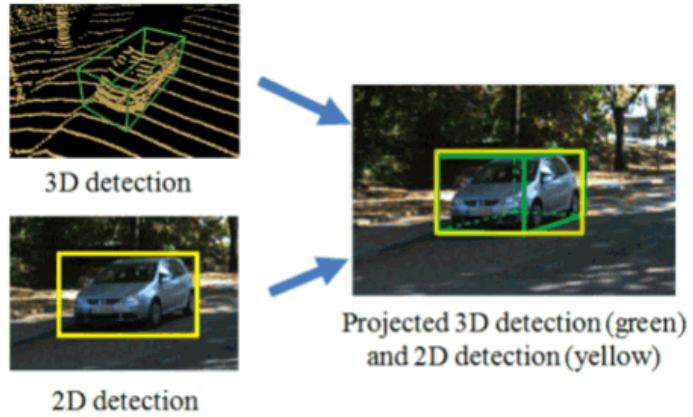


Fig. 2: 2D and 3D object detection. An object that is correctly detected by both a 2D and 3D detector will have highly overlapped bounding boxes in the image plane.

算法流程：

1. Image 目标检测器和 LiDar 目标器分别提出自己的 proposals，在该阶段 proposals 的置信度阈值可以尽可能的低，尽可能提出更多的 proposals。
2. 将两种模态得到的 proposals 编码成 $k \times n \times 4$ 的稀疏张量。k 表示 k 个 2D proposal, n 表示 n 个 3D proposal, 两种模态的每个 proposal, 分别对应编码 T。其中 IoU (图像平面) 表示两种模态的 Geometric-consistency, S(2D) 表示 2D proposal 的置信度, S(3D) 表示 3D proposal 的置信度, d 表示两种模态的在图像平面下 proposal 对应的距离。如果两种 proposal 不符合 geometric consistency 和 semantic consistency。则将 IoU 和 S(2D) 设置为 -1。

$$\mathbf{T}_{i,j} = \{IoU_{i,j}, s_i^{2D}, s_j^{3D}, d_j\} \quad (3)$$

3. 对于非空元素（符合 geometric consistency 和 semantic consistency 的 proposal）采用 1×1 的 2D 卷积进行进一步特征提取和融合，输出每个非空元素的融合置信度。
4. 将非空元素的融合置信度进行反索引，输出 n 个 3D proposals 的置信度，通过正常的置信度阈值筛选出最终的 3D 目标检测结果。

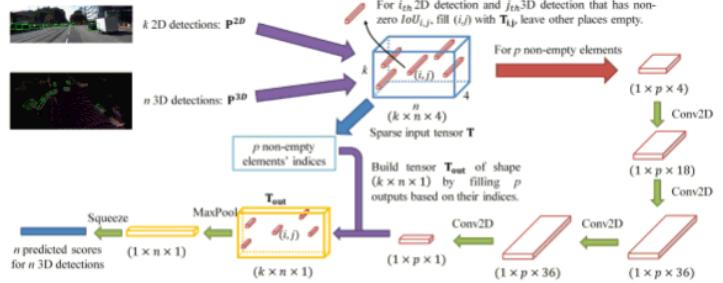


Fig. 3: CLOCs Fusion network architecture. First, individual 2D and 3D detection candidates are converted into a set of consistent joint detection candidates (a sparse tensor, the blue box); Then a 2D CNN is used to process the non-empty elements in the sparse input tensor; Finally, this processed tensor is mapped to the desired learning targets, a probability score map, through maxpooling.

实验：

可以看到 CLOCs 对 3D 目标检测的提升是很大的。

Detector	Input Data	3D AP (%)			Bird's Eye View AP (%)		
		easy	moderate	hard	easy	moderate	hard
SECOND (baseline) [6]	LiDAR	83.34	72.55	65.82	89.39	83.77	78.59
CLOCs_SecCas (SECOND+Cascad R-CNN)	LiDAR+Img	86.38	78.45	72.45	91.16	88.23	82.63
<i>Improvement (CLOCs_SecCas over SECOND)</i>	-	+3.04	+5.90	+6.63	+1.77	+4.46	+4.04
PointRCNN (baseline) [8]	LiDAR	86.23	75.81	68.99	92.51	86.52	81.39
CLOCs_PointCas (PointRCNN+Cascad R-CNN)	LiDAR+Img	87.50	76.68	71.20	92.60	88.99	81.74
<i>Improvement (CLOCs_PointCas over PointRCNN)</i>	-	+1.27	+1.04	+2.21	+0.09	+2.47	+0.35
PV-RCNN (baseline) [22]	LiDAR	87.45	80.28	76.21	91.91	88.13	85.41
CLOCs_PVCas (PV-RCNN+Cascad R-CNN)	LiDAR+Img	88.94	80.67	77.15	93.05	89.80	86.57
<i>Improvement (CLOCs_PVCas over PV-RCNN)</i>	-	+1.49	+0.39	+0.94	+1.14	+1.67	+1.17
F-PointNet [24]	LiDAR+Img	82.19	69.79	60.59	91.17	84.67	74.77
AVOD-FPN [12]	LiDAR+Img	83.07	71.76	65.73	90.99	84.82	79.62
F-ConvNet [25]	LiDAR+Img	87.36	76.39	66.69	91.51	85.84	76.11
UberATG-MMF [26]	LiDAR+Img	88.40	77.43	70.22	93.67	88.21	81.99
UberATG-ContFuse [14]	LiDAR+Img	83.68	68.78	61.67	94.07	85.35	75.88

结论与思考：

个人认为 CLOCs 是一种让人眼前一亮的多模态检测融合方法。

1. CLOCs 对于原有算法的计算开销增加是非常少，也不需要重新训练原有算法就可以使用。

2. 通过减少 2D 和 3D proposal 的置信度阈值尽可能得提出更多对应 proposal，减少算法的漏检现象。通过 **geometric consistency** 和 **semantic consistency** 的先验知识，减少算法的误检现象。最终可明显得改善算法的检测结果。

如下图，使用 CLOCs 可以明显改善原算法的漏检和误检现象。

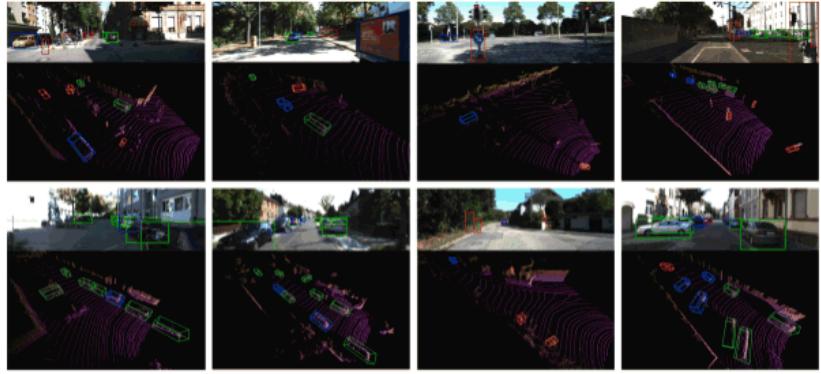


Fig. 5: Qualitative results of our CLOCs on KITTI test set compared to SECOND [6]. Red and blue bounding boxes are false and missed detections from SECOND respectively that are corrected by our CLOCs. Green bounding boxes are correct detections. The upper row in each image is the 3D detection projected to image, the others are 3D detections in LiDAR point clouds.

3.1.4. Asymmetry Fusion

Asymmetry-Fusion										
	2019	-	-	-	60.34	50.34	46.45	-	-	-
VMVS [40]	2019	-	-	-	-	-	-	-	-	-
MLOD [17]	2019	90.25	82.68	77.97	55.09	45.40	41.42	73.03	55.06	48.21
MV3D [12]	2017	86.62	78.93	69.80	-	-	-	-	-	-

- MV3D

MV3D 将点云 BEV-map frontview-map 和 RGB image 的特征进行多模态融合，然后进行定位和目标识别。

输入数据的表达如下：

1. 手工提取的 BEV features。手工特征有三部分构成，分别是 `height map`, `density map`、`intensity map`。
2. front view features，是通以下公式投射到前向图上的得到。

$$\begin{aligned} c &= \lfloor \text{atan2}(y, x) / \Delta\theta \rfloor \\ r &= \lfloor \text{atan2}(z, \sqrt{x^2 + y^2}) / \Delta\phi \rfloor, \end{aligned} \quad (1)$$

3. 前向 RGB image。

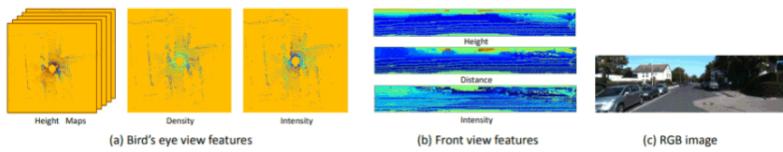


Figure 2: Input features of the MV3D network.

网络结构图如下：

该网络结构主要由两部分组成，分别是 3D proposal network 和 Region based fusion network 构成。

3D proposal network

3D proposal network 可分为三个分支，输入分别是 LiDAR bev map\LiDAR front view\Image(RGB). 这三个分支都是使用 CNN 提取特征，而在 bev map 分支中多增加了 3D proposal (bev map 中包含的空间信息更多，遮挡最少)。将 3D proposal 分别投射到每个分支的 feature map 上，然后对应的 ROI 区域进行 pooling。

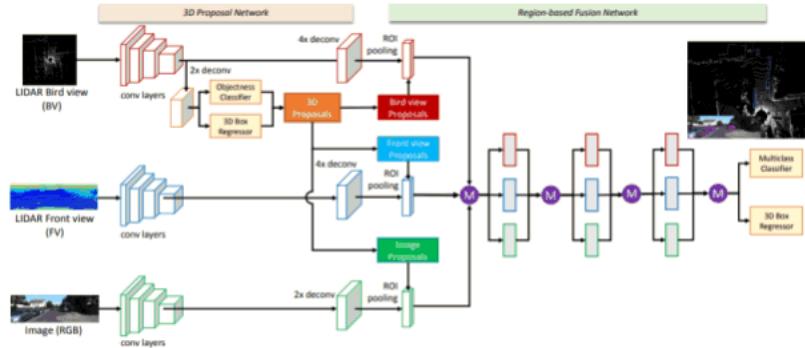


Figure 1: **Multi-View 3D object detection network (MV3D)**: The network takes the bird's eye view and front view of LiDAR point cloud as well as an image as input. It first generates 3D object proposals from bird's eye view map and project them to three views. A deep fusion network is used to combine region-wise features obtained via ROI pooling for each view. The fused features are used to jointly predict object class and do oriented 3D box regression.

Region based fusion network

将 3D proposal network 得到的 ROI 特征进行融合，最终输出目标的类别和坐标位置。

消融实验结果：

Data	AP _{3D} (IoU=0.5)			AP _{loc} (IoU=0.5)			AP _{2D} (IoU=0.7)		
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
FV	67.6	56.30	49.98	74.02	62.18	57.61	75.61	61.60	54.29
RGB	73.68	68.86	61.94	77.30	71.68	64.58	83.80	76.45	73.42
BV	92.30	85.50	78.94	92.90	86.98	86.14	85.00	76.21	74.80
FV+RGB	77.41	71.63	64.30	82.57	75.19	66.96	86.34	77.47	74.59
FV+BV	95.19	87.65	80.11	95.74	88.57	88.13	88.41	78.97	78.16
BV+RGB	96.09	88.70	80.52	96.45	89.19	80.69	89.61	87.76	79.76
BV+FV+RGB	96.02	89.05	88.38	96.34	89.39	88.67	95.01	87.59	79.90

Table 4: **An ablation study of multi-view features**: Performance are evaluated on KITTI validation set.

总结与思考： 1. MV3D 是较早的一种 3D 目标检测算法，也曾是 kitti 的榜一大哥，具有很大的开创性意义。

2. LiDar front view 这一分支对于 3D 目标检测而言，效果提升比较一般，但是增加了接近三分之一的时间消耗。因此不是一个经济实惠的选择。

3.2. Weak Fusion

与强融合不同，弱融合方法不会以多种方式直接从分支融合数据/特征/目标，而是以其他方式操作数据。基于弱融合的方法通常使用基于规则的方法来利用一种模态数据作为监督信号，以指导另一模态的交互。

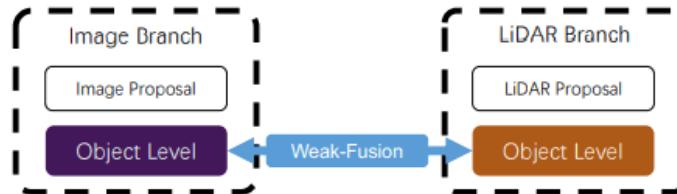


Figure 8. Weak-Fusion Overview

3.2.1. F-PointNet

F-PointNet 和之前介绍的方法不同，并不是将点云和图像进行数据或者特征层面的融合，而是借助图像的目标检测结果，为 LiDar 检测划分感兴趣区域，然后使用 pointnet-based method 进行点云的目标检测。

3D 视锥体(frustum)

图像中的目标检测结果是一个 2d 框，但是投射到点云空间中则是一个视锥体。

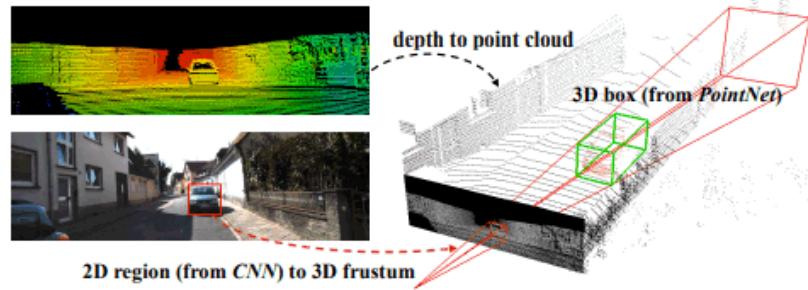


Figure 1. 3D object detection pipeline. Given RGB-D data, we first generate 2D object region proposals in the RGB image using a CNN. Each 2D region is then extruded to a *3D viewing frustum* in which we get a point cloud from depth data. Finally, our frustum PointNet predicts a (oriented and amodal) 3D bounding box for the object from the points in frustum.

算法流程如下：

1. 通过一个 2D 检测器生成 2D 候选框和类别信息。
2. 将 2D 候选框投射到点云空间中，形成一个视锥体。
3. 使用 pointnet-based method 进行点云分割和 3D 目标检测。

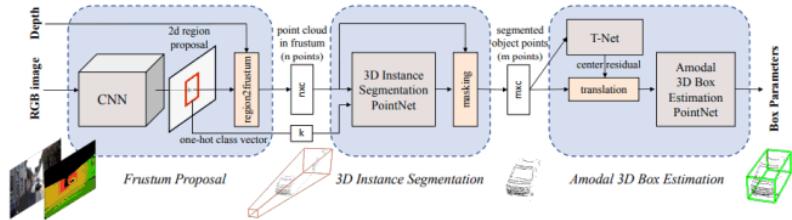


Figure 2. Frustum PointNets for 3D object detection. We first leverage a 2D CNN object detector to propose 2D regions and classify their content. 2D regions are then lifted to 3D and thus become frustum proposals. Given a point cloud in a frustum ($n \times c$ with n points and c channels of XYZ, intensity etc. for each point), the object instance is segmented by binary classification of each point. Based on the segmented object point cloud ($m \times c$), a light-weight regression PointNet (T-Net) tries to align points by translation such that their centroid is close to amodal box center. At last the box estimation net estimates the amodal 3D bounding box for the object. More illustrations on coordinate systems involved and network input, output are in Fig. 4 and Fig. 5.

结论与思考：

1. F-PointNet 非常依赖于图像的目标检测结果，因此光照、可见度等会对检测结果造成比较大的影响。
2. 由于 camera 的 fov 一般比较小，因此该方法的感知范围也比较小，不能发挥 LiDar 的优势。
3. 一般来说 camera 的安装位置都比较矮，因此存在比较大的遮挡问题。