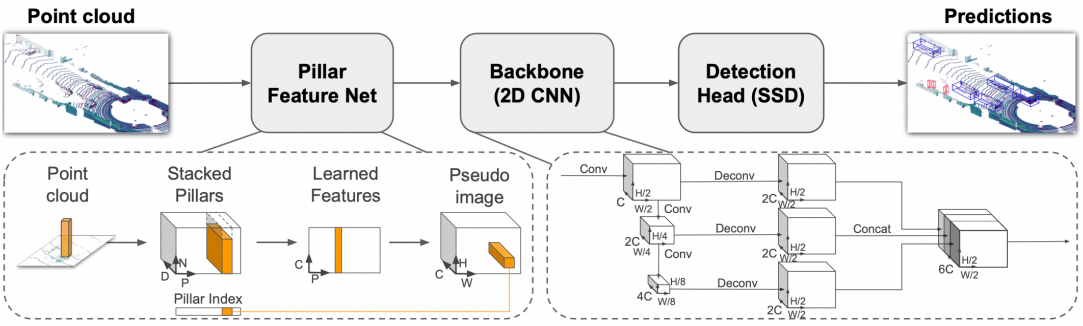
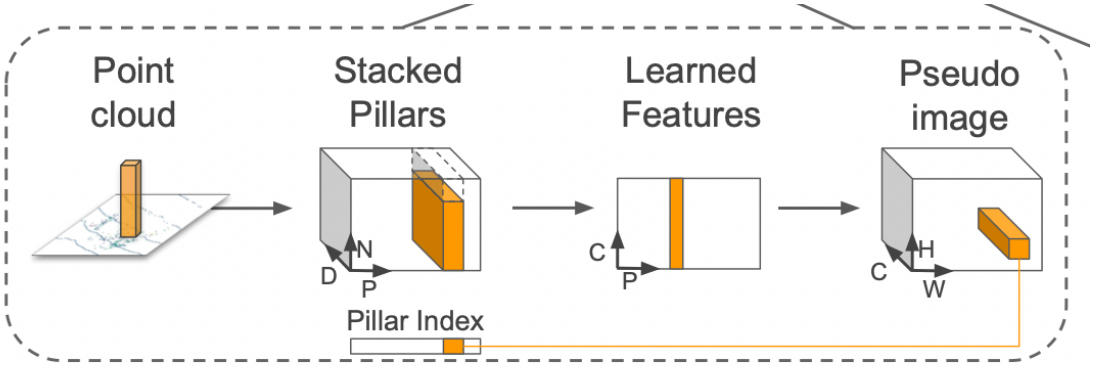
PointPillars: Fast Encoders for Object Detection from Point Clouds





按照点云数据所在的 X，Y 轴（不考虑 Z 轴）将点云数据划分为一个个的网格，凡是落入到一个网格的点云数据被视为其处在一个 pillar 里，或者理解为它们构成了一个 Pillar。

Pointcloud to Pseudo-Image：

点云中一个点表示为 p={x,y,z,r},落在一个网格后，该点表示为 p={x,y,z,r,xc,yc,zc,xp,yp},其中 xc，yc，zc 为该 pillar 内所有点的平均值。xp，yp 为 x-xc，y-yc,是该点对于中心的偏移。

每个 pillar 被固定为 N 个点，多于 N 个点则下采样为 N，少于则用 0 填充。则 pillar 被表示为 p={D,P,N}.D=9，P 为非空 pillar 数量，N 为点数。

接下来对点云的维度进行处理，D 至 C，则 p={C,P,N}.

然后进行 max pooling 操作，（C，P）的特征图。此时将 P 转化为 W*H。此时获得了形如（C，W，H）的伪图片。接着将伪图片作为 2D CNN 的输入，进行类似特征金字塔的特征提取和融合，之后进行目标检测。

BEV perception

方法 1:IPM——逆透视变换（openCV）

采集四个原图中的亚像素点，再定位四个点到变换后的 bev 图。通过计算的到仿射变换矩阵。再通过仿射射变换矩阵计算整张图片的变换。

A Sim2Real Deep Learning Approach for the Transformation of Images from Multiple Vehicle-Mounted Cameras to a Semantically Segmented Image in Bird's Eye View*

（cvpr2020）

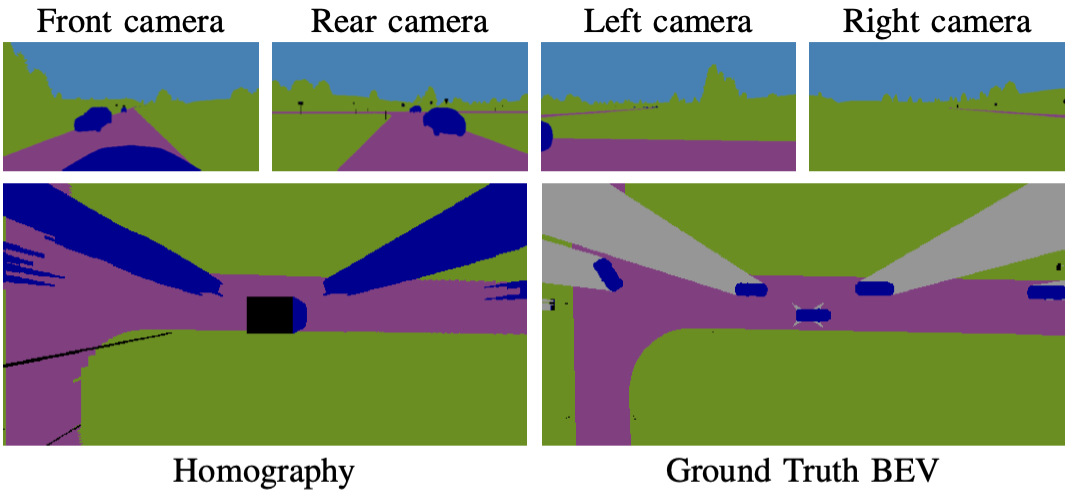
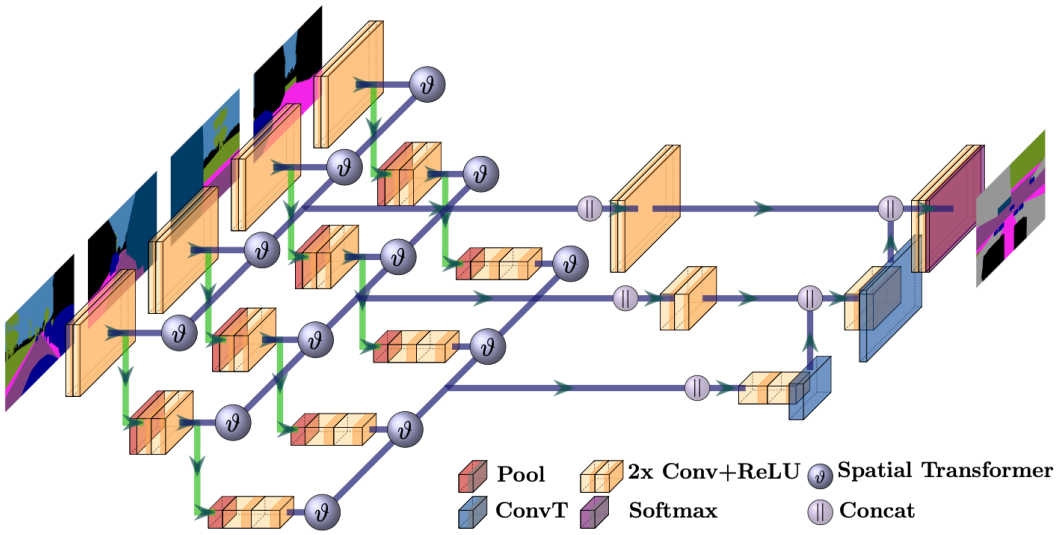
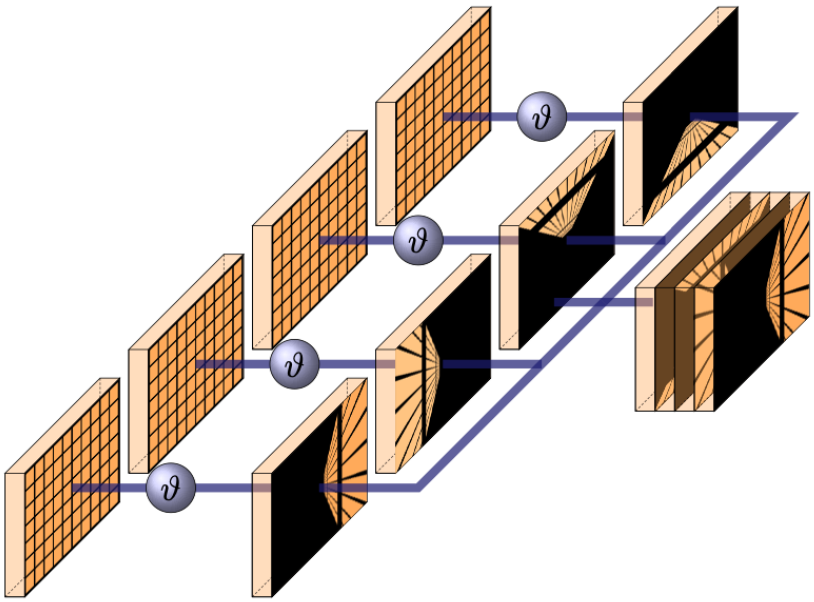


Fig. 1. A homography can be applied to the four semantically segmented images from vehicle-mounted cameras to transform them to BEV. Our approach involves learning to compute an accurate BEV image without visual distortions.



uNetXST 架构为每个输入图像（绿色路径）提供单独的编码器路径。作为每个尺度级别（紫色路径）上跳过连接的一部分，特征图被投影变换（θ-block），与其他输入流（||-block）连接，卷积，最后与上采样输出连接 解码器路径。 此图显示了一个只有两个池化层和两个上采样层的网络，实际训练的网络分别包含四个。



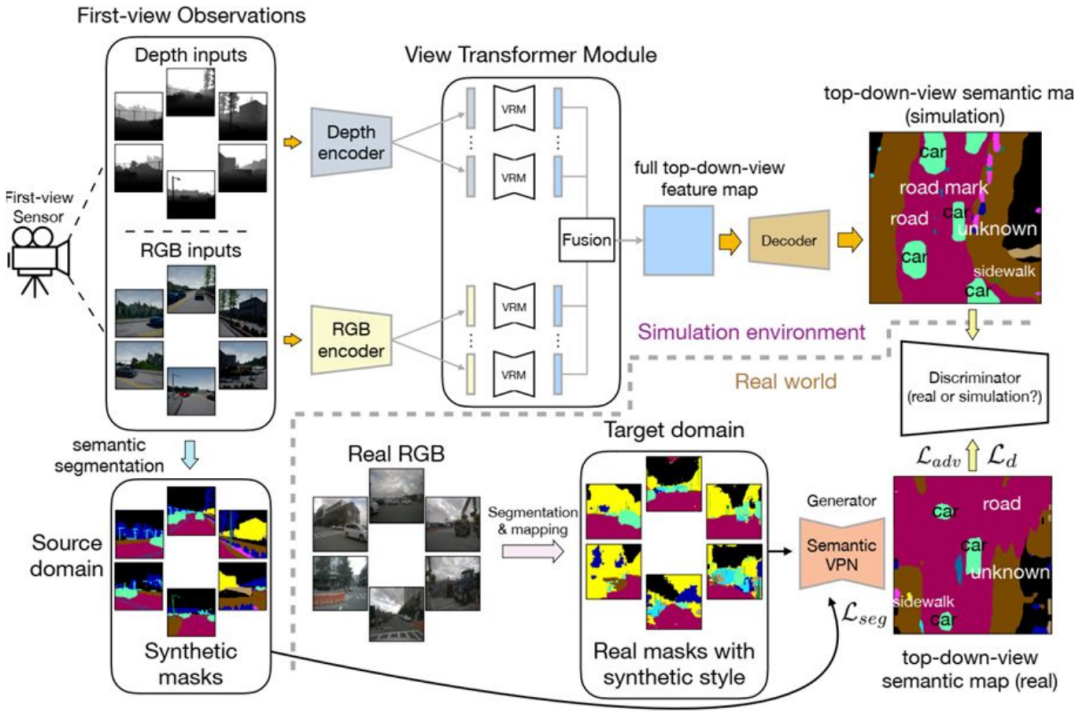
θ 块类似于空间变换器单元。 来自前面卷积层（橙色网格层）的输入特征图由通过 IPM 获得的单应性进行投影变换。 不同相机的输入流之间的转换不同。 建立了空间一致性，因为转换后的特征图都捕获了与地面实况 BEV 相同的视野。 然后将转换后的特征图连接成单个特征图。

方法 2: 学习变换空间

Cross-view Semantic Segmentation for Sensing Surroundings

VPN（Cross-view Semantic Segmentation for Sensing Surroundings）是最早探索 BEV 语义分割的作品之一，将其称为“cross-view 语义分割”。 这篇文章讨论的是跨域视图语义分割，定义一种名为 View Parsing Network （VPN）的框架来解决该问题。在跨视图语义分割任务中，训练后将第一视图观察结果解析为从上到下的

语义图，指出所有目标在像素级的空间位置。此任务的主要问题是缺乏自上而下的视图数据的真实注释。为了缓解这种情况，在 3D 图形环境中训练 VPN，并利用域自适应技术将其迁移到实际数据。



输入由 N 个视角，M 种模态，即 6 个相机，2 个模态（rgb 和 depth）。对 rgb 和 depth，分别进行 cnn 编码，得到 H*W*C 的特征，针对 featuremap 上每个点，学习一个透视空间到 bev 空间的变换 R，即文章中写的

$$f_i[i] = R_i(f[1], \ldots, f[j], \ldots, f[HW])$$

这个类似的实现，在 pointnet 里有类似的操作，即通过 MLP 实现。将 featuremap reshape 到上 HW*C 大小，这样每个点，通过 MLP，在针对 rgb 和 depth 一起 fusion 下，就得到了 BEV 下的特征，后面进行 decode 做分割。