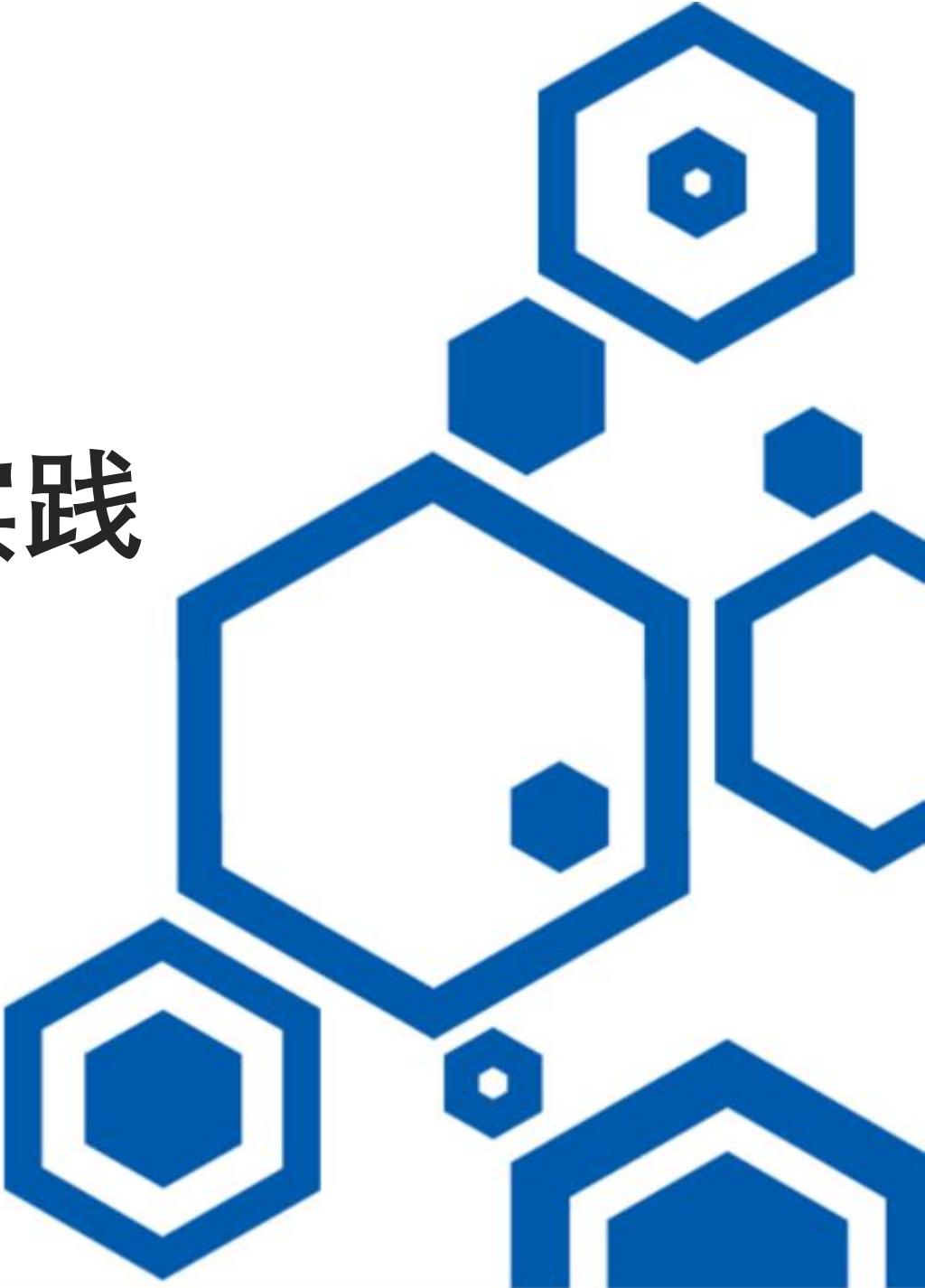


BEV感知系统理论与实践

Dongxu Fu





第二章：从 Multi-View 到 BEV，构建新的感知模型

- ◆ 为什么是BEV (Bird Eye View) ?
- ◆ View Transformation方法
 - ◆ IPM: 最传统的方法
 - ◆ LSS (2D to 3D/从底向上) : 基于深度分布估计的方法
 - ◆ Transformer (3D to 2D/从顶向下) : 基于注意力机制的方法
- ◆ BEV动静态元素检测任务 (含Temporal Fusion)
 - ◆ 基于LSS的动静态元素检测任务
 - ◆ 基于Transformer的动静态元素检测任务



第二章：从 Multi-View 到 BEV，构建新的感知模型

- ❖ 为什么是BEV (Bird Eye View) ?

- ❖ View Transformation方法

- ❖ IPM：最传统的方法
- ❖ LSS (2D to 3D/从底向上)：基于深度分布估计的方法
- ❖ Transformer (3D to 2D/从顶向下)：基于注意力机制的方法

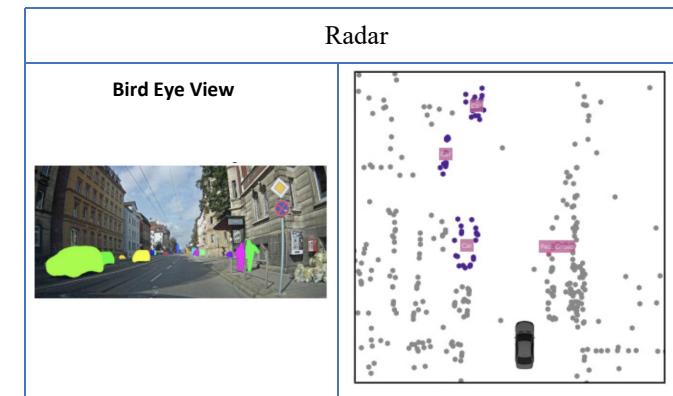
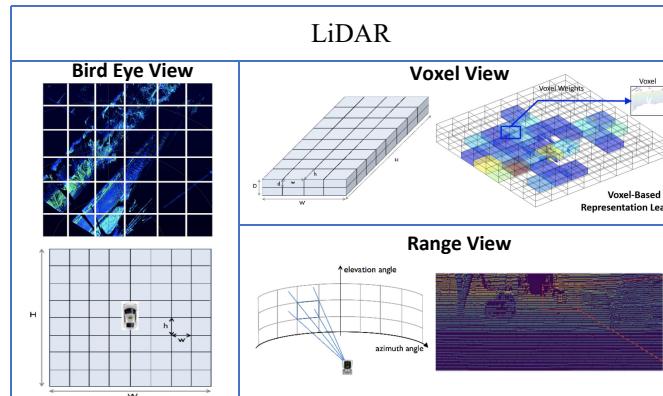
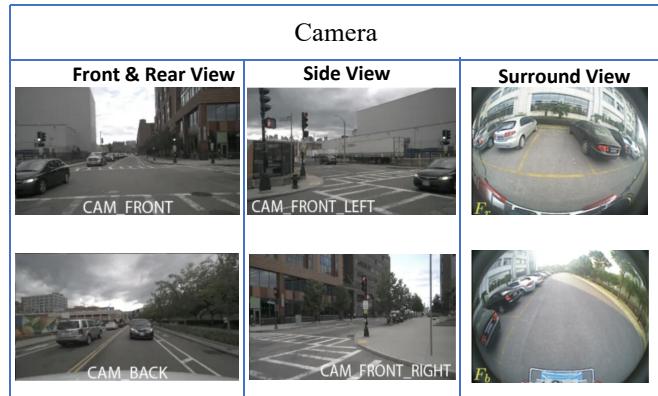
- ❖ BEV动静态元素检测任务 (含Temporal Fusion)

- ❖ 基于LSS的动静态元素检测任务
- ❖ 基于Transformer的动静态元素检测任务

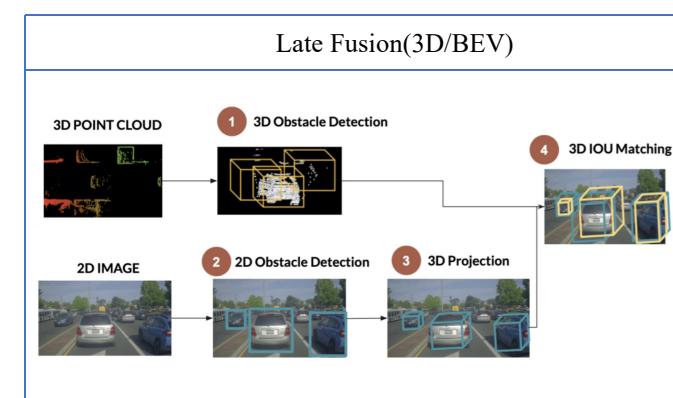
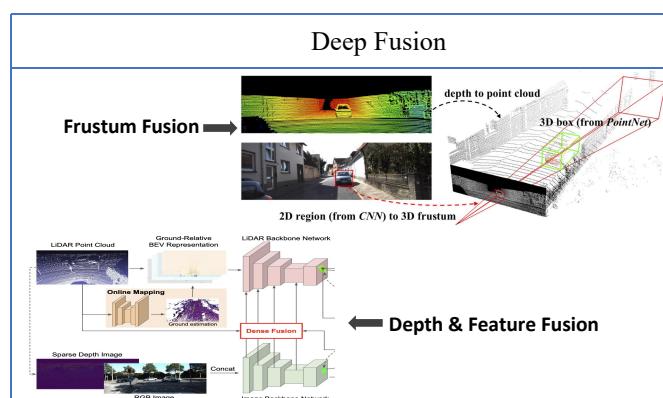
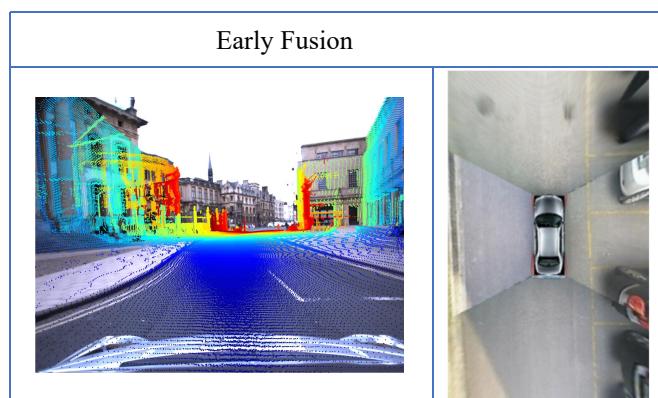


为什么BEV是更合适的特征空间?

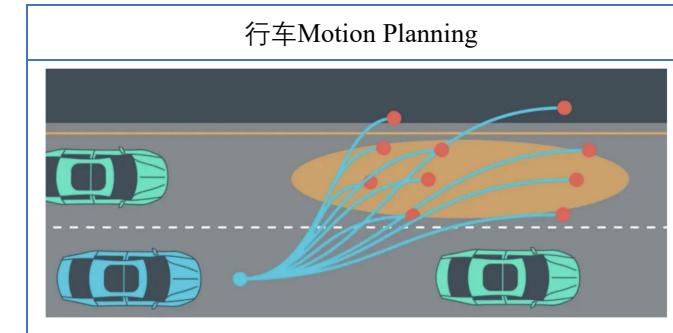
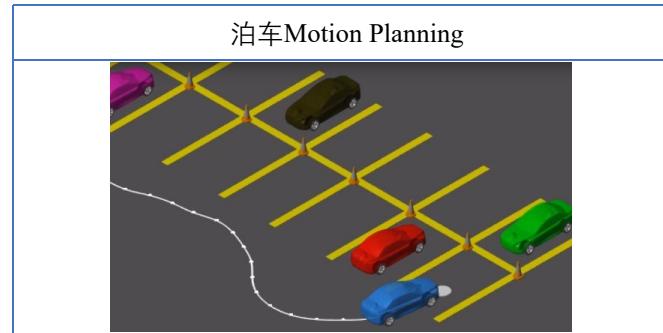
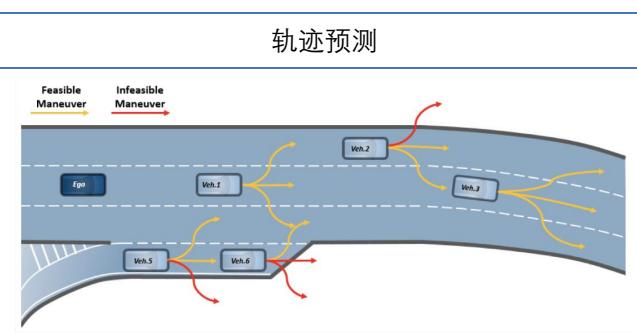
■ Sensor Views



■ Perception Views



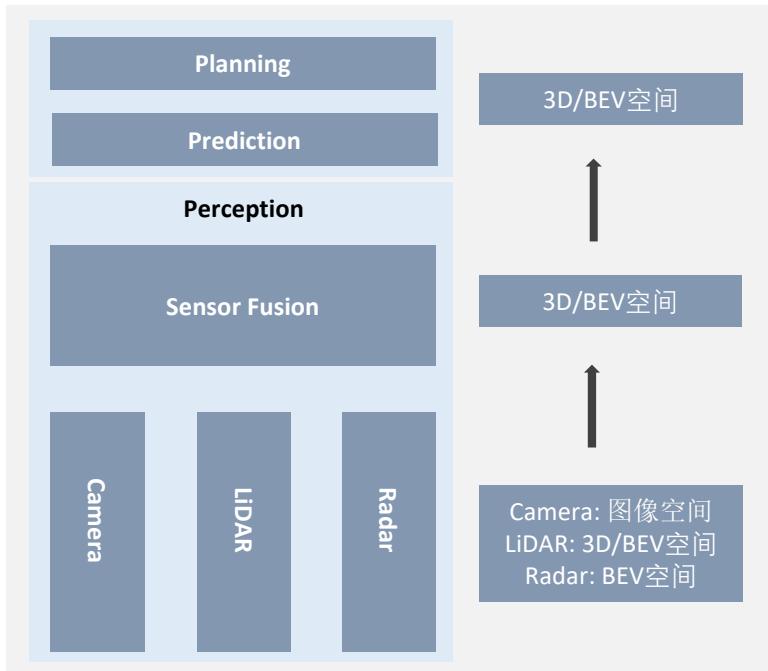
■ Prediction View (3D/BEV)



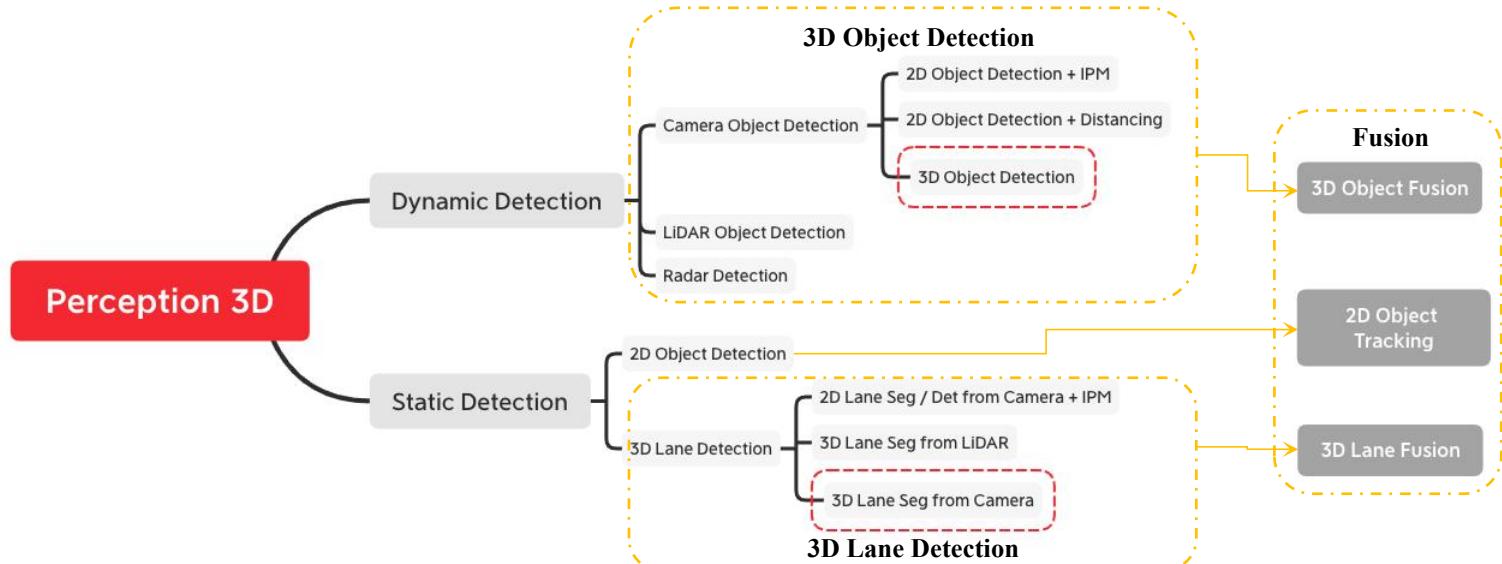


为什么BEV是更合适的特征空间?

□ 感知求解空间



□ 感知问题: 从多空间特征还原3D/BEV元素



□ 问题分析

- 求解空间隔离
 - 任务间优化目标不一致, A模块迭代提升, 但系统性能下降
 - 场景还原(特别是测距)一致性差, 需要显式设计rules做权衡, 泛化能力弱
- 模态间信息稠密度差异大:
 - 目标越远, 距离估计能力衰减程度不同, 后融合需要rules, 泛化能力弱
 - 多传感器融合性能产生瓶颈: Early fusion和Deep Fusion进入瓶颈期

□ 求解思路:

- 为什么是BEV空间, 而不是3D空间?
- 为什么是BEV空间, 而不是Vector Space?
- 最重要的问题: **如何从图像特征提取BEV特征?**



第二章：从 Multi-View 到 BEV，构建新的感知模型

- ❖ 为什么是BEV (Bird Eye View) ?

- ❖ View Transformation方法

- ❖ IPM：最传统的方法
- ❖ LSS (2D to 3D/从底向上)：基于深度分布估计的方法
- ❖ Transformer (3D to 2D/从顶向下)：基于注意力机制的方法

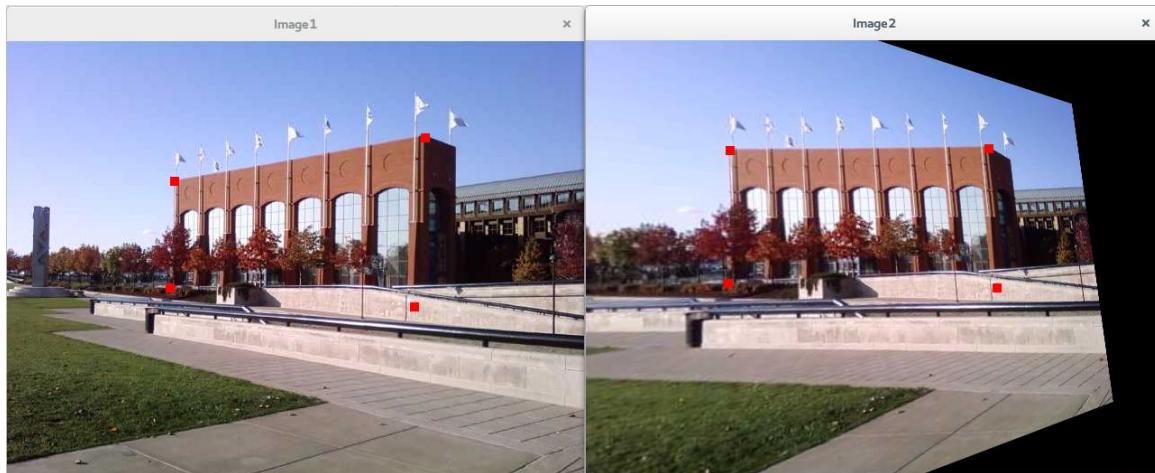
- ❖ BEV动静态元素检测任务 (含Temporal Fusion)

- ❖ 基于LSS的动静态元素检测任务
- ❖ 基于Transformer的动静态元素检测任务



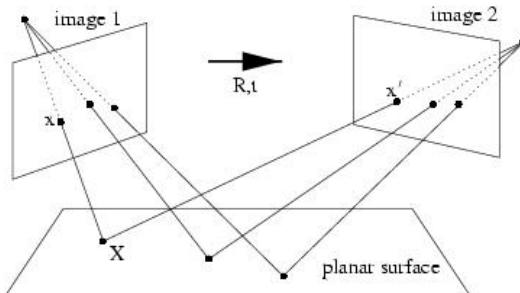
最原始的特征空间转换: IPM (Inverse Perspective Mapping)

- 我们经常能看到这样的应用

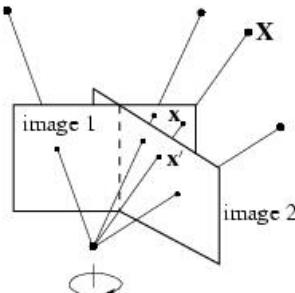


- 单应性变换: 对空间中同一平面的点成像, 当相机位置或角度变化

- 平移+旋转



- 仅旋转



- 单应 (Homography) 矩阵

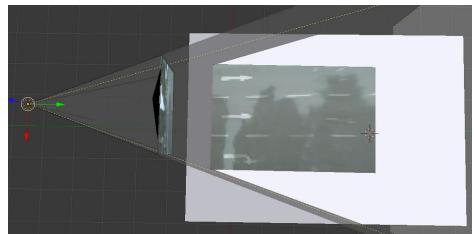
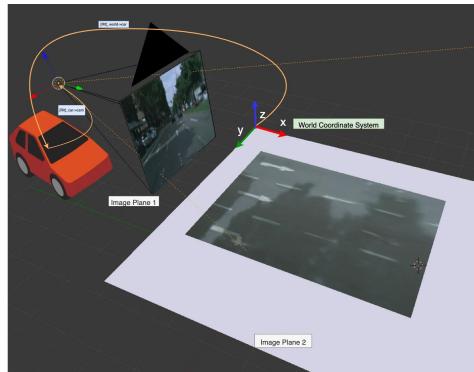
$$x = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = KT \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$x' = \begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} = KT' \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$x' = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} x = Hx$$

单应性: 空间中同一个平面上的点, 在图像空间中两个平面之间的变换关系

- 当目标平面变成地平面呢?



在地平面假设下, 利用单应性, 可以将像素直接投影到地平面

- 总结

- 精度低, 对内外参敏感, 鲁棒性差
- 地平面假设在实际工况中, 只有较少的场景下满足, 且距离越远效果越差
- 对有高度的目标, 投影后被拉长, 畸变严重
- 成本低
- 特征转换过程非常直观, 计算量相对比较小



举个例子:

- ◆ 为什么是BEV (Bird Eye View) ?

- ◆ View Transformation方法

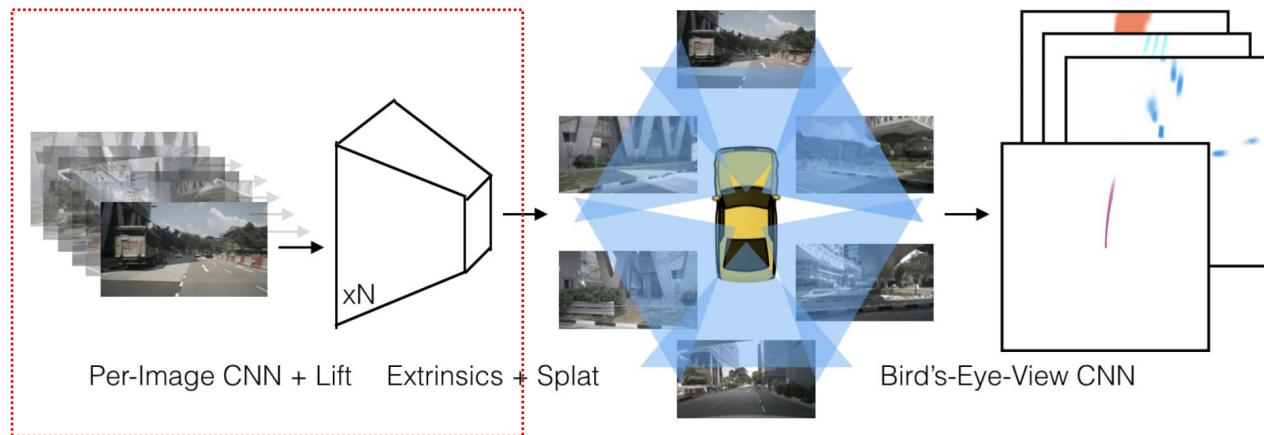
- ◆ IPM：最传统的方法
- ◆ LSS (2D to 3D/从底向上)：基于深度分布估计的方法
- ◆ Transformer (3D to 2D/从顶向下)：基于注意力机制的方法

- ◆ BEV动静态元素检测任务 (含Temporal Fusion)

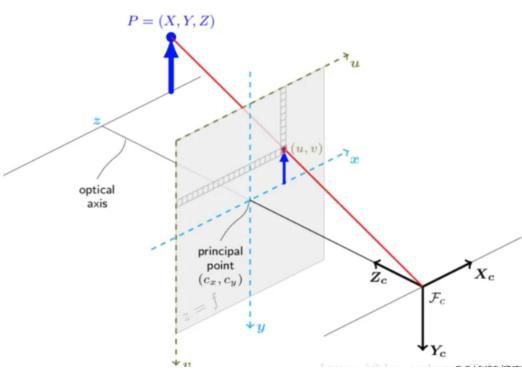
- ◆ 基于LSS的动静态元素检测任务
- ◆ 基于Transformer的动静态元素检测任务

视觉特征转换至BEV特征：LSS - Lift, Splat, Shoot

- LSS[1](Lift, Splat, Shoot) @202008: 通过预测每个像素的深度，还原BEV特征

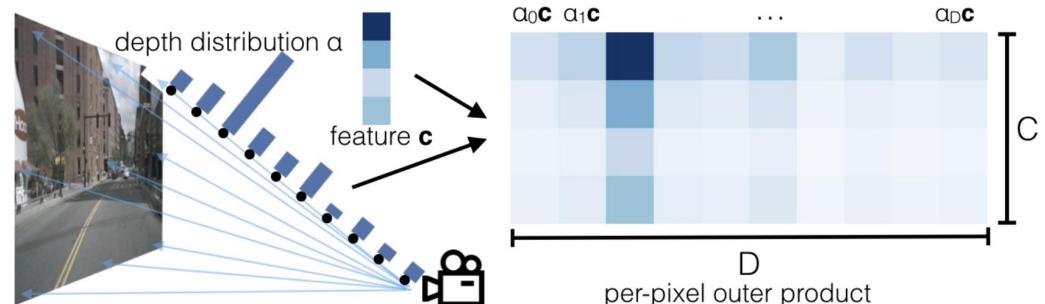


- 阶段1 Lift: 首先提取图像特征，估计每个像素深度分布估计



- 对每个相机的每个像素：

- 沿着光心和像素点的射线（视锥）方向均匀采样D个点
- 每个像素的特征向量为 $c \in \mathbb{R}^C$
- 用 c 作为输入，通过MLP预测一个D维向量，用来表示该像素在每个深度上的概率 $\alpha = [\alpha_0 \quad \alpha_1 \quad \dots \quad \alpha_{D-1}]$



- lift

- 通过向量 $c \in \mathbb{R}^C$ 和深度概率 α 的外积，得到每个像素对应每个深度的特征向量 $\alpha_i c$
- 通过像素估计到的每个深度特征为 $(u \quad v \quad d_i \quad \alpha_i c)$
- 结合相机成像原理，已知内外参的情况下，可求出像素对应的3D坐标

$$d_i \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = K[R \ t] \begin{pmatrix} x \\ y \\ z \end{pmatrix} = T \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = d_i T^{-1} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}$$

- 每一个像素对应的3D特征向量即完成Lift过程 $\phi(x, y, z) = \alpha_i c$

- 总结

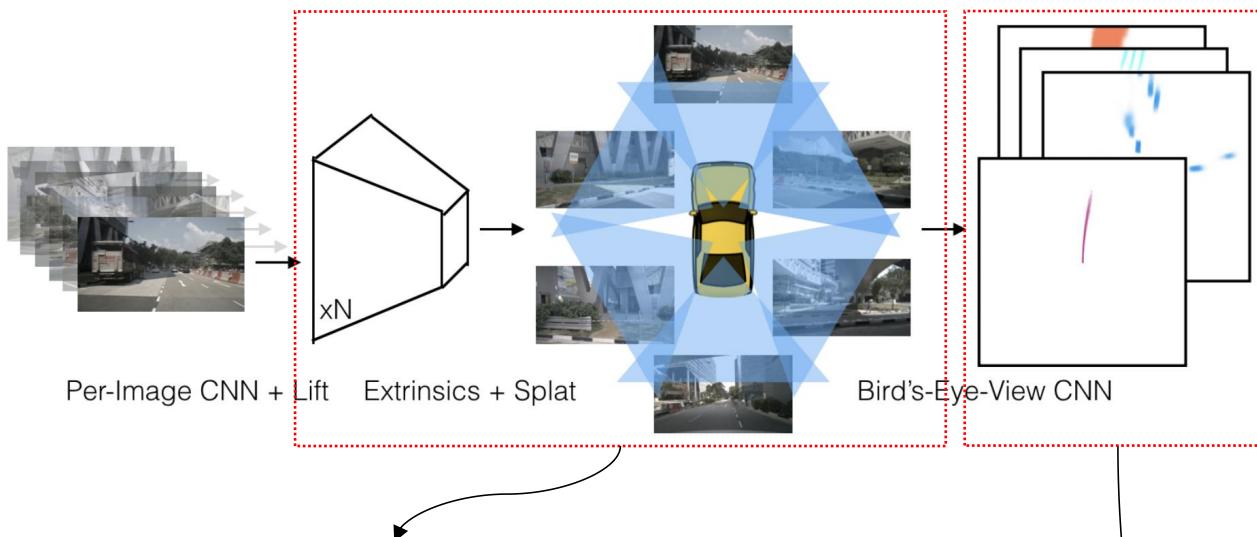
- Lift过程是通过估计深度，来实现2D特征转化为3D特征
- 此处的深度估计有几种不同的选择，优缺点如下：

特征向量形式	按预测概率分布	均匀分布	One-Hot
优势	可以识别深度，特征转换更精准	计算快，无学习参数	深度估计的识别度更高，实际为估计每个像素对应的点云
劣势	需要MLP做概率估计	深度估计无差异化	鲁棒性不好



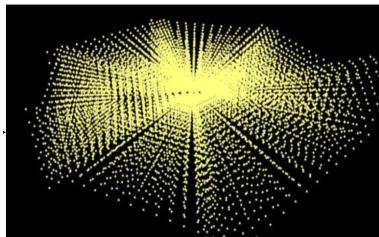
视觉特征转换至BEV特征：LSS - Lift, Splat, Shoot

□ LSS [1](Lift, Splat, Shoot) @202008



□ 阶段2 Splat: 将Lift的点特征压缩为BEV特征

• 输入分析



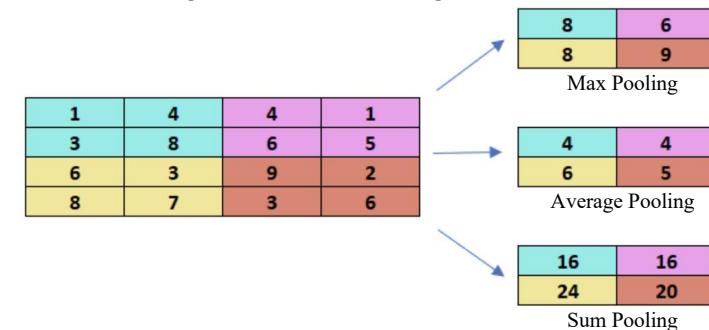
- 实际为均匀采样点云
- 每个点的特征为 $\phi(x, y, z) = \alpha_i \mathbf{c}$

• 难点分析

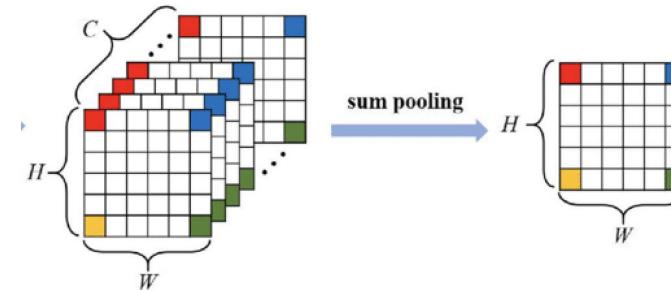
- 多相机存在重叠区域，特征压缩时不应有差异
- 点数量多：百万级的点 ($\pm 50m$ 范围，采样间隔为 $0.5m$)

• Splat

- 参考 PointPillars 将 BEV 空间划分为 $H \times W$ 的网格
- 对每一个估计的 3D 点 (x, y, z) ，将其特征归属至最近的 pillar
- 利用归属到某个特定 pillar 的特征进行 Sum Pooling



- 生成的 BEV 特征图如下：



- 点数量多，耗时长，将在工程部分讲解

□ 阶段3: Shoot是基于BEV特征，进行路径规划，在此不展开



第二章：从 Multi-View 到 BEV，构建新的感知模型

- ❖ 为什么是BEV (Bird Eye View) ?

- ❖ View Transformation方法

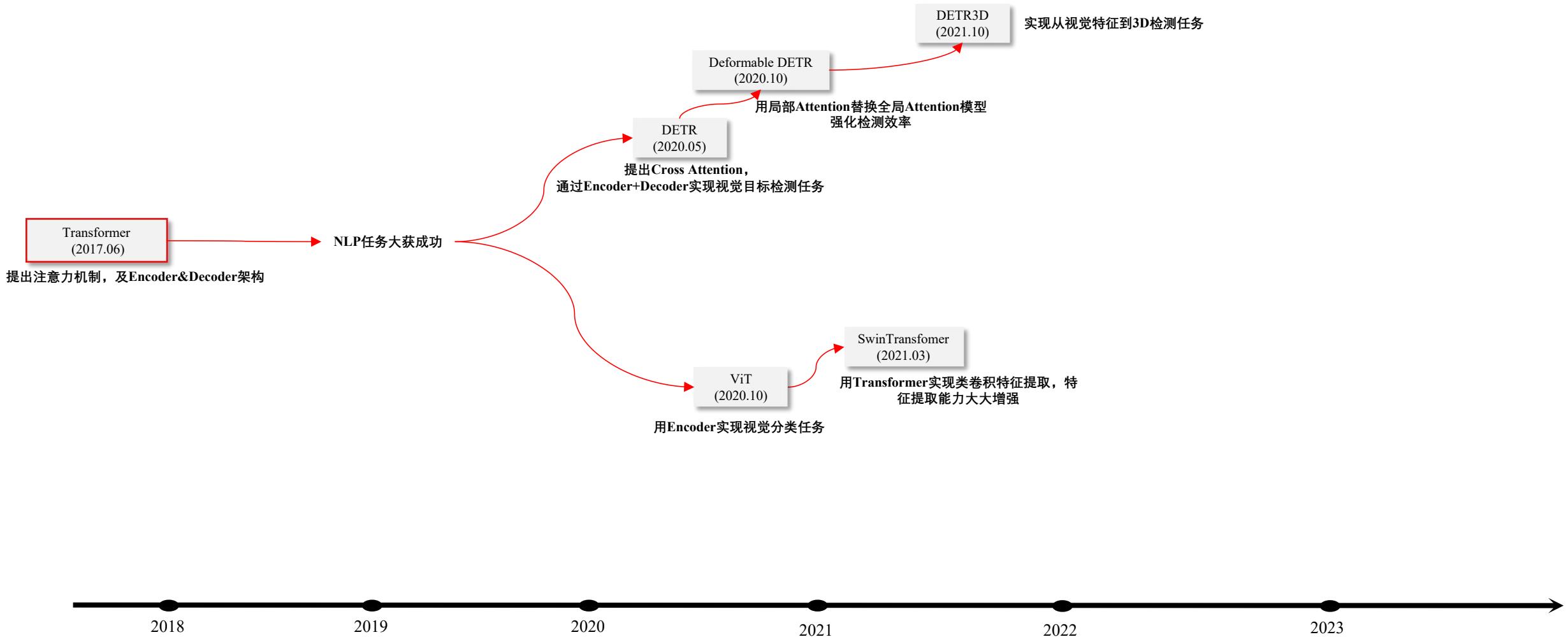
- ❖ IPM：最传统的方法
- ❖ LSS (2D to 3D/从底向上)：基于深度分布估计的方法
- ❖ Transformer (3D to 2D/从顶向下)：基于注意力机制的方法

- ❖ BEV动静态元素检测任务 (含Temporal Fusion)

- ❖ 基于LSS的动静态元素检测任务
- ❖ 基于Transformer的动静态元素检测任务



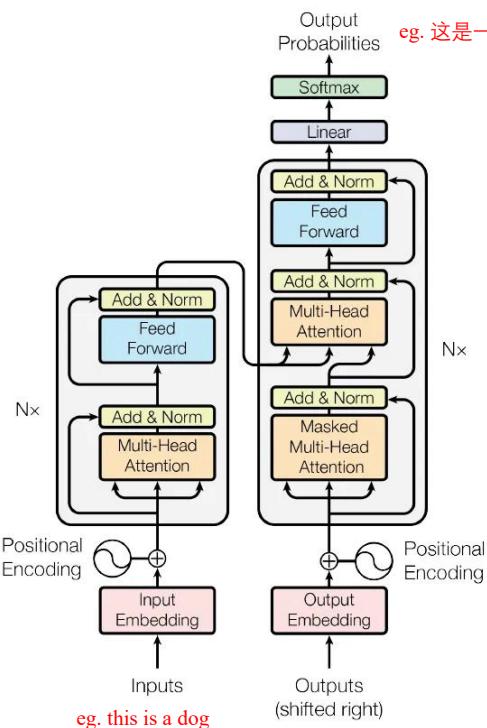
Transformer在视觉任务中的发展脉络





Transformer提出，在语音识别领域大获成功

- Transformer @201706 用于处理序列化数据，如翻译任务



- Self Attention

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}((\mathbf{Q} * \mathbf{K}^T) / \sqrt{d_k}) * \mathbf{V}$$

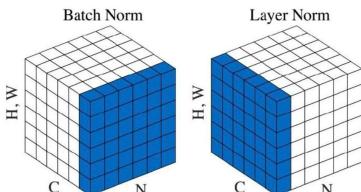
$$\begin{aligned}
 \mathbf{x}_1 * \mathbf{W}^Q &= \mathbf{q}_1 & \rightarrow \mathbf{q}_1 * \mathbf{k}_1 &= \mathbf{score}_1 & \rightarrow \text{softmax}(\mathbf{score}_1 / \sqrt{d_k}) * \mathbf{v}_1 \\
 \mathbf{this} \mathbf{x}_1 * \mathbf{W}^K &= \mathbf{k}_1 \\
 \mathbf{x}_1 * \mathbf{W}^V &= \mathbf{v}_1 \\
 \\
 \mathbf{x}_2 * \mathbf{W}^Q &= \mathbf{q}_2 & \rightarrow \mathbf{q}_1 * \mathbf{k}_2 &= \mathbf{score}_2 & \rightarrow \text{softmax}(\mathbf{score}_2 / \sqrt{d_k}) * \mathbf{v}_2 \\
 \mathbf{is} \mathbf{x}_2 * \mathbf{W}^K &= \mathbf{k}_2 \\
 \mathbf{x}_2 * \mathbf{W}^V &= \mathbf{v}_2
 \end{aligned}$$

- MultiHead Self Attention

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\mathbf{head}_1, \dots, \mathbf{head}_n) * \mathbf{W}^O$$

$$\mathbf{head}_i = \text{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V)$$

- LayerNorm: 与BN作用一致，正则化 + 优化收敛速度



- Feed Forward: 就是两层MLP, Hidden Layer会升维

$$\text{FeedForwardNetwork}(x) = \max(0, x\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2$$

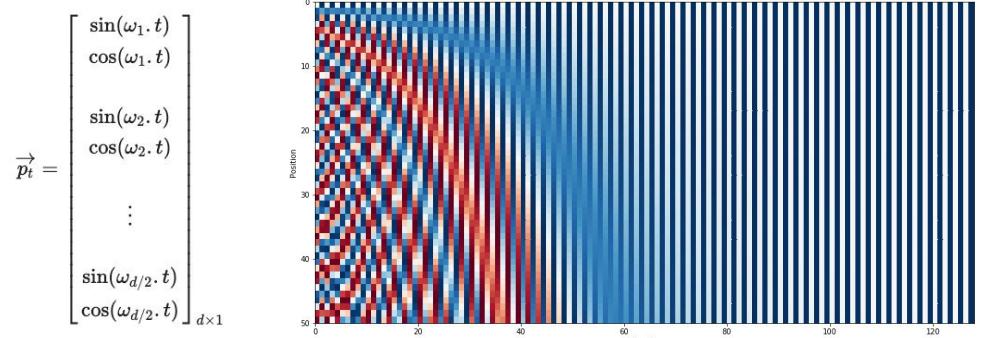
- Positional Encoding[2]: 用多维向量来表示位置编码

0 :	0 0 0 0	8 :	1 0 0 0
1 :	0 0 0 1	9 :	1 0 0 1
2 :	0 0 1 0	10 :	1 0 1 0
3 :	0 0 1 1	11 :	1 0 1 1
4 :	0 1 0 0	12 :	1 1 0 0
5 :	0 1 0 1	13 :	1 1 0 1
6 :	0 1 1 0	14 :	1 1 1 0
7 :	0 1 1 1	15 :	1 1 1 1

$$\vec{p}_t^{(i)} = f(t)^{(i)} := \begin{cases} \sin(\omega_k \cdot t), & \text{if } i = 2k \\ \cos(\omega_k \cdot t), & \text{if } i = 2k + 1 \end{cases}$$

$$t \text{ 表示位置, } \vec{p}_t \in \mathbb{R}^d \text{ 表示位置编码 } \omega_k = \frac{1}{10000^{2k/d}}$$

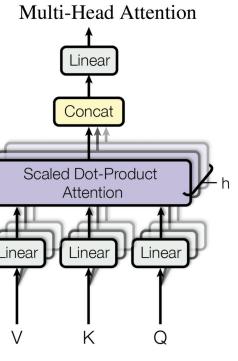
eg: 用128维向量表达50的长度



[1] Attention and Transformer Models, <https://towardsdatascience.com/attention-and-transformer-models-fe667f958378>

[2] The Positional Encoding, https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

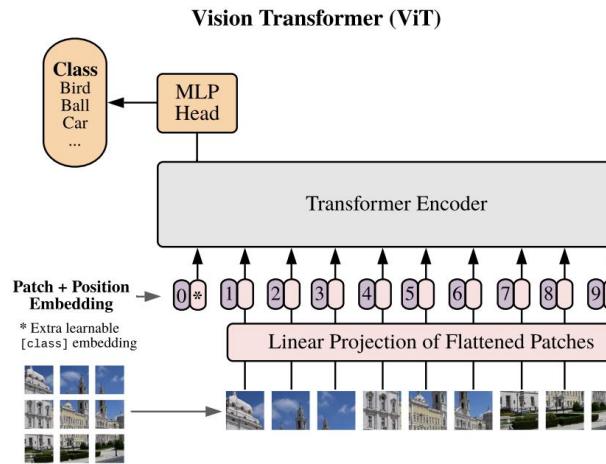
[3] Layer Normalization, <https://leimao.github.io/blog/Layer-Normalization/>



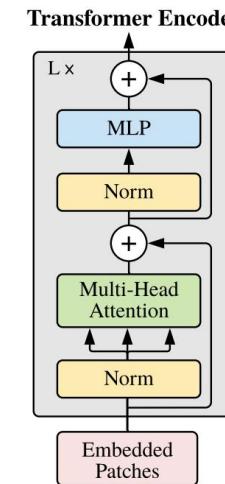


Transformer的Encoder结构，用于视觉分类任务和更复杂的目标检测任务

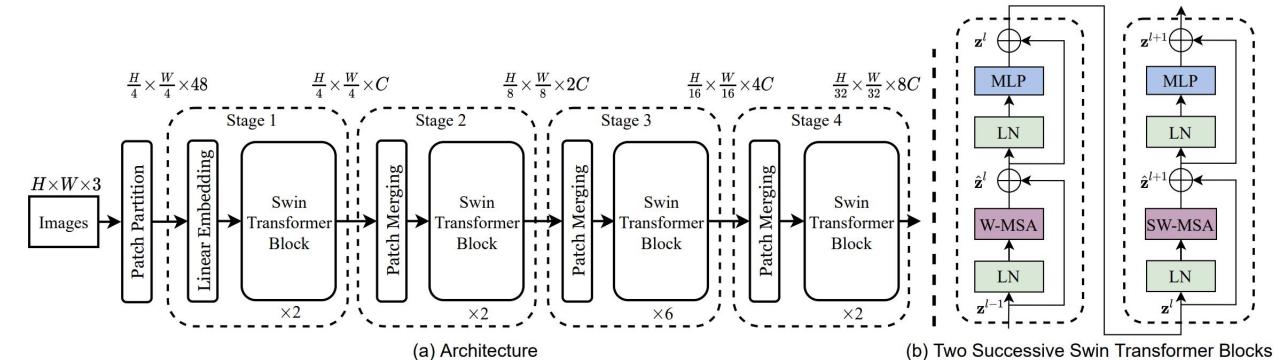
□ ViT(Vision Transformer)[1] @202010, 将Transformer引入视觉任务，仅用Encoder做分类任务



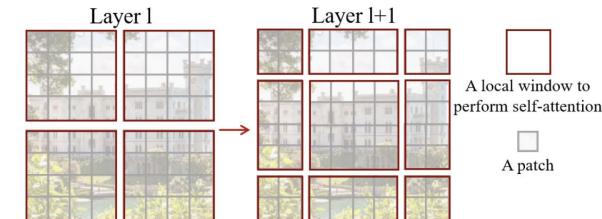
- 为避免在像素级别做全图自注意力，选择在Patch级别做自注意力，对于较大图片算力开销巨大
- 仅使用了Encoder结构



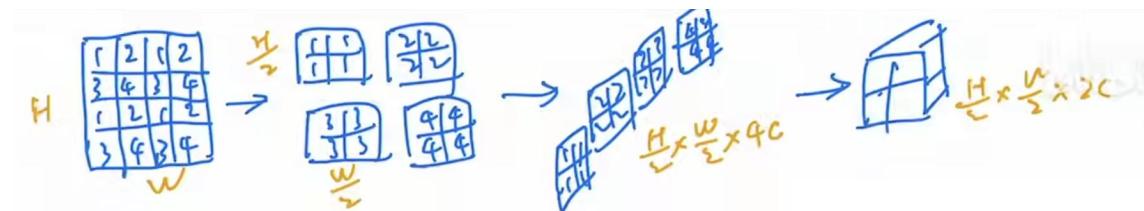
□ SwinTransformer[2] @202103, 用Transformer实现类卷积结构，优化分类任务检测结果



- Sliding Window机制使得自注意力跳出Window范围



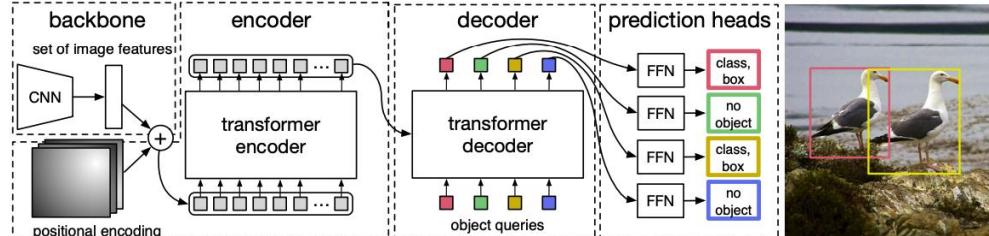
- 通过Patch Merging实现下采样+Channel增加



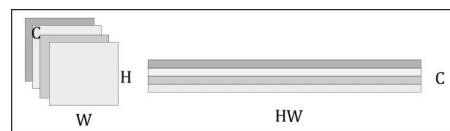
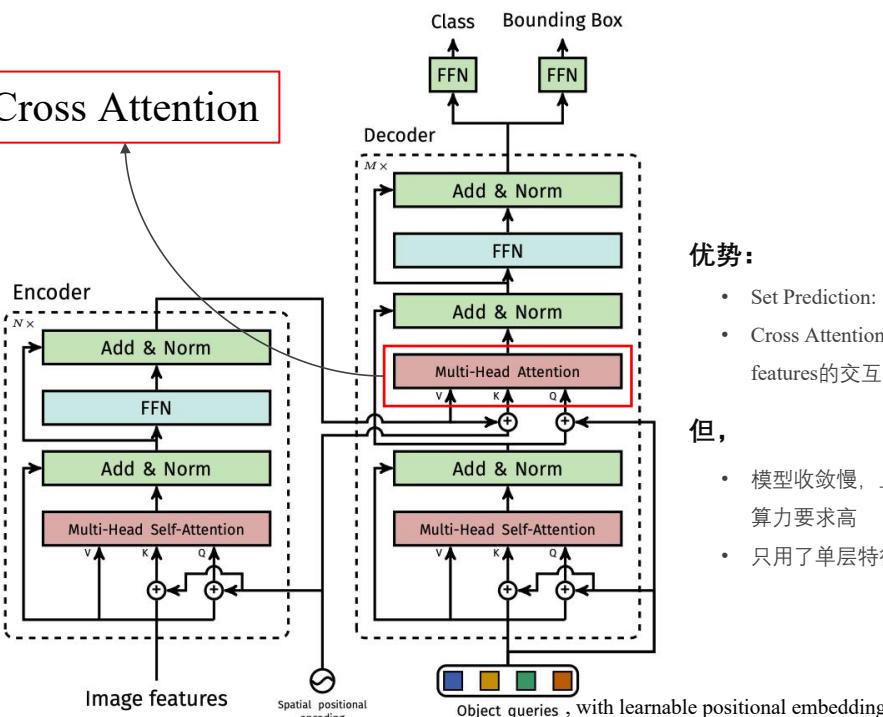


DETR提出Cross Attention，实现端到端2D检测任务，Deformable DETR进一步优化模型效率

□ DETR[1] @202005: 用Encoder + Decoder架构进行目标检测任务

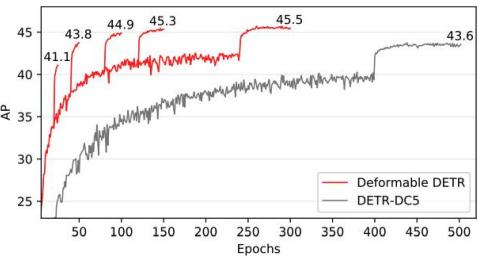
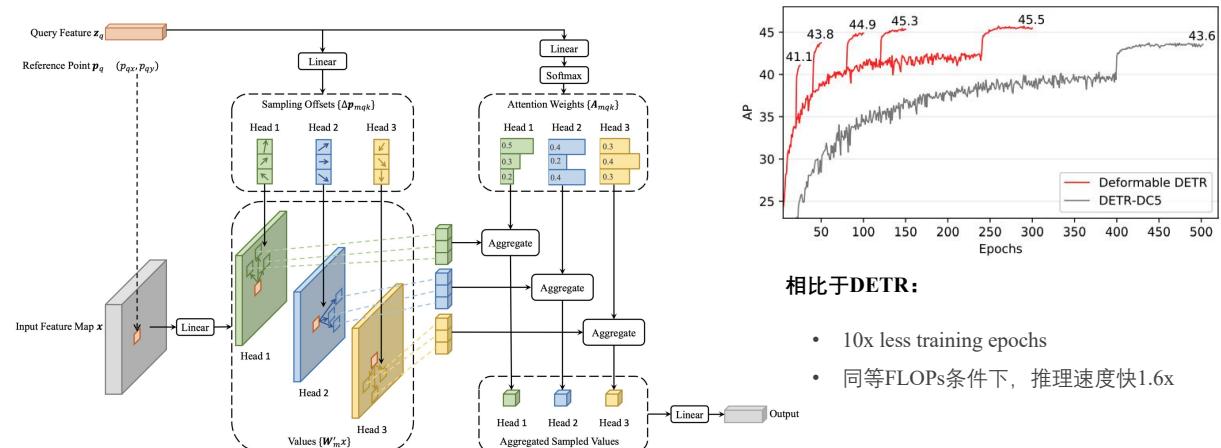
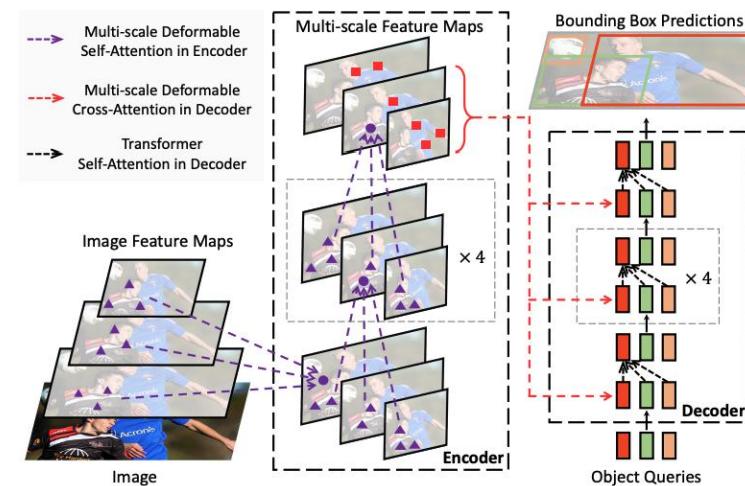


Cross Attention



- Feature map是下采样后的结果, 且通过1x1卷积降通道数
- 通过Spatial Positional Encoding进行位置编码

□ Deformable DETR[2] @202010: 将注意力约束在局部区域, 提升模型效率



相比于DETR:

- 10x less training epochs
- 同等FLOPs条件下, 推理速度快1.6x

每个query采样K个位置

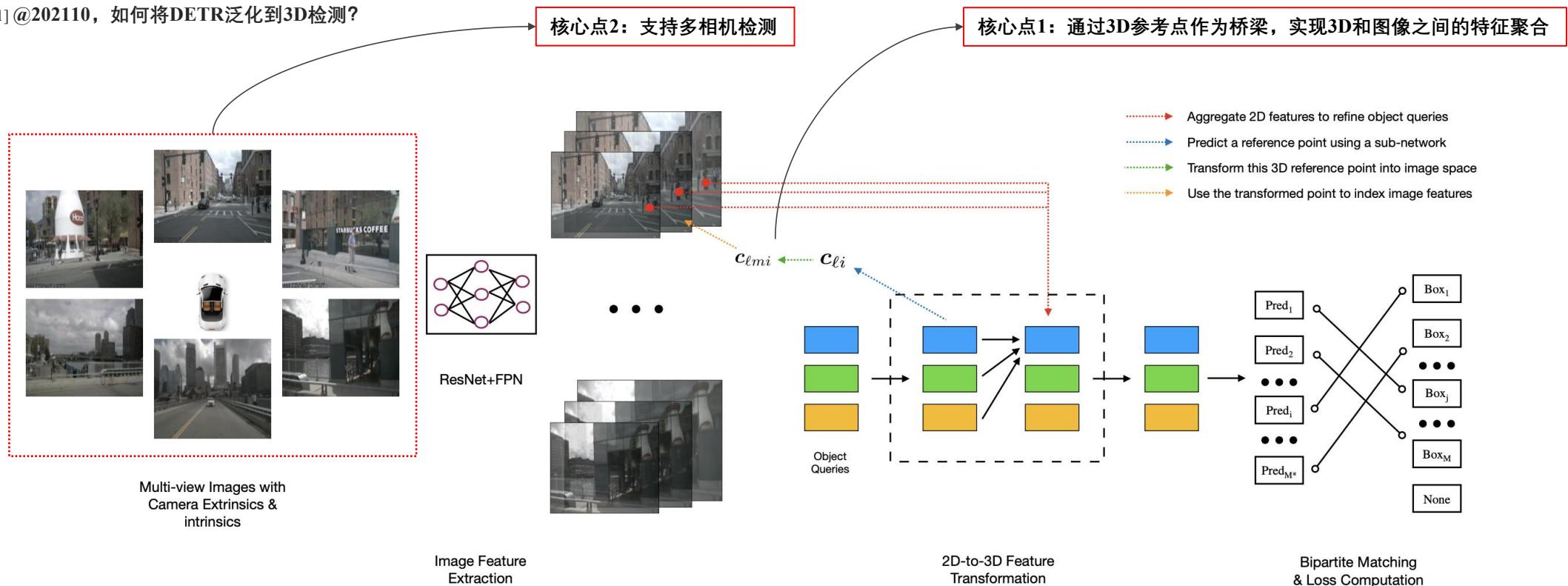
$$\text{MSDeformAttn}(z_q, \hat{p}_q, \{\mathbf{x}^l\}_{l=1}^L) = \sum_{m=1}^M \mathbf{W}_m \left[\sum_{l=1}^L \sum_{k=1}^K A_{mlqk} \cdot \mathbf{W}'_m \mathbf{x}^l(\phi_l(\hat{p}_q) + \Delta p_{mlqk}) \right]$$

参考点 偏移



DETR3D，实现视觉到3D的目标检测

■ DETR3D[1] @202110, 如何将DETR泛化到3D检测?



• 初始化

- 经过ResNet和FPN处理的多尺度特征: $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3, \mathcal{F}_4$

$$\mathcal{F}_k = \{\mathbf{f}_{k1}, \dots, \mathbf{f}_{k6}\} \subset \mathbb{R}^{H \times W \times C}$$

- 每一层的Object Queries, 经过自回归方式预测下一层 $\mathcal{Q}_{\ell+1}$

$$\mathcal{Q}_{\ell} = \{\mathbf{q}_{\ell 1}, \dots, \mathbf{q}_{\ell M^*}\} \subset \mathbb{R}^C, \text{ 其中 } \ell \in \{0, \dots, L-1\}$$

- 每个Query都用MLP预测一个3D点

$$\mathbf{c}_{\ell i} = \Phi^{\text{ref}}(\mathbf{q}_{\ell i}) \quad \text{其中 } \mathbf{c}_{\ell i} \in \mathbb{R}^3$$

• 特征聚合

- 将对应的3D点, 通过内外参投影回相机平面, 其中 m 指不同的相机

$$\mathbf{c}_{\ell i}^* = \mathbf{c}_{\ell i} \oplus 1$$

$$\mathbf{c}_{\ell mi} = T_m \mathbf{c}_{\ell i}^*$$

- 对 $\mathbf{c}_{\ell mi}$ 做归一化处理, 在不同尺度的特征图上采样特征

$$\mathbf{f}_{\ell km i} = f^{\text{bilinear}}(\mathcal{F}_{km}, \mathbf{c}_{\ell mi})$$

- 采样后的特征求和, 其中 $\sigma_{\ell km i}$ 表示投影点是否在某个相机中存在

$$\mathbf{f}_{\ell i} = \frac{1}{\sum_k \sum_m \sigma_{\ell km i} + \epsilon} \sum_k \sum_m \mathbf{f}_{\ell km i} \sigma_{\ell km i}$$

• Query更新

- 更新至下一层Queries

$$\mathbf{q}_{(\ell+1)i} = \mathbf{f}_{\ell i} + \mathbf{q}_{\ell i}$$

- 最后一层Queries用来预测最后的结果:

$$\hat{\mathbf{b}}_{\ell i} = \Phi_{\ell}^{\text{reg}}(\mathbf{q}_{\ell i}) \quad \hat{\mathbf{c}}_{\ell i} = \Phi_{\ell}^{\text{cls}}(\mathbf{q}_{\ell i})$$

其中:

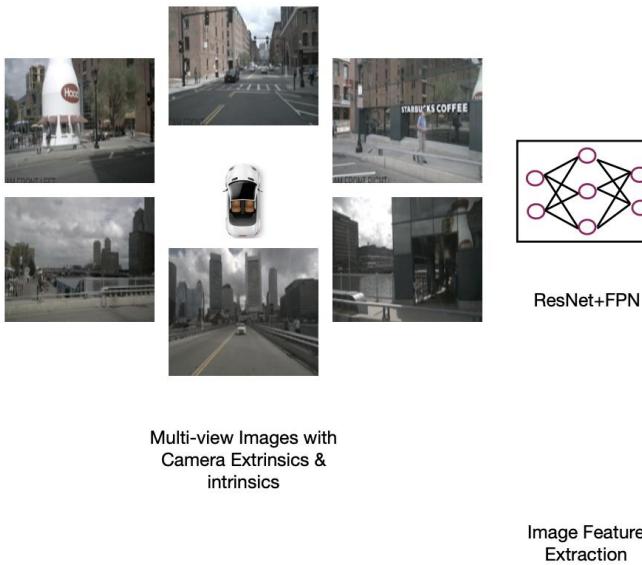
- $\hat{\mathbf{b}}_{\ell i}$ 为3D Bounding Box, $\hat{\mathbf{c}}_{\ell i}$ 为分类结果
- 3D Bounding Box通常包含BEV空间下的: 位置, 尺寸, heading angle和速度



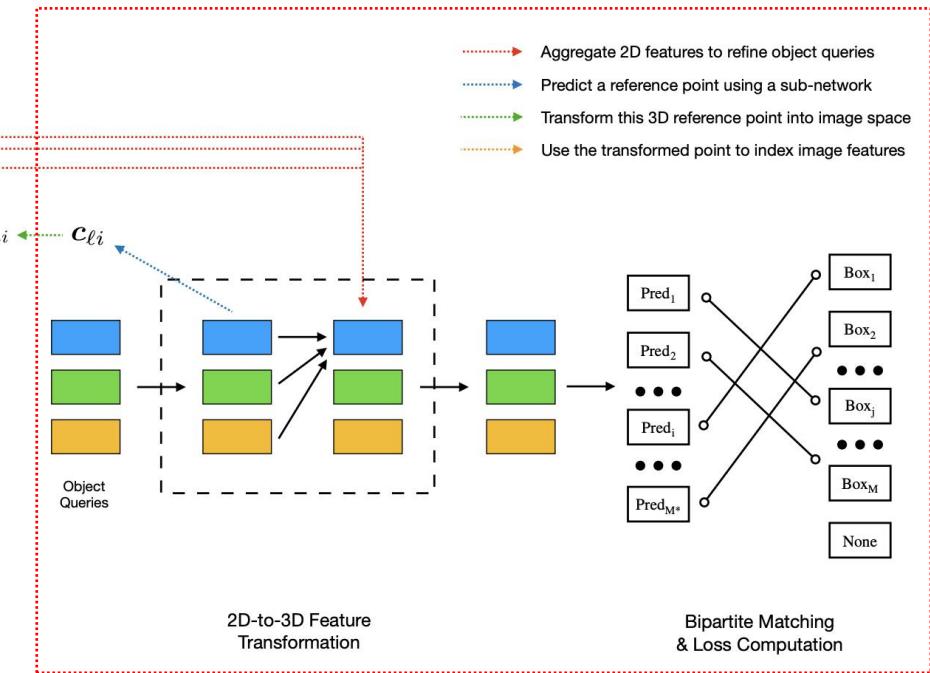
视觉特征转换至BEV特征：Transformer

- 从DETR3D可以看到，通过Cross Attention可以进行特征空间转换

稀疏Query

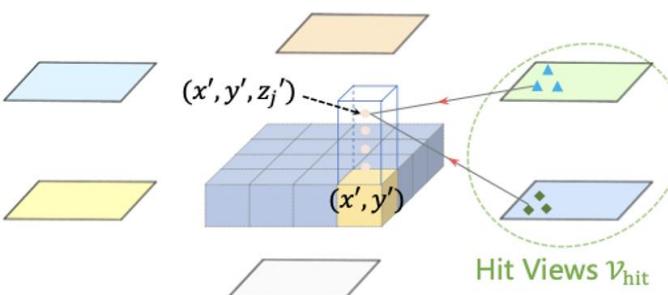


- 当Query为特定目标时：



- 是否可能让BEV网格中的每一个grid作为Query呢？

稠密Query



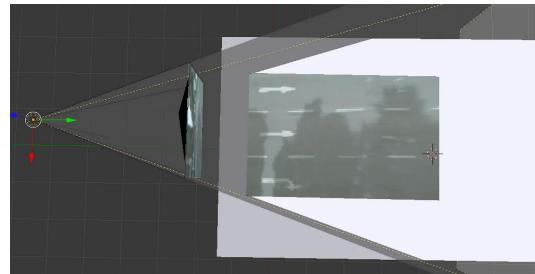
步骤：

- 初始化一个grid shaped的可学习参数 $Q \in \mathbb{R}^{H \times W \times C}$, 划分为 H, W 对应于的网格BEV空间的大小, C 作为每个grid的query, positional embedding为可学习的参数
- 对于每一个grid的中心点 (x', y') , 在高度上采样 N 个点得到 $(x', y', z'_j)_{j=1}^N$
- N 个点投影到不同图像可得到对应的BEV点特征
- 最终聚合成BEV特征



特征空间转换 (View Transformation) 方法汇总

□ IPM: 2D像素特征直接投影至BEV空间

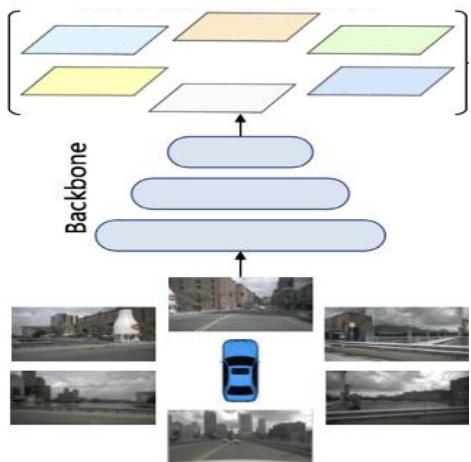


计算成本低

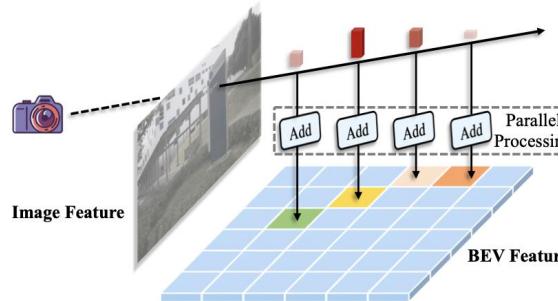
鲁棒性低

精度低

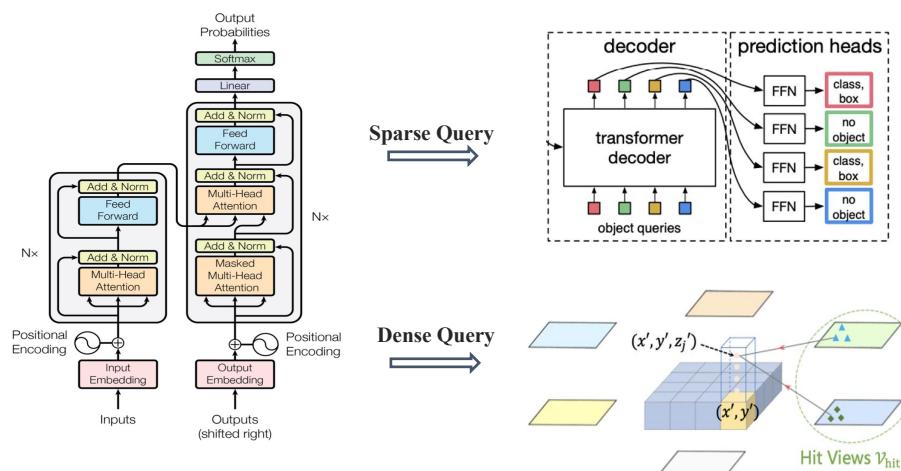
Feature Extraction



□ 2D to 3D: 2D图像预测视锥深度分布，通过camera内外参投影到BEV空间



□ 3D to 2D: 3D Query通过transformer查询2D feature，生成BEV特征



计算成本高

鲁棒性高

精度高