# 3D目标检测精炼

## 1. 数据集

### 1.1 kitti

https://github.com/chinabing/PointPillars

原始数据目录

```
kitti
├── testing
│   ├── calib
│   ├── image_2
│   ├── velodyne
├── training
│   ├── calib
│   ├── image_2
│   ├── label_2
│   ├── velodyne
```

执行 python pre_process_kitti.py --data_root  your_path_to_kitti

数据目标转换成如下所示，增加了kitti_gt_database、kitti_infos_xxx.pkl文件

```
kitti
├── testing
│   ├── calib
│   ├── image_2
│   ├── velodyne
│   ├── velodyne_reduced
├── training
│   ├── calib
│   ├── image_2
│   ├── label_2
│   ├── velodyne
│   ├── velodyne_reduced
├── kitti_gt_database
```

```
14 │   ├── xxxxx.bin
15 ├── kitti_infos_train.pkl
16 ├── kitti_infos_val.pkl
17 ├── kitti_dbinfos_train.pkl
18 ├── kitti_infos_test.pkl
19 ├── kitti_infos_trainval.pkl
```

- `kitti_gt_database/xxxxx.bin`：训练数据集中包含在 3D 标注框中的点云数据。文件个数为总的目标数，19700个bin文件。

```
1 000000_Pedestrian_0.bin
2 其中:
3 000000: 代表该目标所对应的图像或者点云文件的名称id
4 Pedestrian: 代表该目标所对应的类别名称
5 0: 代表该目标在当前000000文件中的目标id（假设一共8个检测目标，该目标为第0个）
6
```

- `kitti_infos_train.pkl`：训练数据集属性打包文件，数据结构如下

```
1 [
2   {（第一个样本）
3       point_cloud:{
4                     num_features: 使用的特征数 3 或者 4 （xyz 还是xyzi等等）
5                     lidar_idx: "000000" （雷达bin文件对应的文件名称）
6                   },
7       image: {
8                 image_idx : 图像文件对应的文件名称,
9                 image_shape: 图像的大小高宽
10                image_path: 图像路径
11              },
12      calib: {
13                P2: 矫正后的0号相机相对于2号相机的投影矩阵
14                R0_rect: 0号相机矫正矩阵，为了让各个相机可以共面，这样代码中的深度
    计算才正确
15                Tr_velo_to_cam: 0号相机坐标系相对于雷达坐标系的变换矩阵
16              },
17      annos: {
18                name: 当前样本中对应的检测目标的名称数组
19                truncated: 当前样本中对应的检测目标的截断系数数组。 系数越大，截断越
    多
20                occluded: 遮挡系数 0，1，2 越大遮挡越大
```

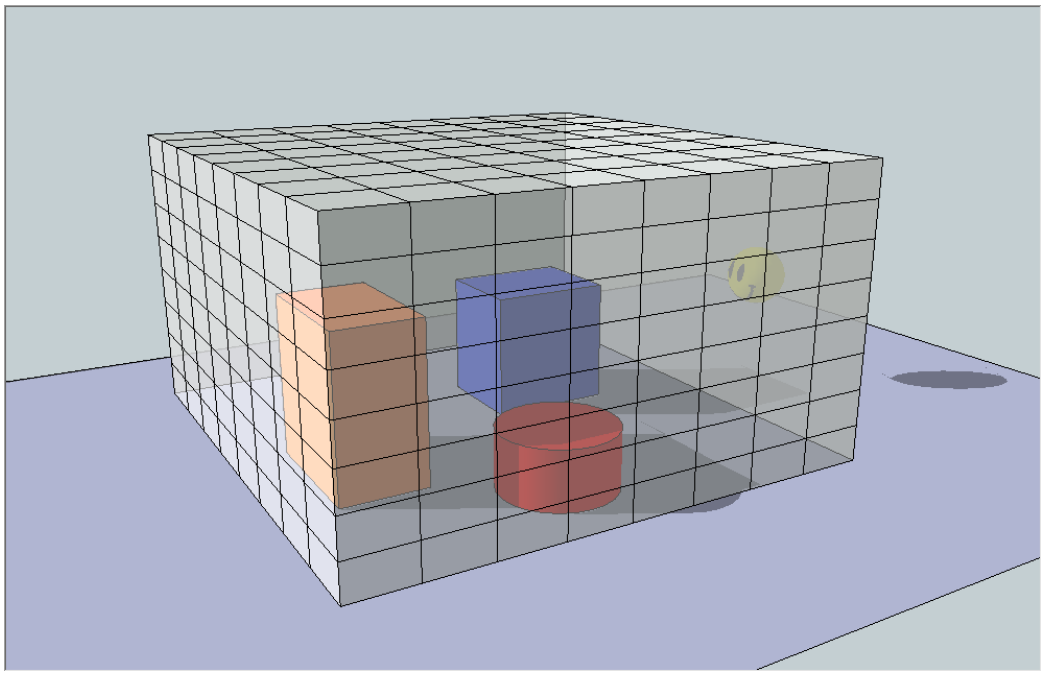
```
                alpha: 将目标转至正前方观测其与相机x轴之间的角度
                bbox: 目标在2号相机下的box框的x1y1x2y2格式
                demensions: 物体的长高宽
                location: 目标框的底部中心点在矫正后的0号坐标系下的坐标
                rotation_y: 从目标行驶方向前方观测目标和相机坐标系x轴的夹角
                score:置信度 一般不使用
                difficulty: 难易程度，该参数按照图像的高度、截断程度、遮挡程度来进行综合区分得到
                index: 对应数组中的检测目标在总的目标中的id
                gt_boxes_lidar: 雷达坐标系下的物体的x y z l w h yaw
                num_points_in_gt: 各个目标框中存在的点个数
            }
    },
    {（第二个样本）
        ......
    },
    ......
    {（第N个样本）
        ......
    }
]
```

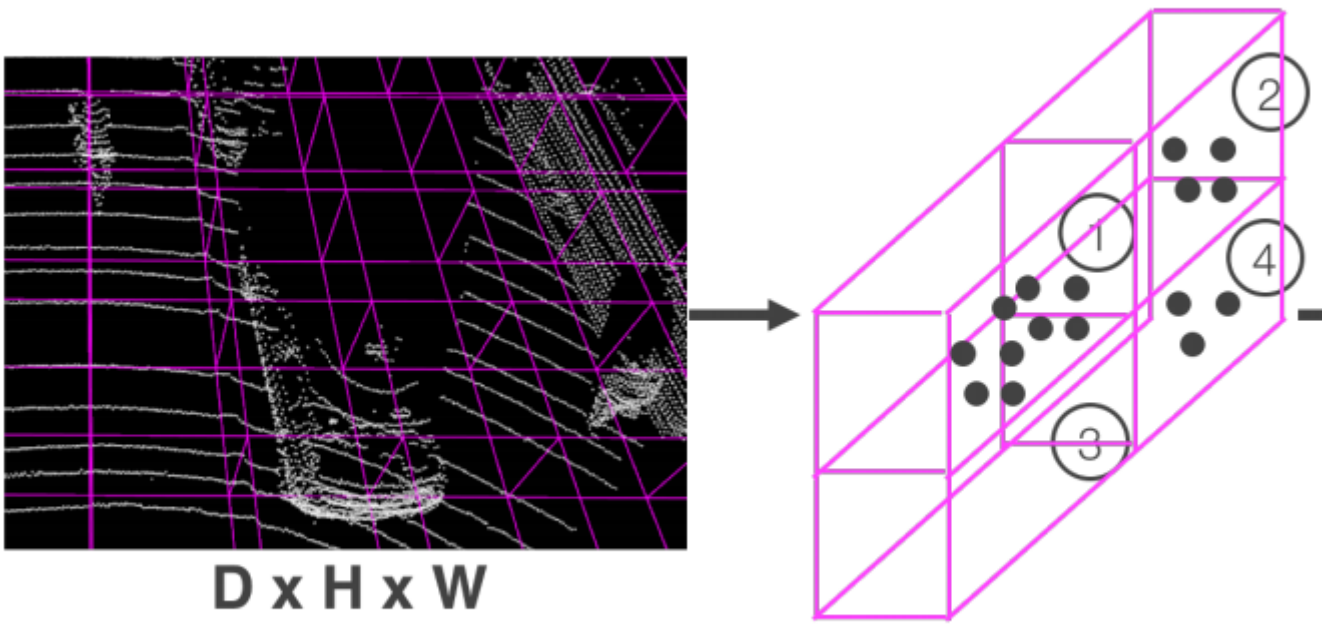
### 1.2 NuScenes

## 2. 3D目标检测算法

### 2.1 PointNet

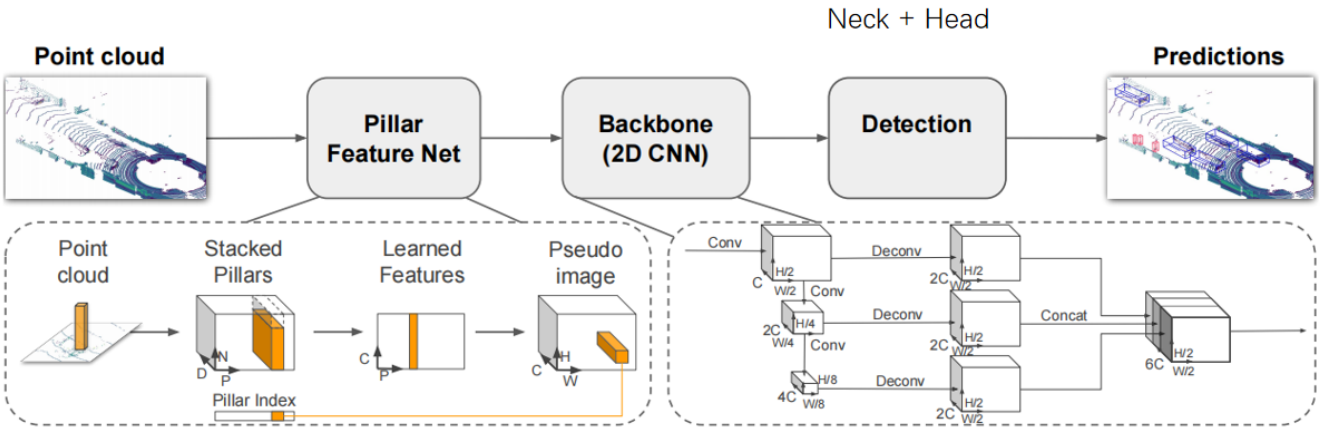
### 2.2 VoxelNet






## 2.3 PointPillars

# PointPillars




Voxelization

pillars内部去均值、pillar之间去中心、特征embedding

scatter

[18279,4] ⟶ [6815,32,4] ⟶ [6815,32,9] ⟶ [6815,9,32] ⟶ [6815,64,32] ⟶ [6815,64] ⟶ [64,496,432]

## 2.4 CenterPoint

# 3. 代码讲解

https://github.com/chinabing/PointPillars