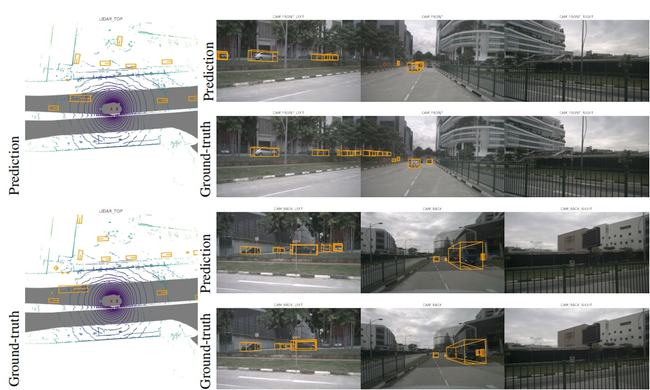
# Detr3D Profile

## 1. Detr3D introduction

* 模型输入：来自6个camera的image (如右侧所示)，以及camera to lidar的转换矩阵

* 模型输出：object在lidar坐标系下的3dbbox，project到鸟瞰图下如左图所示



### 1.1 Model design

论文配套的代码已经开源，参考：[DETR3D](https://github.com/WangYueFt/detr3d) , inference流程有4个Step

1. 从nuscenes数据集中加载某个timestamp的6张RGB图像，组织成NCHW的 (6, 3, 900, 1600)

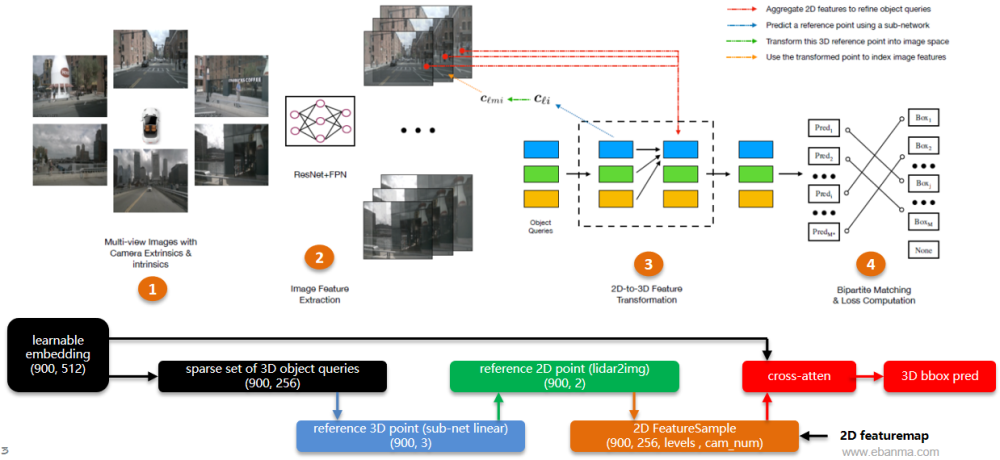
1. 6作为batch维度，进行Resnet+FPN抽取视觉语义，得到Image space上的featuremap

1. 预测BEV视角下的3D bbox (lidar坐标系)，操作类似DETR中的decoder，依赖相机-Lidar转换参数

* 1. 结合Lidar2cam的转换矩阵将可学习的3d reference point转到image space上，并采样出step2中对应pixel位置的features

* 1. 另一组可学习的object queries先做self-attention， 然后结合a中的features做cross-attention

1. transformer的输出经过Linear后得到cls\_branch和bbox\_branch，作为3d bbox 的pred的结果



### 1.2 flops summary

除了原opensource提供的resnet101 backbone和vovnet backbone，额外的进行了regnet-400mf的裁剪实验

* 实验基于单张GTX-2080，在FP32下进行latency profile

* 原codebase的resnet101带有deformable-conv，不被torchscript导出所支持，这里被移除

* 新增regnet-400mf除了backbone比较轻量，decoder也由6层降低为3层，FPN输出level=2

使用mmdet3d中的get\_flops工具进行不同backbone的DETR3D算力统计：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Detr3D's**  **Backbone** | **input-size** | **decoder-layers** | **Latency**  **(ms)** | **Params**  **(MB)** | **Total**  **GFlops** | **Backbone+Neck** | |
| **GFlops** | **比重** |
| resnet101 | (6, 3, 900, 1600) | 6 | 398 | 51.53 | 1469 | 1465 | 99.8% |
| vovnet | (6, 3, 900, 1600) | 6 | 620 | 77.51 | 3324 | 3320 | 99.8% |
| regnet-400mf | (6, 3, 900, 1600) | 3 | 77 | 9.95 | 102 | 100 | 98% |

Note. GFlops来自 mmdet3d的flops\_counter, Latency来自torch.autograd.profiler (part 2.1)

### 1.3 Detr3D和Tesla workload之间的Gap

* Tesla单个camera分辨率为1280x720x1，DETR3D的单个camera为1600x900x3

* Tesla是8个camera进行fusion，DETR3D是6个camera fusion(nuscenes dataset)

* Telsa使用BiFPN, DETR3D使用的是FPN，前者算力开销稍多一些，额外增加一条融合路径

* DETR3D只进行了multi-camera 3d bbox的预测，不包含时序上的信息融合、语义地图的预测

* Tesla可能使用了稠密的80x20个object queries来送入BEV Transformer，DETR3D使用的是900个稀疏的object queries

## 2. Export results

### 2.1 export torchscript

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **backbone** | **torchscript** | **torch.autograd**  **.profile 's log** | **mmdet3d 's log** | **mmdet3d config** |
| resnet101 | [📎detr3d\_resnet101.pt](https://yuque.antfin.com/attachments/lark/0/2022/pt/21256453/1654502312890-e718d1b4-3853-427d-8d28-88ede2875b92.pt) | [📎detr3d\_resnet101.txt](https://yuque.antfin.com/attachments/lark/0/2022/txt/21256453/1654502312892-55b11bfb-5bdf-4809-ba3e-c130b79e354d.txt) | [📎detr3d\_resnet101\_mmdet3d.txt](https://yuque.antfin.com/attachments/lark/0/2022/txt/21256453/1654502312900-cc1c676f-0883-41eb-b24c-ed52bfcb016f.txt) | [📎detr3d\_res101\_gridmask\_modify.py](https://yuque.antfin.com/attachments/lark/0/2022/py/21256453/1654502312921-a869bddc-618e-4176-afd0-0f2227e91d8e.py) |
| vovnet | [📎detr3d\_vovnet.pt](https://yuque.antfin.com/attachments/lark/0/2022/pt/21256453/1654502312890-9102cd79-4fb0-4e6c-9e92-4da1f8f0e5cb.pt) | [📎detr3d\_vovnet.txt](https://yuque.antfin.com/attachments/lark/0/2022/txt/21256453/1654502313425-66c28864-05ab-4f42-bed5-97c674501b71.txt) | [📎detr3d\_vovnet\_mmdet3d.txt](https://yuque.antfin.com/attachments/lark/0/2022/txt/21256453/1654502313172-b8ecdf4f-9bcc-4912-8dfa-669a34870d1c.txt) | [📎detr3d\_vovnet\_gridmask\_det\_final\_trainval\_cbgs.py](https://yuque.antfin.com/attachments/lark/0/2022/py/21256453/1654502313176-3652f0f2-0bb7-4fdd-b9e1-7b5edcd4f6e8.py) |
| regnet-400mf | [📎detr3d\_regnet400mf.pt](https://yuque.antfin.com/attachments/lark/0/2022/pt/21256453/1654502313185-b3a7a56e-143b-452e-b426-50613710bca8.pt) | [📎detr3d\_regnet-400mf.txt](https://yuque.antfin.com/attachments/lark/0/2022/txt/21256453/1654502313209-d5c1814a-6cbf-4e8a-b074-6ae745fdb2d5.txt) | [📎detr3d\_regnet-400mf\_mmdet3d.txt](https://yuque.antfin.com/attachments/lark/0/2022/txt/21256453/1654502313380-a7797786-e679-491f-82c1-0048c0299d2a.txt) | [📎detr3d\_regnet400mf\_gridmask.py](https://yuque.antfin.com/attachments/lark/0/2022/py/21256453/1654502313447-7abec7b1-f415-450d-a5d3-798086057aca.py) |

### 2.2 input & output info

input: img, torch.Size([6, 3, 928, 1600])  
output: bboxes, torch.Size([300, 9])  
output: scores, torch.Size([300])  
output: labels, torch.Size([300])

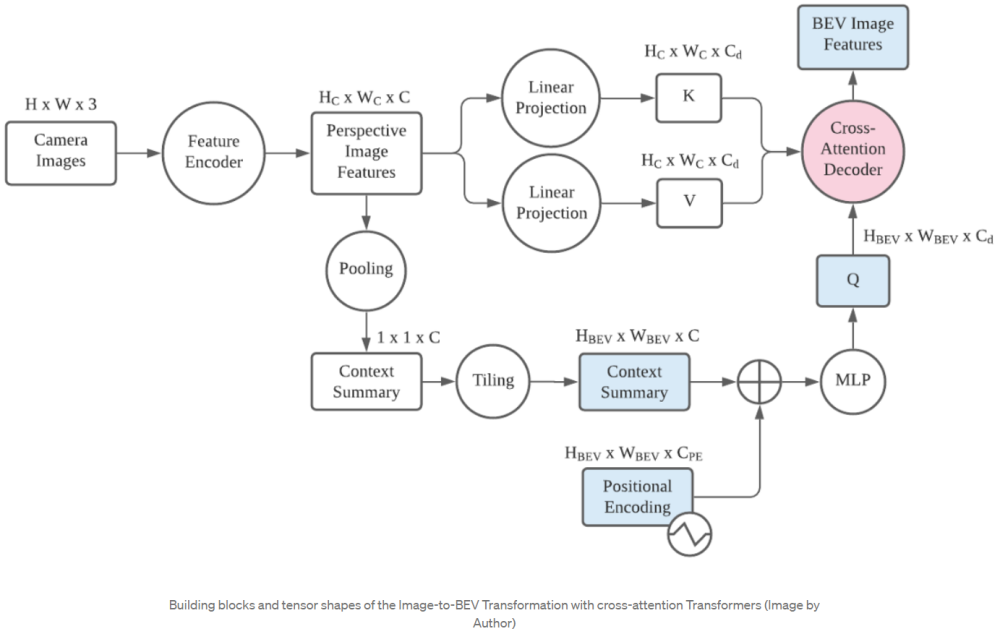
## 3. Reference

### 3.1 非官方的Tesla模型结构猜测

来自[这里](https://towardsdatascience.com/monocular-bev-perception-with-transformers-in-autonomous-driving-c41e4a893944)

* 和DETR3D有不同的是cross-attention中query来自对image space feature map的pooling

* 这个不一致可能对算力开销影响比较小



### 3.2 关于语义地图的预测

方式之一是补充一些line到queries中，进行车道线、马路边界等信息的预测，来自[STSU](https://arxiv.org/abs/2110.01997)

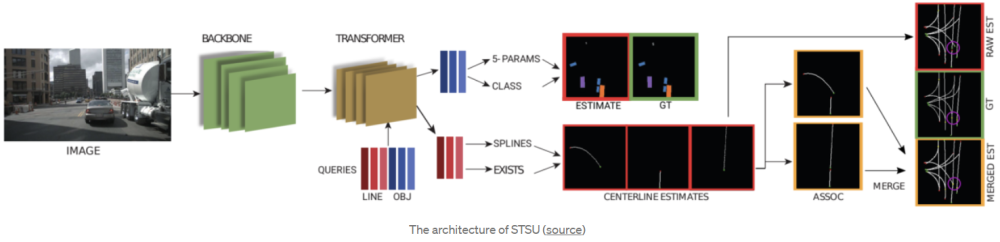
* STSU uses two sets of query vectors, one set for centerlines and one for objects. What is most interesting is its prediction of the structured road layout. The lane branch includes several prediction heads.

* + The detection head predicts if the lane encoded by a certain query vector exists.

* + The control head predicts the location of R [Bezier curve](https://en.wikipedia.org/wiki/B%C3%A9zier_curve) control points.

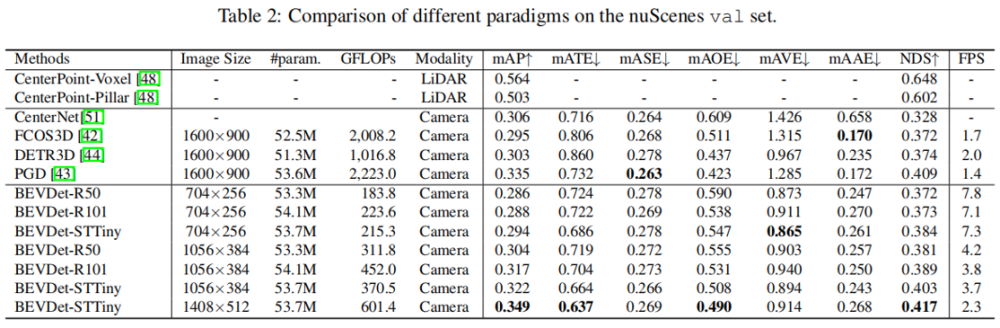
* + The association head predicts an embedding vector for clustering.

* + The association classifier takes in 2 embedding vectors and judges whether the centerline pairs are associated.



### 3.3 Multi-Camera做fusion的算力下限在哪里？

[BEVDet](https://zhuanlan.zhihu.com/p/454569125) 释放了一些nuscenes上的GFlops，目前看到的最小Flops在183GFlops (掉点也比较明显)



* 纯camera的方案比lidar based方案(centerPoint)的mAP还是要低不少

* 这篇paper对DETR3D的GFLOPS统计是使用了DCN的resnet101

* + 我们使用MMDET3d的get\_flops统计DETR3d-resnet101的算力为996GFlops，Params为53.34MB，比较接近论文对DETR3D的描述数据

* + 替换DCN为普通卷积后，用MMDET3D的Flops统计，算力从996->1469Gflops