SqueezeSeg: Convolutional Neural Nets with Recurrent CRF for Real-Time Road-Object Segmentation from 3D LiDAR Point Cloud

`translation`

https://arxiv.org/abs/1710.07368

SqueezeSeg代码地址：SqueezeSeg: Convolutional Neural Nets with Recurrent CRF for Real-Time Road-Object Segmentation from 3D LiDAR Point Cloud

摘要

在本文中，我们从三维激光雷达点云的角度对道路目标进行了语义分割。我们特别希望检测和分类感兴趣的实例，例如汽车、行人和骑自行车的人。我们制定这个问题作为一个逐点分类的问题,并提出一个端到端的管道称为SqueezeSeg基于卷积神经网络(CNN):CNN需要改变激光雷达点云直接输出作为输入,并逐点地标签地图,然后精制的条件随机场(CRF)实现为复发性层。然后用传统的聚类算法得到实例级的标签。我们的CNN模型是在来自KITTI1数据集的激光雷达点云上训练的，我们的逐点分割标签来自于KITTI的3D边框。为了获得额外的训练数据，我们在广受欢迎的视频游戏《侠盗飞车V》(GTA-V)中构建了一个激光雷达模拟器，以合成大量真实的训练数据。我们的实验表明,SqueezeSeg以惊人的快速和稳定性，每帧(8.7±0.5)ms的高精度运行,高度可取的自主驾驶的应用程序。此外，对综合数据的训练可以提高对真实数据的验证准确性。我们的源代码和合成数据将是开源的。

1.介绍

自动驾驶系统依赖于对环境的准确、实时和鲁棒的感知。自动驾驶汽车需要精确地分类和定位“道路物体”，我们将其定义为与驾驶有关的物体，如汽车、行人、自行车和其他障碍物。不同的自动驾驶解决方案可能有不同的传感器组合，但3D激光雷达扫描仪是最普遍的组件之一。激光雷达扫描仪直接产生环境的距离测量，然后由车辆控制器和计划人员使用。此外，激

光雷达扫描仪在几乎所有的光照条件下都是健壮的，无论是白天还是黑夜，有或没有眩光和阴影。因此，基于激光雷达的感知任务引起了广泛的研究关注。

在这项工作中，我们关注道路目标分割使用(Velodyne风格)三维激光雷达点云。给定激光雷达扫描仪的点云输出，任务的目标是隔离感兴趣的对象并预测它们的类别，如图1所示。以前的方法包括或使用以下阶段的部分:删除地面，将剩余的点聚到实例中，从每个集群中提取(手工制作)特性，并根据其特性对每个集群进行分类。这种模式,尽管它的受欢迎程度2,3,4,5,有几个缺点:a)地面分割在上面的管道通常依赖于手工特性或决策规则,一些方法依赖于一个标量阈值6和其他需要更复杂的特性,比如表面法线7或不变的描述符4,所有这些可能无法概括,后者需要大量的预处理。b)多级管道存在复合误差的聚合效应，上面管道中的分类或聚类算法无法利用上下文，最重要的是对象的直接环境。c)很多去除地面的方法都依赖于迭代算法，如RANSAC (random sample consensus) 5， GP-INSAC (Gaussian Process Incremental sample consensus)2，agglomerative clustering2。这些算法组件的运行时间和精度取决于随机初始化的质量，因此可能不稳定。这种不稳定性对于许多嵌入式应用程序(如自动驾驶)来说是不可接受的。我们采取了另一种方法:使用深度学习来提取特征，开发一个单阶段的管道，从而避开步骤迭代算法。

本文提出了一种基于卷积神经网络(CNN)和条件随机场(CRF)的端到端管道。CNNs和CRFs已成功应用于二维图像8、9、10、[11]的分割任务。为了将CNNs应用于三维激光雷达点云，我们设计了一个CNN，它接受变换后的激光雷达点云，并输出标签点地图，通过CRF模型进一步细化。然后，通过对一个类别中的点应用传统的聚类算法(如DBSCAN)来获得实例级标签。为了将3D点云提供给2D CNN，我们采用球面投影将稀疏的、不规则分布的3D点云转换为密集的2D网格表示。所提出的CNN模型借鉴了squeeze zenet[12]的思想，经过精心设计，降低了参数大小和计算复杂度，目的是降低内存需求，实现目标嵌入式应用程序的实时推理速度。将CRF模型重构为一个循环神经网络(RNN)模块为[11]，可以与CNN模型进行端到端训练。我们的模型是在基于KITTI数据集1的激光雷达点云上训练的，点分割标签是从KITTI的3D边框转换而来的。为了获得更多的训练数据，我们利用Grand Theft Auto V (GTA-V)作为模拟器来检索激光雷达点云和点级标签。
实验表明，这种方法精度高、速度快、稳定性好，适用于自动驾驶。我们还发现，用人工的、噪声注入的模拟数据替代我们的数据集进一步提高了对真实世界数据的验证准确性。

2. 相关工作

A. 3维激光雷达点元的语义分割

以前的工作在激光雷达分割中看到了广泛的粒度范围，处理从特定组件到整个管道的任何事情。7提出了基于网格的地面和基于局部表面凹凸性的目标分割。2总结了几种基于迭代算法的诸如RANSAC (random sample consensus)和GP-INSAC (gaussian process incremental sample consensus)的地面去除方法。最近的工作也集中在算法效率上。5提出了有效的地面分割和聚类算法，而[13]绕过地面分割直接提取前景对象。4将重点扩展到整个管道，包括分割、聚类和分类。提出了将点斑块重新划分为不同类别的背景和前景对象，然后使用EMST-RANSAC5进一步集群实例。

B. 3D点云CNN

CNN方法考虑的是二维或三维的激光雷达点云。处理二维数据时考虑的是用激光雷达点云投影自顶向下[14]或从许多其他视图[15]投影的原始图像。其他工作考虑的是三维数据本身，将空间离散为体素和工程特征，如视差、平均和饱和度[16]。无论数据准备如何，深度学习方法都考虑利用二维卷积[17]或三维卷积[18]神经网络的端对端模型。

C. 图像的语义分割

CNNs和CRFs都被用于图像的语义分割任务。8提议将经过分类训练的CNN模型转换为完全卷积网络来预测像素级标签。9提出了一种用于图像分割的CRF公式，并用均值-场迭代算法近似求解。CNNs和CRFs合并在10中，CNN用于生成初始概率图，CRF用于细化和恢复细节。在[11]中，平均场迭代被重新表述为一个递归神经网络(RNN)模块。

D. 模拟数据采集

获取注释，特别是点或像素级的注释对于计算机视觉任务来说通常是非常困难的。因此，合成数据集引起了越来越多的关注。在自动驾驶社区中，视频游戏《侠盗猎车手》被用来检索数据，用于目标检测和分割[19]、[20]。

3.方法描述

A. 点云转换

传统CNN模型操作图像,可以由3-dimentional张量的大小 H × W × 3表示。前二维编码空间位置，其中H和W分别为图像高度和宽度。最后一个维度编码特性，最常见的是RGB值。然而，三维激光雷达点云通常表示为一组笛卡尔坐标(x, y, z)，也可以包含额外的特征，如强度或RGB值。与图像像素的分布不同，激光雷达点云的分布通常是稀疏而不规则的。因此，纯粹地

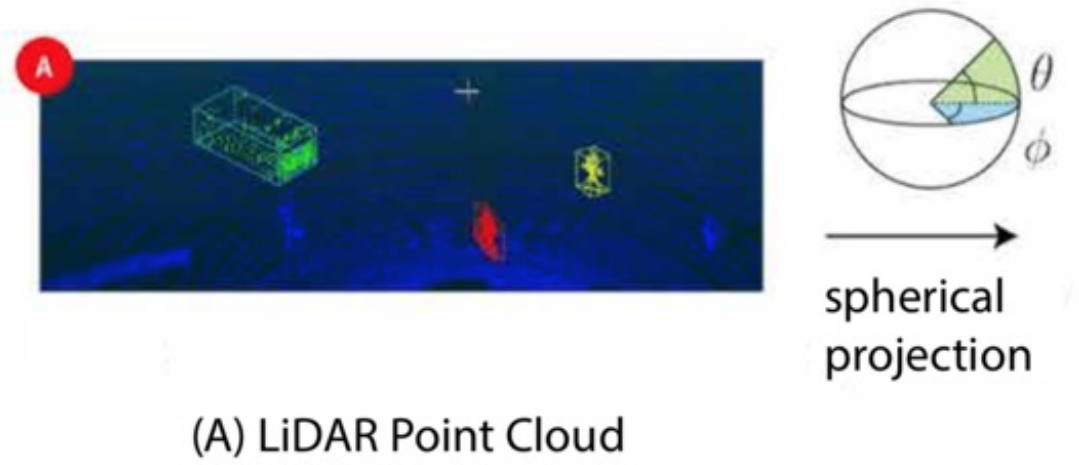
将3D空间离散为立体像素会导致过多的空voxels。处理这样的稀疏数据是低效的，浪费计算。

为了获得更紧凑的表示，我们将激光雷达点云投射到一个球体上，以实现密集的、基于网格的表示：

$$\theta = arcsin \frac{z}{\sqrt{x^2+y^2+z^2}}, \tilde{\theta} = \lfloor \theta / \Delta\theta \rfloor,$$

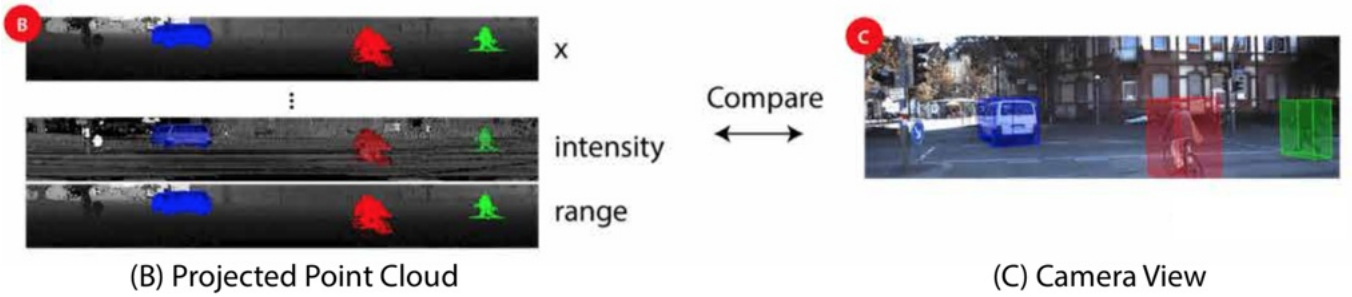
$$\phi = arcsin \frac{z}{\sqrt{x^2+y^2}}, \tilde{\phi} = \lfloor \phi / \Delta\phi \rfloor \qquad (1)$$

ϕ 和 θ分别为方位角和顶角，如图2中A所示。



(A) LiDAR Point Cloud

$\Delta\theta$ 和 $\Delta\phi$ 是离散化的分辨率，($\tilde{\theta}$, $\tilde{\phi}$)表示2D球面网格上的点的位置。将等式（1）应用于云中的每个点，我们可以获得大小为H × W × C的3D张量。在本文中，我们考虑从具有64个垂直通道的Velodyne HDL-64E LiDAR收集的数据，因此H = 64。受KITTI数据集的数据注释的限制，我们只考虑90°的前视图区域并将其划分为512个网格所以W = 512。C是每个点的特征数。在我们的实验中，我们为每个点使用了5个特征：3个笛卡尔坐标（x，y，z），强度测量和范围 $r = \sqrt{x^2+y^2+z^2}$。投影点云的示例可以在图2（B）中找到。可以看出，这种表示是密集且规则地分布的，类似于普通图像（图2（C））。



这种特征使我们能够避免手工制作的功能，从而提高我们的表现形式所概括的几率。

B. 网络结构

我们的卷积神经网络结构如图3所示。

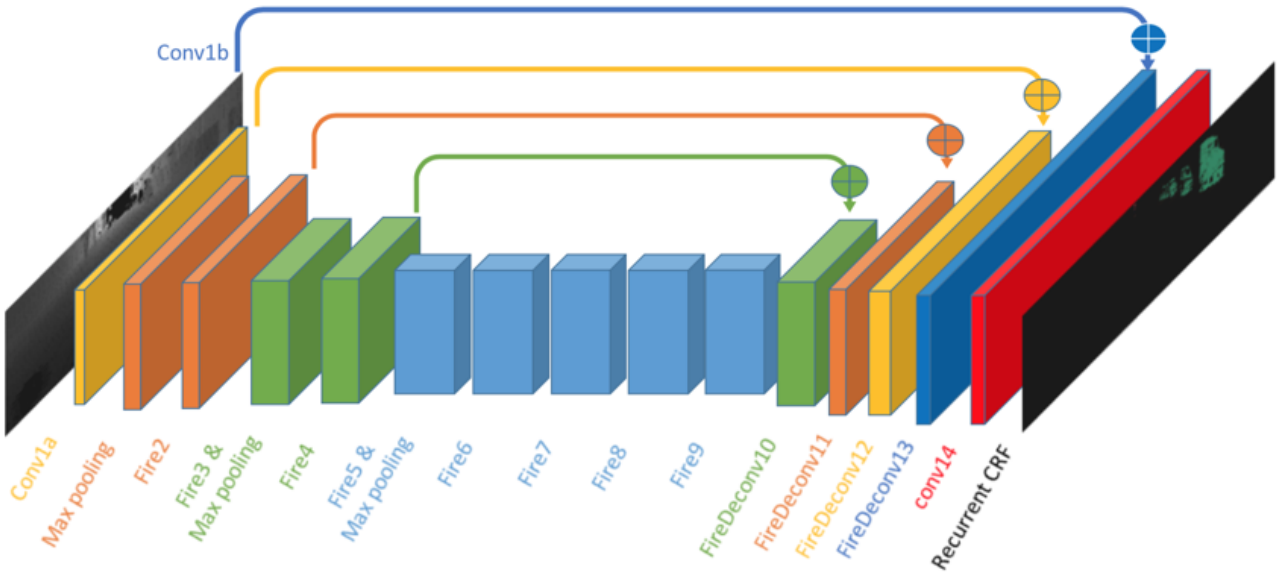


Fig. 3: Network structure of SqueezeSeg.

SqueezeSeg源自SqueezeNet[12]，这是一种轻量级CNN，可以实现AlexNet[21]级精度，参数减少50倍。

SqueezeSeg的输入是64 × 512 × 5张量，如上一节所述。我们从SqueezeNet移植层（conv1a到fire9）以进行特征提取。SqueezeNet使用max-pooling来对宽度和高度尺寸的中间特征图进行下采样，但由于我们的输入张量高度远小于宽度，我们只对宽度进行下采样。fire9的输出是一个下采样的特征映射，它对点云的语义进行编码。

为了获得每个点的全分辨率标签预测，我们使用反卷积模块（更确切地说，“转置卷积”）来对宽度维度中的特征映射进行上采样。 我们使用跳过连接将上采样特征映射添加到相同大小的低级特征映射，如图3所示。输出概率图由具有softmax激活的卷积层（conv14）生成。概

率图由循环CRF层进一步细化，这将在下一节中讨论。
为了减少模型参数和计算的数量，我们用fireModules [12]和fireDeconvs替换了卷积和反卷积层。两个模块的体系结构如图4所示。

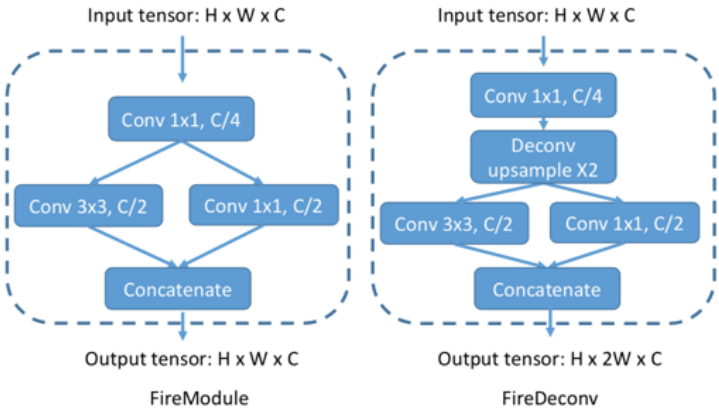


Fig. 4: Structure of a *FireModule* (left) and a *fireDeconv* (right).

在fireModule中，大小为H×W×C的输入张量首先被馈入1x1卷积，以将信道大小减小到C/4。接下来，使用3x3卷积来融合空间信息。与并行1x1卷积一起，它们恢复C的通道大小。输入1x1卷积称为挤压层，并行1x1和3x3卷积合称为扩展层。给定匹配的输入和输出大小，3x3卷积层需要 $9C^2$ 参数以及 $9HWC^2$ 的运算量，而fireModule只需要 $\frac{3}{2}C^2$ 参数和 $\frac{3}{2}HWC^2$ 的计算。在fireDeconv模块中，用于对特征贴图进行上采样的解卷积图层位于挤压和扩展图层之间。~~要将宽度尺寸上采样2~~，常规的1x4反卷积层必须包含 $4C^2$ 参数和 $4HWC^2$ 计算。然而，使用fireDeconv，我们只需要 $\frac{7}{4}C^2$ 参数和 $\frac{7}{4}C^2$ 计算。

C. 条件随机场

通过图像分割，CNN模型预测的标签图往往具有模糊的边界。 这是由于在下采样操作（例如最大池）中丢失了低级细节。 SqueezeSeg中也观察到类似的现象。
准确的逐点标签预测不仅需要了解对象和场景的高级语义，还需要了解低级细节。 后者对于标签分配的一致性至关重要。 例如，如果云中的两个点彼此相邻并且具有相似的强度测量值，则它们可能属于同一对象并因此具有相同的标签。 在10之后，我们使用条件随机场（CRF）来细化由CNN生成的标签图。 对于给定的点云和标签预测c，其中c_i表示第i个点的预测标签，CRF模型使用能量函数：

$$E(c) = \sum_i u_i(c_i) + \sum_{i,j} b_{i,j}(c_i, c_j) \qquad (2)$$

一元多项式 $u_i(c_i) = -logP(c_i)$ 考虑来自CNN分类器的预测概率 $P(c_i)$。二元多项式定义了用于将不同标签分配给一对相似点的“惩罚”，并且被定义为

$$b_{i,j}(c_i, c_j) = \mu(c_i, c_j) \sum_{m=1}^{M} \omega_m k^m(f_i, f_j)$$

其中 $\mu(c_i, c_j) = 1, c_i \neq c_j$且$c_i, c_j \neq 0$ ，k^m是第m个高斯核，其取决于点i和j的特征f，并且 ω_m 是相应的系数。

在我们的工作中，我们使用了两个高斯核：

$$\omega_1 exp(-\frac{|| \mathcal{P}_i - \mathcal{P}_j ||^2}{2\sigma_\alpha^2} - \frac{|| \mathcal{X}_i - \mathcal{X}_j ||^2}{2\sigma_\beta^2}) + \omega_2 exp(-\frac{||\mathcal{P}_i - \mathcal{P}_j||^2}{2\sigma_\gamma^2}). \qquad (3)$$

第一项取决于两个点的角位置 $\mathcal{P}(\tilde{\theta}, \tilde{\phi})$ 和笛卡尔坐标 $X(x, y, z)$。第二项仅取决于角度位置。 σ_α ，σ_β 和 σ_γ 是根据经验选择的三个超参数。还可以包括强度和RGB值等额外功能。

最小化上述CRF能量函数产生精细的标签分配。方程（2）的精确最小化是难以处理的，但9提出了一种平均场迭代算法来近似有效地求解它。 [11]将平均场迭代重新表述为递归神经网络（RNN）。我们将读者引用9和[11]来详细推导平均场迭代算法及其作为RNN的表述。这里，我们仅提供作为RNN模块的平均场迭代的实现的简要描述，如图5所示。

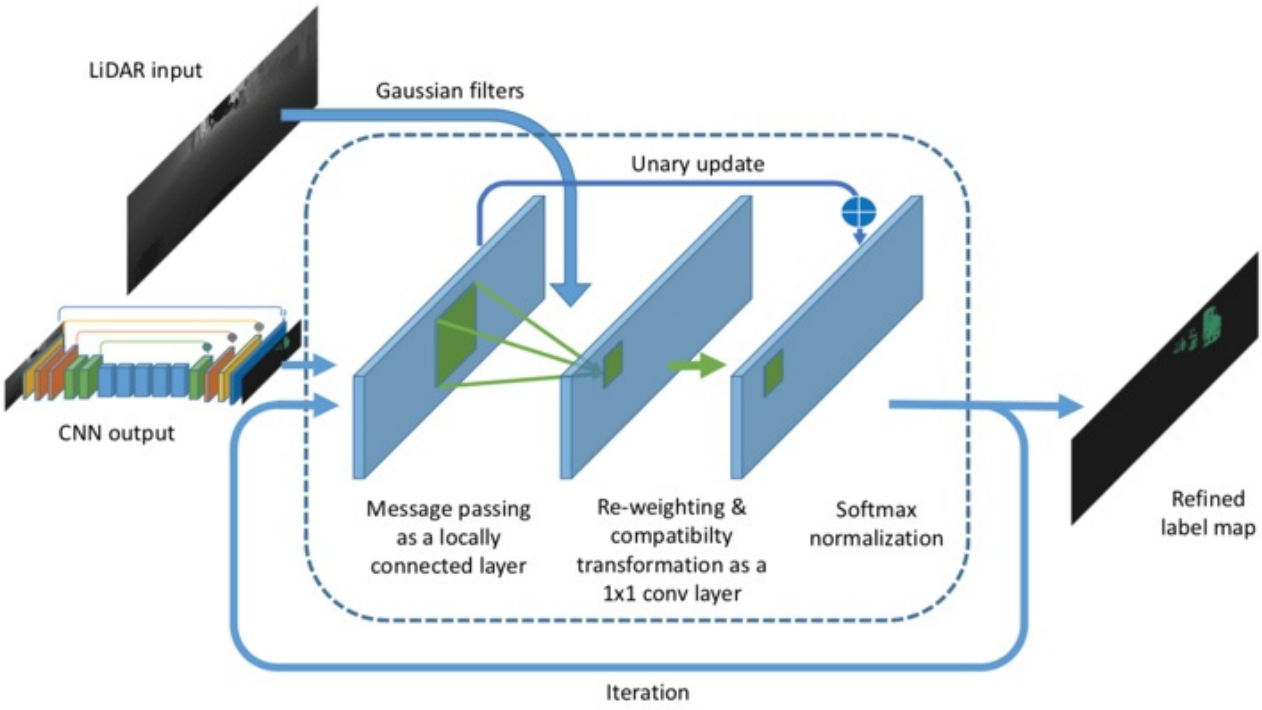


Fig. 5: Conditional Random Field (CRF) as an RNN layer.

CNN模型的输出作为初始概率图被馈送到CRF模块中。接下来，我们基于输入特征计算高斯核，如等式（3）。

最小化上述CRF能量函数产生精细的标签分配。方程（2）的精确最小化是难以处理的，但9提出了一种平均场迭代算法来近似有效地求解它。 [11]将平均场迭代重新表述为递归神经网络（RNN）。我们将读者引用9和[11]来详细推导平均场迭代算法及其作为RNN的表述。这里，我们仅提供作为RNN模块的平均场迭代的实现的简要描述，如图5所示.CNN模型的输出作为初始概率图被馈送到CRF模块中。接下来，我们基于输入特征计算高斯核作为等式（3）。随着两点之间的距离（在3D笛卡尔空间和2D角空间中）增加，上述高斯核的值下降得非常快。因此，对于每个点，我们将内核大小限制为输入张量上的3×5的小区域。接下来，我们使用上面的高斯核过滤初始概率图。这个步骤在[11]中也被称为消息传递，因为它基本上聚合了相邻点的概率。该步骤可以实现为具有上面的Guassian内核作为参数的本地连接层。接下来，我们对聚合概率进行重新加权并使用“兼容性转换”来确定它改变每个点的分布的程度。此步骤可以实现为1x1卷积，其参数在训练期间学习。接下来，我们通过将初始概率添加到1x1卷积的输出来更新初始概率，并使用softmax对其进行标准化。模块的输出是精确的概率图，可以通过迭代地应用该过程来进一步细化。在我们的实验中，我们使用3次迭代来获得准确的标签贴图。这种经常性的CRF模块与CNN模型一起可以端到端地一起训练。通过单阶段管道，我们可

以回避多阶段工作流中存在的传播错误的线索，并相应地利用上下文信息。

D. 数据集

我们的初始数据来自KITTI原始数据集，该数据集提供按顺序组织的图像，LiDAR扫描和3D边界框。 逐点注释从3D边界框转换。 对象的3D边界框内的所有点都被视为目标对象的一部分。 然后，我们为每个点分配相应的标签。这种转换的一个例子可以在图2（A，B）中找到。使用这种方法，我们收集了10,848个带有逐点标签的图像。
为了获得更多的训练样本（点云和逐点标签），我们在GTA-V中构建了一个LiDAR模拟器。模拟器的框架基于DeepGTAV[1]，它使用Script Hook V[2]作为插件。
我们在游戏中的汽车上安装了一个虚拟的LiDAR扫描仪，然后将其设置为自动驾驶。系统收集LiDAR点云和游戏屏幕。在我们的设置中，虚拟激光雷达和游戏摄像机位于相同的位置，这提供了两个优点：首先，我们可以轻松地对收集的数据进行健全性检查，因为点和图像需要保持一致。其次，点和图像可以用于其他研究项目，例如传感器融合等。

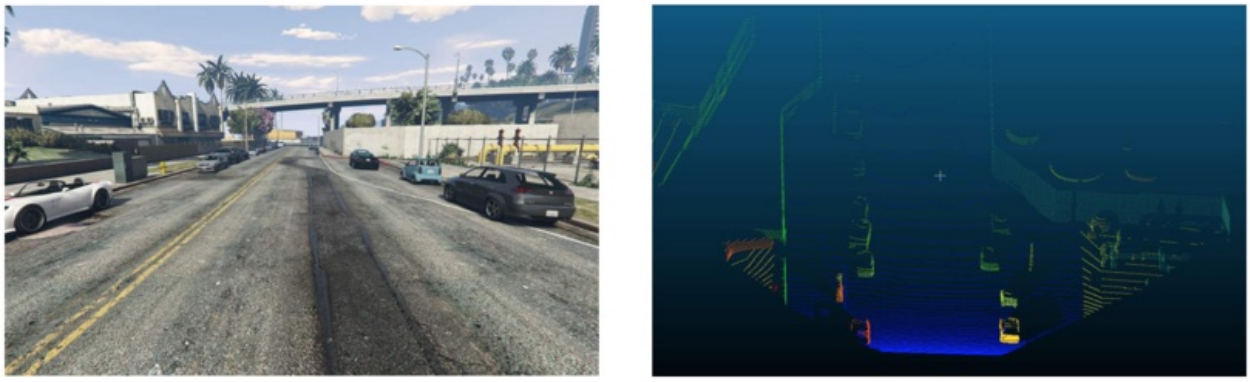


Fig. 6: Left: Image of game scene from GTA-V. Right: LiDAR point cloud corresponding to the game scene.

我们使用光线投射来模拟LiDAR发射的每个激光射线。每个激光射线的方向基于LiDAR设置的几个参数：垂直视场（FOV），垂直分辨率，俯仰角和点云扫描中的射线索引。通过一系列API，可以获得与每条射线相关的以下数据：a）射线命中的第一个点的坐标，b）命中对象的类，c）命中对象的实例ID（即对于实例分割等有用），d）对象命中的中心和边界框。
使用这个模拟器，我们构建了一个包含8,585个样本的合成数据集，大约是我们训练集大小的两倍。 为了使数据更加真实，我们进一步分析了KITTI点云噪声的分布（图7）。

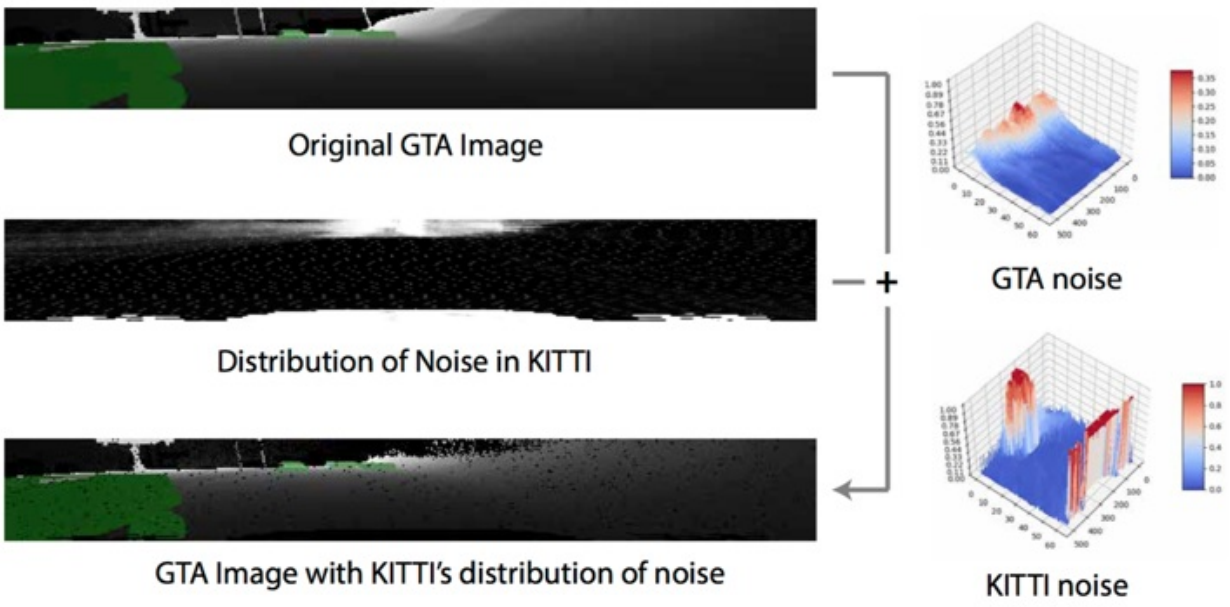


Fig. 7: Fixing distribution of noise in synthesized data

我们在每个径向坐标处获取噪声的经验频率并归一化以获得有效的概率分布：1）设P_i是前面在III-A部分中描述3D张量，表示第i个KITTI点云的球面投影的“像素值”的格式。对于n个KITTI点云中的每一个，考虑（$\tilde{\theta}$，$\tilde{\phi}$）坐标处的像素是否包含“噪声”。为简单起见，我们认为“噪声”是缺失数据，其中所有像素通道都为零。 然后（$\tilde{\theta}$，$\tilde{\phi}$）坐标处的噪声经验频率为：

$$\epsilon(\tilde{\theta},\tilde{\phi}) = \frac{1}{n}\sum_{i=1}^{n}\mathbb{I}_{P_i[\tilde{\theta},\tilde{\phi}]=0}$$

然后我们可以使用KITTI数据中的噪声分布来增加合成数据。 对于合成数据集中的任何点云，在点云的每个（$\tilde{\theta}$，$\tilde{\phi}$）坐标处，我们通过将所有特征值设置为0，以概率ϵ（$\tilde{\theta}$，$\tilde{\phi}$）随机地添加噪声。

值得注意的是，GTA-V为行人使用了非常简单的物理模型，通常会将人员减少到汽缸。此外，GTA-V不为自行车手编写单独的类别，而是在所有账户上单独标记人员和车辆。出于这些原因，我们决定在使用我们的综合数据集进行训练时，专注于KITTI评估的“汽车”类。

4. 实验

A. 评估指标

我们在类级别和实例级别的分割任务中评估模型的性能。对于类级分割，我们将预测与地面实况标签进行逐点比较，并评估精度，recall和IoU（交叉结合）分数，其定义如下：

$$\mathcal{P}_{r_c} = \frac{|\mathcal{P}_c \bigcap \mathcal{G}_c|}{|\mathcal{P}_c|},$$
$$recall_c = \frac{|\mathcal{P}_c \bigcap \mathcal{G}_c|}{|\mathcal{G}_c|},$$
$$IoU_c = \frac{|\mathcal{P}_c \bigcap \mathcal{G}_c|}{|\mathcal{P}_c \bigcup \mathcal{G}_c|}$$

其中$\mathcal{P}_c$和$\mathcal{G}_c$分别表示属于class-c的预测和地面真实点集。$|\cdot|$表示集合的基数。IoU评分用作我们实验中的主要准确度指标。

对于实例级分割，我们首先将每个预测的实例-i与地面实例匹配。该索引匹配过程可以表示为 $\mathcal{M}(i) = j$ ， $i \in \{1, \ldots, N\}$ 是预测实例的索引，$j \in \{\emptyset, 1, \ldots, M\}$ 是地面的真实实例索引。如果没有地面真实与实例i匹配，则 $\mathcal{M}_i$ 为 $\emptyset$，匹配过程 $\mathcal{M}(\cdot)$ ：
1）按地点数量对地面实例进行排序，
2）对于每个地面实例，找到具有最大IoU的预测实例。评估脚本将与源代码一起发布。

对于每个class-c，我们将实例级精度，recall和IoU分数计算为：

$$\mathcal{P}_{r_c} = \frac{\sum_i |\mathcal{P}_{i,c} \bigcap \mathcal{G}_{\mathcal{M}(i),c}|}{|\mathcal{P}_c|},$$
$$recall_c = \frac{\sum_i |\mathcal{P}_{i,c} \bigcap \mathcal{G}_{\mathcal{M}(i),c}|}{|\mathcal{G}_c|},$$
$$IoU_c = \frac{\sum_i |\mathcal{P}_{i,c} \bigcap \mathcal{G}_{\mathcal{M}(i),c}|}{|\mathcal{P}_c \bigcup \mathcal{G}_c|}$$

$\mathcal{P}_{i,c}$ 表示属于class-c的第i个预测实例。不同的实例集互斥，则 $\sum_i |\mathcal{P}_{i,c}| = |\mathcal{P}_c|$。 对于 $\mathcal{G}_{\mathcal{M}_{(i),c}}$ 也同样适用。如果没有地面真实实例与prediction-i匹配，则 $\mathcal{G}_{\mathcal{M}_{(i),c}}$ 就是 $\emptyset$ 。

B. 实验设置

我们的主要数据集是上面描述的转换后的KITTI数据集。 我们将公开可用的原始数据集拆分为具有8,057帧的训练集和具有2,791帧的验证集。 请注意，如果KITTI LiDAR扫描来自同一序

列，则它们可以在时间上相关。 在我们的分割中，我们确保训练集中的帧不出现在验证序列中。 我们的培训/验证分组也将发布。 我们在Tensorflow[22]中训练生成了我们的模型，并使用NVIDIA TITAN X GPU进行实验。 由于KITTI数据集仅为前视LiDAR扫描提供可靠的3D边界框，因此我们将水平视野限制为前向90°。 我们的模型训练协议的详细信息将在我们的源代码中发布。

C. 实验结果

SqueezeSeg的分割精度总结在表1中。

TABLE I: Segmentation Performance of SqueezeSeg

		Class-level			Instance-level		
		P	R	**IoU**	P	R	**IoU**
car	w/ CRF	66.7	95.4	64.6	63.4	90.7	59.5
	w/o CRF	62.7	95.5	60.9	60.0	91.3	56.7
pedestrian	w/ CRF	45.2	29.7	21.8	43.5	28.6	20.8
	w/o CRF	52.9	28.6	22.8	50.8	27.5	21.7
cyclist	w/ CRF	35.7	45.8	25.1	30.4	39.0	20.6
	w/o CRF	35.2	51.1	26.4	30.1	43.7	21.7

Summary of SqueezeSeg's segmentation performance. *P, R, IoU* in the header row respectively represent precision, recall and intersection-over-union. IoU is used as the primary accuracy metric. All the values in this table are in percentages.

我们比较了SqueezeSeg的两种变体，一种是复发CRF层，另一种是没有。尽管我们提出的度量标准非常具有挑战性——因为高IoU需要逐点正确性——SqueezeSeg仍然可以获得高IoU分数，特别是对于汽车类别。请注意，汽车类别的级别级别和实例级别recall都高于90％，这对于自动驾驶来说是理想的，因为假阴性更容易导致事故而不是误报。我们将行人和骑车人类别的较低表现归因于两个原因：1）数据集中的行人和骑车人的实例较少。2）行人和骑车人的实例尺寸要小得多，并且细节更精细，因此更难分割。

通过将我们的CNN与CRF相结合，我们显着提高了汽车类别的准确度（IoU）。性能提升主要来自精度的提高，因为CRF可以更好地过滤边界上错误分类的点。与此同时，我们也注意到CRF导致行人和骑车人分割任务的表现稍差。这可能是由于行人和骑车人缺乏CRF超参数调整。

两个SqueezeSeg模型的运行时间总结在表II中。

TABLE II	Average runtime (ms)	Standard deviation (ms)
SqueezeSeg w/o CRF	8.7	0.5
SqueezeSeg	13.5	0.8
DBSCAN clustering	27.3	45.8

在TITAN X GPU上，没有CRF的SqueezeSeg仅需要8.7 ms来处理一个LiDAR点云帧。结合CRF层，模型每帧需要13.5 ms。这比目前大多数LiDAR扫描仪的采样率要快得多。例如，Velodyne HDL-64E LiDAR的最大旋转速率为20Hz。在计算资源更加有限的车载嵌入式处理器上，SqueezeSeg可以轻松地在速度和其他实际问题（例如能效或处理器成本）之间进行权衡。另请注意，两个SqueezeSeg模型的运行时标准偏差非常小，这对整个自动驾驶系统的稳定性至关重要。然而，我们的实例分割目前依赖于传统的聚类算法，例如DBSCAN3[3]，相比之下，它需要更长的时间并且具有更大的方差。需要更高效和稳定的集群实现，但这超出了本文的范围。

在对GTA模拟数据进行训练时，我们测试了模型在KITTI数据上的准确性——其结果总结在表III中。

TABLE III: Segmentation Performance on the Car Category with Simulated Data

	Class-level			Instance-level		
	P	R	**IoU**	P	R	**IoU**
KITTI	58.9	95.0	57.1	56.1	90.5	53.0
GTA	30.4	86.6	29.0	29.7	84.6	28.2
KITTI + GTA	69.6	92.8	66.0	66.6	88.8	61.4

我们的GTA模拟器目前仍然有限，无法为行人和骑车人提供逼真的标签，因此我们只考虑汽车的分段性能。此外，我们的模拟点云不包含强度测量值;因此，我们将强度在网络的输入特征中排除。为了量化训练对合成数据的影响，我们在KITTI训练集上训练了SqueezeSeg模型，不使用强度测量，并在KITTI验证集上进行了验证。模型的性能显示在表格的第一行。与Table.I相比，由于强度通道的损失，IoU得分更差。如果我们在GTA模拟数据上完全训练模型，我们会发现性能明显更差。然而，将KITTI训练集与我们的GTA模拟数据集相结合，我们发现显着提高的准确度甚至比Table.I更好。

SqueezeSeg与地面实况标签的分割结果可视化可以在图8中看到。

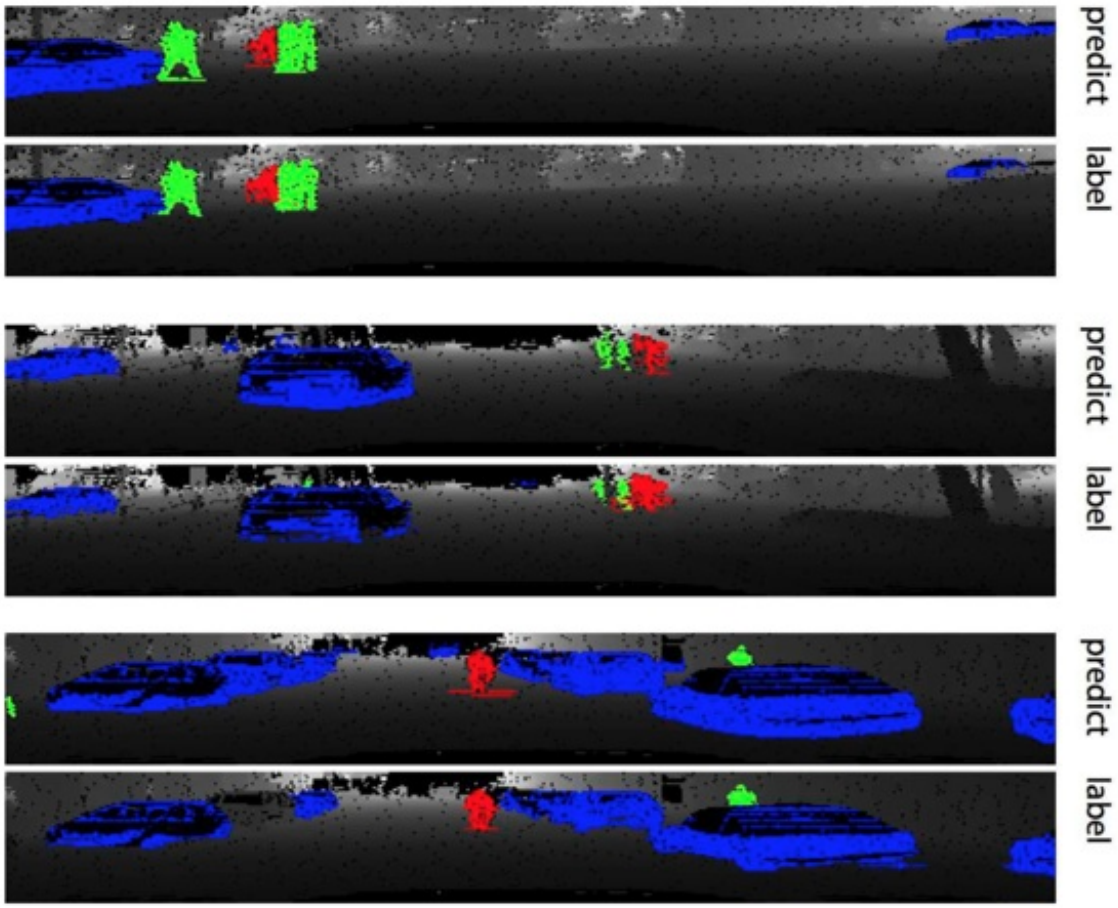


Fig. 8: Visualization of SqueezeSeg's prediction on a projected LiDAR depth map. For comparison, visualization of the ground-truth labels are plotted below the predicted ones. Notice that SqueezeSeg additionally and accurately segments objects that are unlabeled in ground truth.

SqueezeSeg与地面实况标签的分割结果可视化可以在图8中找到。对于大多数物体，预测结果几乎与地面实况相同，除了目标物体下方的地面。另外请注意SqueezeSeg，并准确地分割在地面实况中未标记的对象。这些物体可能被遮挡或太小，因此被置于KITTI基准的“不关心”类别中。

5. 总结

我们提出了SqueezeSeg，一种准确，快速和稳定的端到端方法，用于从LiDAR点云进行道路对象分割。为解决引言中讨论的先前方法的不足，我们的深度学习方法1）不依赖于手工制作的功能，而是利用通过培训学到的卷积滤波器; 2）使用深度神经网络，因此不依赖于迭代算法，如RANSAC，GP-INSAC和凝聚聚类; 3）将管道减少到一个阶段，回避传播错误的问题，并允许模型充分利用对象上下文。该模型以快于实时的推理速度实现非常高的精度，并且具有小的方差，如自动驾驶等应用所需。此外，我们综合了大量的模拟数据，然后在使用合成数据进行训练并验证真实数据时，表现出显着的性能提升。我们使用选择类作为概念验证，授予合成数据在未来自动驾驶数据集中的潜在作用。

致谢

这项工作部分得到了DARPA PER- FECT计划，奖项HR0011-12-2-0016，以及ASPIRE实验室赞助商英特尔，以及实验室附属公司HP，华为，Nvidia和SK海力士的支持。 这项工作也得到了宝马，英特尔和三星全球研究组织的个人礼物的部分赞助。

---

[1] https://github.com/ai-tor/DeepGTAV ↩

[2] http://www.dev-c.com/gtav/scripthookv/ ↩

[3] http://scikit-learn.org/0.15/modeules/generated/sklearn.cluster.DBSCAN.html ↩