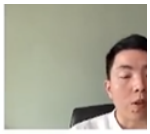
BEV

一 .why bev perception

1. self-driving is a 3D/BEV perception problem
2. 2D to 3D直接人为用2D检测投影到3D，由于较远处像素信息较少，噪声较多，容易失真。



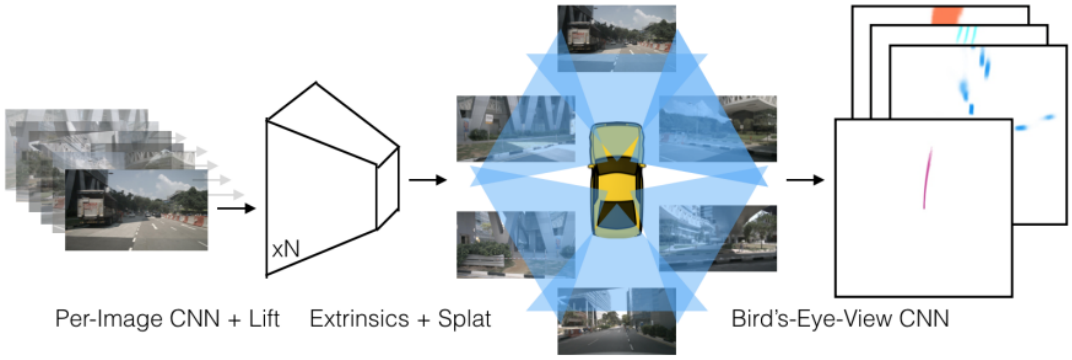
3. make it easy for sensor fusion(多模态融合)

二 .LSS

将单视图的检测扩展到多视图为什么不可行：具体来说，针对来自n个相机的图像数据，我们使用一个单视图检测器，针对每个相机的每张图像数据进行检测，然后将检测结果根据对应相机的内外参数，转换到车辆本体参考下，这样就完成了多视图的检测。

这样简单的后处理无法data-driving，因为上面的转换是单向的，也就是说，我们无法反向区分不同特征的坐标系来源，因此我们无法轻易的使用一个端到端的模式来训练改善我们的自动感知系统

Method



1. **Lift: Latent Depth Distribution**：将2D图像特征生成3D特征

对于每个像素，生成一系列离散的深度值（alpha，alpha为D*1矩阵），在模型训练的时候，由网络自己选择合适的深度。

为什么要对每个像素定义一系列离散的深度值？因为2D图像中的每个像素点可以理解成一条世界中某点到相机中心的一条射线，现在不知道的是该像素具体在射线上位置(也就是不知道该像素的深度值)。本文是在距离相机5m到45m的视锥内，每隔1m有一个模型可选的深度值(这样每个像素有41个可选的离散深度值)

然后将alpha与feature c做外积，得到一个D*C的矩阵，作为per-pixel outer product，下面这个图，由于第三个深度下特征最为显著，因此该位置的深度值为第三个

经过此操作，则估计出了pixel对应的depth信息，将2D图像特征生成3D特征

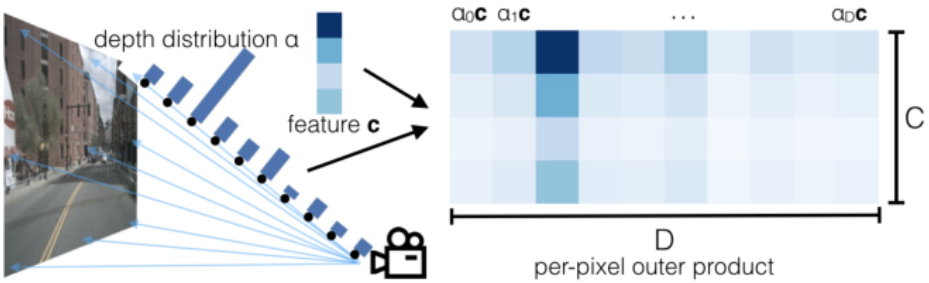


Fig. 3: We visualize the "lift" step of our model. For each pixel, we predict a categorical distribution over depth $\alpha \in \triangle^{D-1}$ (left) and a context vector $\mathbf{c} \in \mathbb{R}^C$ (top left). Features at each point along the ray are determined by the outer product of α and $\mathbf{c}$ (right).

2. **Splat: Pillar Pooling**：将3D特征通过相机内外参，投影到一个统一的坐标系中

   将多个相机中的像素点投影在同一张俯视图中，先过滤掉感兴趣域(以车身为中心200*200范围)外的点。然后需要注意的是，在俯视图中同一个坐标可能存在多个特征，这里有两个原因:1是单张2D图像不同的像素点可能投影在俯视图中的同一个位置,2是不同相机图像中的不同像素点投影在俯视图中的同一个位置，例如不同相机画面中的同一个目标。对于同一个位置的多个特征，作者使用了sum-pooling的方法计算新的特征，最后得到了200x200xC的feature

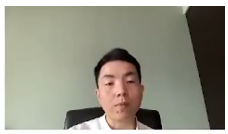
   最后接个一个BevEncode的模块将200x200xC的特征生成200x200x1的特征用于loss的计算

Conclusion

纯视觉bev检测的难点主要在于单目深度估计困难，本文提供了一种解决方法，但是由于对每一个pixel生成D*C的feature，导致计算量较大。

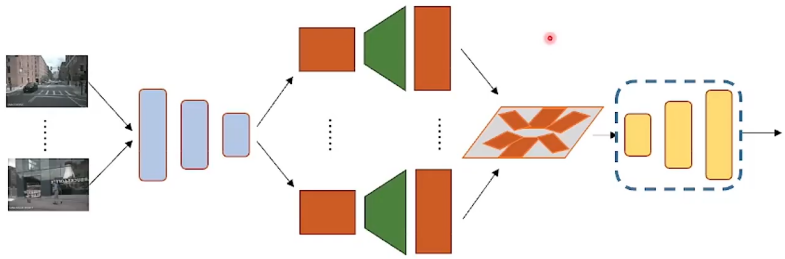
三 .HDMapNet

赵行老师视频中提到，但我并未细看，后续可能完善

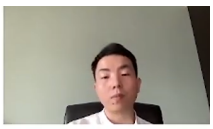


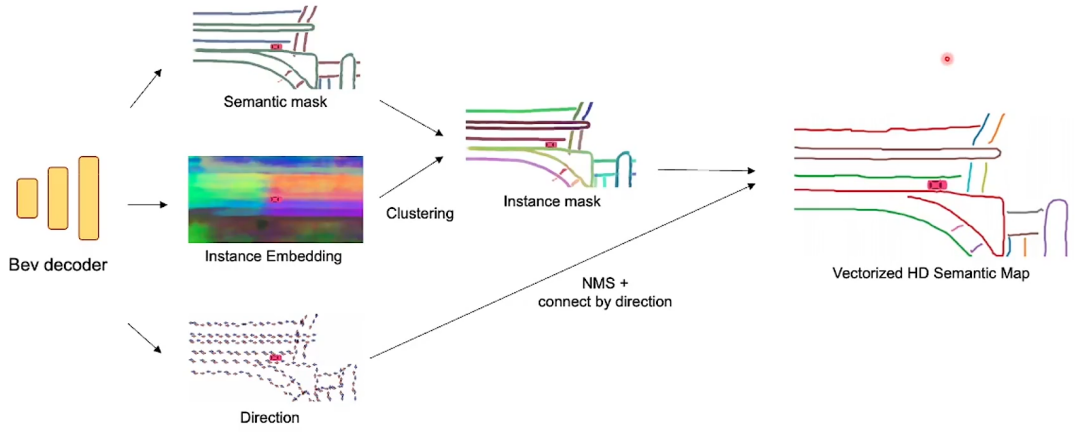
Model Architecture

1. Cam-view encoder
2. View transformation: implicit with NN
3. View projection: explicit with camera configurations
4. BEV decoder



Vectorization





四 .DETR

首先回顾一下attention机制，attention可以看作是value的加权和，而value的权重是由对应的key与query的相似度决定的。

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

使用transformer，取代了现在模型需要手工设计的部分，例如NMS和anchor generation，真正做到了END TO END

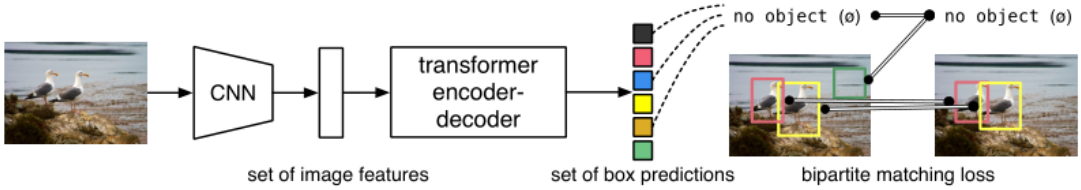


Fig. 1: DETR directly predicts (in parallel) the final set of detections by combining a common CNN with a transformer architecture. During training, bipartite matching uniquely assigns predictions with ground truth boxes. Prediction with no match should yield a "`no object`" ($\varnothing$) class prediction.

Object detection set prediction loss

DETR最后输出N个box，N为固定值，本文设定N=100，再将N个框与gt做二分图匹配，从而得到真实的预测框，再算loss，舍弃了NMS，做到了postprocessing-free

二分图匹配问题采用匈牙利算法解算，cost matrix中填入L_{match}

the predicted box as $\hat{b}_{\sigma(i)}$. With these notations we define $\mathcal{L}_{\text{match}}(y_i, \hat{y}_{\sigma(i)})$ as $-\mathbb{1}_{\{c_i \neq \varnothing\}} \hat{p}_{\sigma(i)}(c_i) + \mathbb{1}_{\{c_i \neq \varnothing\}} \mathcal{L}_{\text{box}}(b_i, \hat{b}_{\sigma(i)})$.

DETR architecture

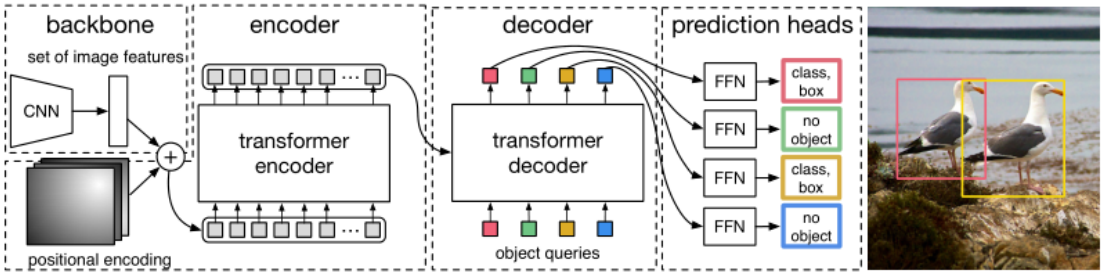
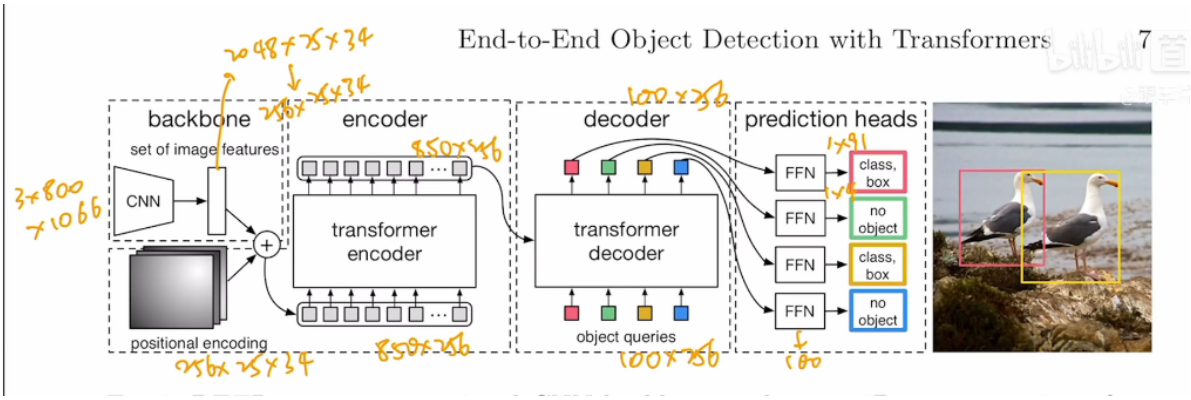


Fig. 2: DETR uses a conventional CNN backbone to learn a 2D representation of an input image. The model flattens it and supplements it with a positional encoding before passing it into a transformer encoder. A transformer decoder then takes as input a small fixed number of learned positional embeddings, which we call *object queries*, and additionally attends to the encoder output. We pass each output embedding of the decoder to a shared feed forward network (FFN) that predicts either a detection (class and bounding box) or a "`no object`" class.



1. CNN+positional encoding

   先用cnn提取图像特征，因为要送入transformer中，需要位置编码，所以进行positional encoding

2. encoder

   类似vit

3. decoder

   加入了object queries，此为一个learnable positional embedding，decoder的输入是object queries（100*256，分别对应之前的256和最后输出的100个框），另一个输入是encoder的输出；最后得到一个100 * 256的输出，本质上是一个cross-attention

4. prediction heads

   FFN：feed forward network，就是一些mlp层，输出最终特征，供后续做二分图匹配和算loss

五 .Deformable DETR

DETR的缺点：训练困难，较难收敛；小目标检测效果不理想；参数量较大

造成以上缺点的原因可能是：在初始化时，attention模块对特征图中的所有像素点施加几乎一致的注意权值，所以需要较多epoch的训练来将注意力focus到稀疏的有意义的区域。

Deformable DETR结合了Deformable convolution(可变形卷积)的优秀的稀疏空间采样能力和transformer的关系建模能力

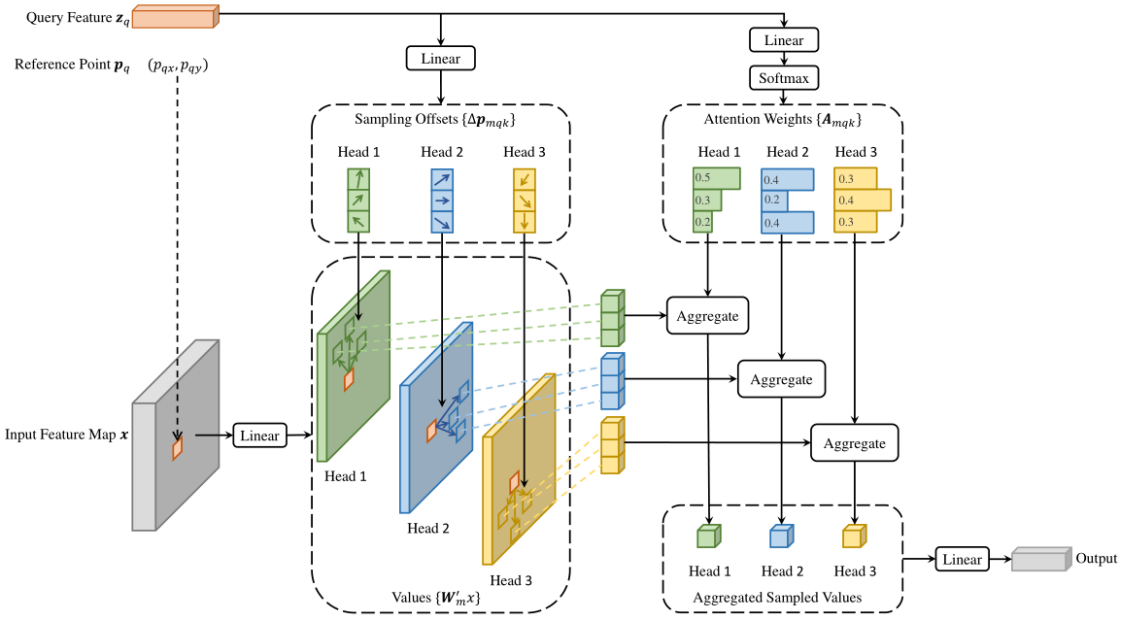
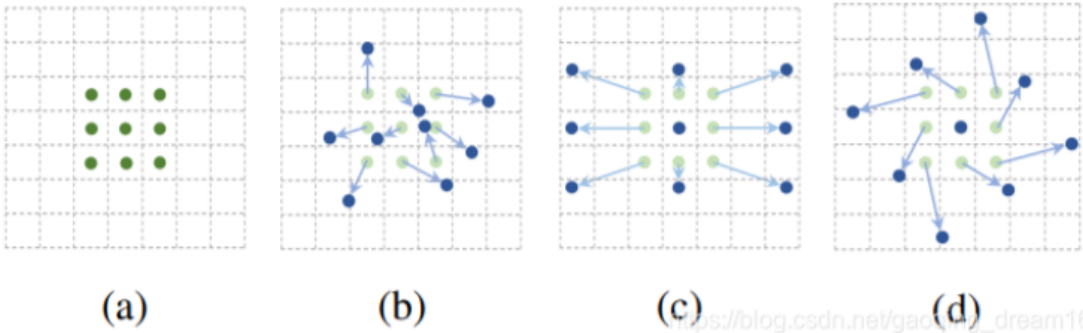


Figure 2: Illustration of the proposed deformable attention module.

Deformable Convolution（可变形卷积）



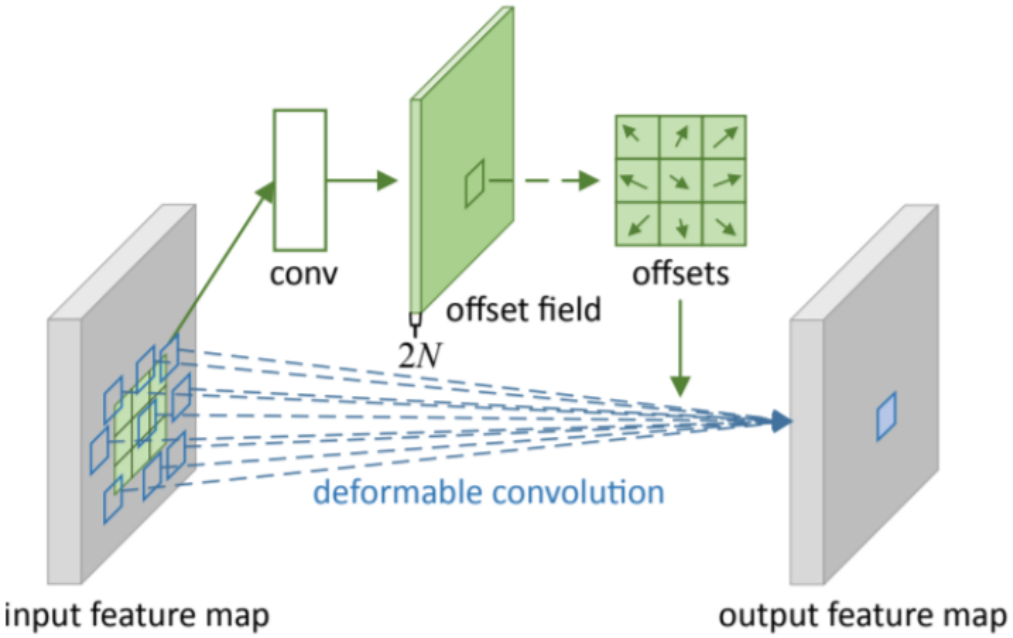


Figure 2: Illustration of 3×3 deformable convolution.

offset是一个learnable的参数，可根据图像内容发生自适应的变化，从而适应不同物体的形状、大小等几何形变，可以高效的处理稀疏空间。

Deformable Attention Module

将transformer运用到cv中一个很大的问题就是他的query会与所有key相关（key对应空间位置，一些空间位置可能不与该query相关），导致计算量很大，Deformable Attention Module将只关注参考点周围的一小部分key采样点，为每个query分配少量固定数量的key

先回顾一下基本的**multihead attention**计算方法

$$\text{MultiHeadAttn}(z_q, x) = \sum_{m=1}^{M} W_m \Big[\sum_{k \in \Omega_k} A_{mqk} \cdot W'_m x_k \Big],$$

其中，zq看作query，由x经过线性变换生成，q是对应的索引，k是key的索引，Omega k即所有的k集合，m代表是第几个注意力头部，Wm是对注意力施加在value后的结果进行线性变换从而得到不同头部的输出结果，W'm 用于将xk变换成value，Amqk代表归一化的注意力权重。即所有的query都要与所有位置的key计算注意力权重，并且对应施加在所有的value上

Deformable Attention:

$$\text{DeformAttn}(z_q, p_q, x) = \sum_{m=1}^{M} W_m \Big[\sum_{k=1}^{K} A_{mqk} \cdot W'_m x(p_q + \Delta p_{mqk}) \Big],$$

比multihead attention多了p_q和delta_P_{mqk}，p_q可理解为代表zq的位置（理解成坐标即可），是2d向量，作者称其为reference points，而delta_P_{mqk}是实际采样点相对于参考点的offset，是一个learnable的参数。

实际操作过程中，delta_P_{mqk}和注意力权重都由query经过一个3MK channel的全连接层算出，前2MK个channel计算delta_P_{mqk}，最后MK个channel经过softmax作为Amqk，即注意力权重。此处与传统transformer不同，注意力权重不是通过query和key的相关性计算的！

Multi-scale Deformable Attention Module

$$\mathrm{MSDeformAttn}(\boldsymbol{z}_q, \hat{\boldsymbol{p}}_q, \{\boldsymbol{x}^l\}_{l=1}^{L}) = \sum_{m=1}^{M} \boldsymbol{W}_m \Big[\sum_{l=1}^{L}\sum_{k=1}^{K} A_{mlqk} \cdot \boldsymbol{W}'_m \boldsymbol{x}^l(\phi_l(\hat{\boldsymbol{p}}_q) + \Delta \boldsymbol{p}_{mlqk})\Big], \quad (3)$$

将可变形注意力模块运用到多尺度上，每个query在每个特征层都会采样K个点，共有L层特征，从而在每个头部内共采样LK个点，注意力权重也是在这LK个点之间进行归一化，因为cross-attention以及可以建立不同特征层之间的关系，所以不用FPN结构。

Deformable Transformer Encoder & Decoder

Encoder

主要是用Deformable Attention替换了DETR中Encoder中的self-attention，由之前的对于一个query，每一层采集K个点作为keys，转换成，对一个query，所有层均采K个点作为keys

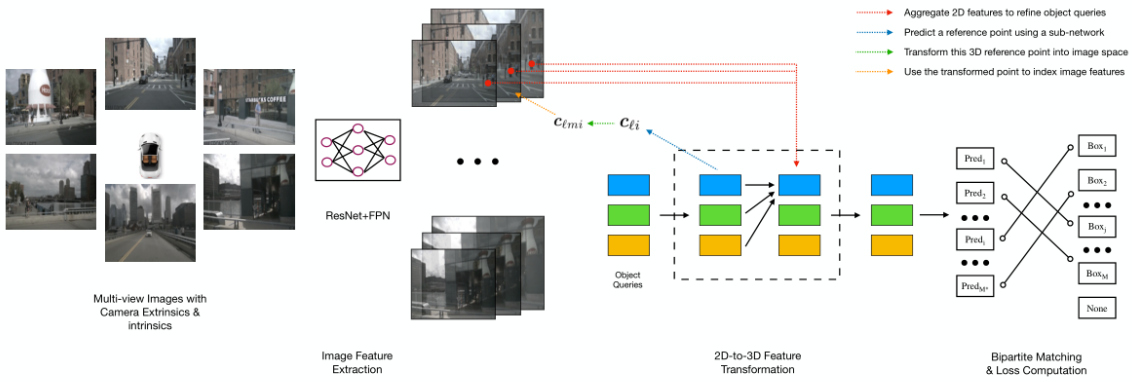
Decoder

self-attention+cross-attention

六 .DETR3D

首先是介绍了一种伪激光雷达的3D物体检测方案，即对一张2D的图片进行深度的预测，得到一个类似于激光雷达的点云图，运用一些点云的检测方法进行3D检测，这种方法具有一定的问题

1. 此方法为two-stage：深度预测+liadr-based 3D Detection，很难得到精确的深度估计，深度预测带来的误差影响后面
2. 像素级别的室外场景深度真值较难获取，一般深度真值都是用lidar获取，但lidar的采样点达不到像素级别的稠密，所以在2D图像上很多像素点是没有真值深度的



输入为Multi-view images，经过Resnet+FPN提取特征（没有transformer encoder模块）

Detection Head：

整个流程可简述为：

1. 生成object queries以及对应的3D参考点

2. 将3D参考点通过相机内外参映射到对应的2D图像上
3. object queries直接和相应的3D参考点投影至2D处位置的特征（key）进行融合（经过双线性插值）

一般的网络：在单张图像中检测密集的bbox，过滤掉冗余的bbox，再在多摄像头中融合（自底向上的方法 bottom-up approach），此类方法的缺点在于：

1. 稠密的bbox预测需要精确的深度信息
2. NMS等操作开销较大

Ours：使用L个layer进行bbox的计算，每个layer生成m个object queries，再将object queries经过线性映射得到3D的参考点C_{li}

$$\boldsymbol{c}_{\ell i} = \Phi^{\mathrm{ref}}(\boldsymbol{q}_{\ell i}),$$

C_{li}可以认为是第i个bbox的中心点的假设（C_{li}属于R^3，为3D点）

现在已经生成了3D的参考点，接下来该将3D的参考点与2D的特征做交互，文中使用内外参的先验信息，将3D reference points投影到各个视角的图片上。由于多相机之间存在共视区域和盲区问题，一个参考点可能投影到多个视角，也可能一个视角也投不到，所以作者加了一个二进制的mask代表当前视角是否被投影成功

$$\boldsymbol{c}^*_{\ell i} = \boldsymbol{c}_{\ell i} \oplus 1 \qquad \boldsymbol{c}_{\ell m i} = T_m \boldsymbol{c}^*_{\ell i}, \tag{2}$$

where $\oplus$ denotes concatenation, and $\boldsymbol{c}_{\ell m i}$ is the projection of the reference point onto the m-th camera. To remove the effects of the feature map size and gather features across different levels, we normalize $\boldsymbol{c}_{\ell m i}$ to $[-1, 1]$. Next, the images features are collected by

$$\boldsymbol{f}_{\ell k m i} = f^{\mathrm{bilinear}}(\mathcal{F}_{k m}, \boldsymbol{c}_{\ell m i}), \tag{3}$$

where $\boldsymbol{f}_{\ell k m i}$ is the feature for i-th point from k-th level of m-th camera at ℓ-th layer.

得到了3D参考点在2D平面的投影，下一步是将object queries与该投影处（该投影即为采样点）的特征通过双线性插值的方法进行融合。

此处DETR3D是直接融合，与DETR中object queries与全图交互，和Deformable DETR中先根据object queries预测一些参考点，再预测一些以参考点为基准的采样点，然后和采样点的特征交互的方法均不同。

Conclusion

object queries是预先生成的，通过一个可学习的线性映射生成3D参考点，3D参考点通过相机内外参投影到2D上，投影的点即为最终的采样点，也即作为key和value，再进行cross-attention，最后算loss。

稍微对比一下LSS和DETR3D可以知道，他们的思路是截然相反的，核心思想都是解决单目相机深度未知的问题：

LSS：2D to 3D，既然深度未知，那就预测深度

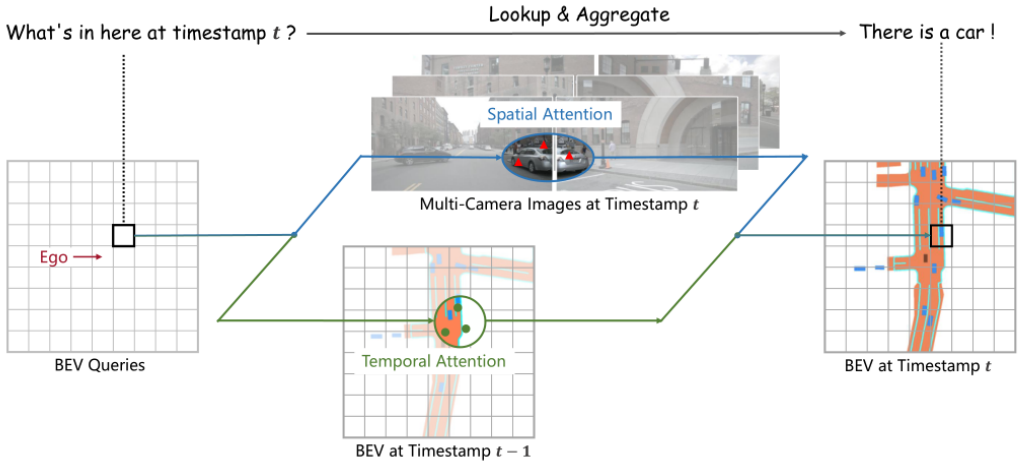
DETR3D：3D to 2D，将3D参考点（由object queries生成）投影到2D平面，再提取特征

七 .BEVFormer

Why bev

1. bev更适合融合多摄像头信息
2. bev是连接时间和空间信息的桥梁（这也是本文的核心）

Key designs



主要由三部分组成：

1. grid-shaped BEV queries
2. 用于融合空间上多相机信息的cross-attention
3. 用于融合历史BEV特征的self-attention module

Architecture

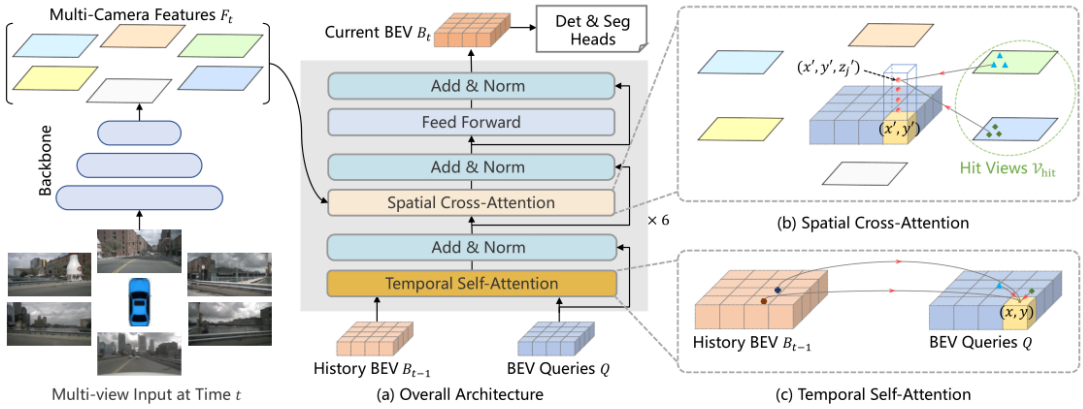
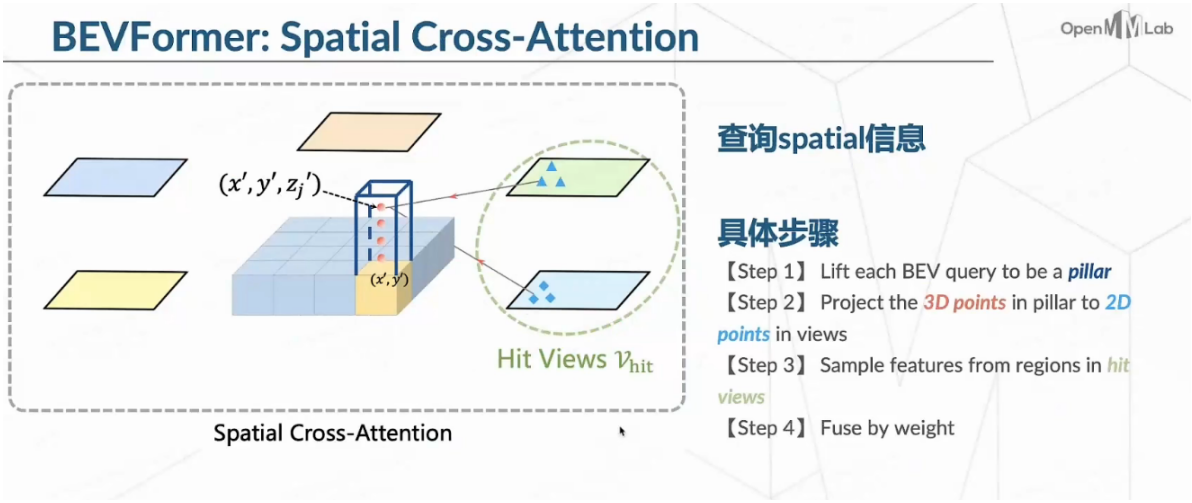


Figure 2: **Overall architecture of BEVFormer.** (a) The encoder layer of BEVFormer contains grid-shaped BEV queries, temporal self-attention, and spatial cross-attention. (b) In spatial cross-attention, each BEV query only interacts with image features in the regions of interest. (c) In temporal self-attention, each BEV query interacts with two features: the BEV queries at the current timestamp and the BEV features at the previous timestamp.

BEV queries：为H * W * C，其中HW为bev平面的高和宽（不是图像的）

Spatial Cross-Attention：



首先将2D的bev queries升维成3D（bev queries是栅格化的queries，其实这个queries是与真实世界的俯视图对应的，而某个坐标的query就负责该栅格区域对应的真实世界俯视图的区域，将俯视图升维就还原到了3D空间点）

再将3D的reference point 通过相机参数投影到2D平面（image features），得到2D的参考点，此2D参考点所在的平面被称为Vhit，2D参考点位置的特征作为value和key，此image features上的2D参考点对应的2D bev queries作为query（有点小绕，这里有两个平面：bev queries平面，image features平面；两个3D空间：真实世界空间，拉伸后的3D bev queries空间，其实都是一 一对应的），然后就类似于DETR3D，使用稀疏的Deformable attention进行cross-attention

$$\text{SCA}(Q_p, F_t) = \frac{1}{|\mathcal{V}_{\text{hit}}|} \sum_{i \in \mathcal{V}_{\text{hit}}} \sum_{j=1}^{N_{\text{ref}}} \text{DeformAttn}(Q_p, \mathcal{P}(p, i, j), F_t^i),$$

Temporal Self-Attention

BEV features：Bt，Bt-1

BEV queries：Q

首先将Bt-1和Q利用自身车辆的运动信息对齐，使同一网格上的特征对应于相同的现实位置，我个人的理解是Bt-1的坐标原点是（0，0），而经过t时间的车辆自身运动之后，此时的Bt-1相对于Q来说，相当于坐标原点往后移了，所以需要对齐。

$$\text{TSA}(Q_p, \{Q, B'_{t-1}\}) = \sum_{V \in \{Q, B'_{t-1}\}} \text{DeformAttn}(Q_p, p, V), \tag{5}$$

TSA模块的输入query是reference point对应的learnable BEV feature，输入V是初始BEV query（每个序列的第一个样本）或者是使用ego-motion与当前BEV query进行对齐后的History BEV

这里有个小细节：此处Deformable attention的offset是由BEV query和History BEV先concat，再经过linear预测得到的。（原始DeformAttn中的offest是直接将query经过一个Linear计算得到的）

Conclusion

主要贡献点是融合了时序信息，对物体移动速度预测提升较高，并且对于一些被遮挡物体的识别提升较高（这一帧被遮挡，之前帧可能没有），recall提高