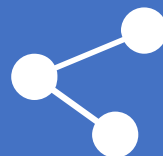




南京邮电大学
Nanjing University of Posts and Telecommunications



工作汇报



汇报人：解君豪

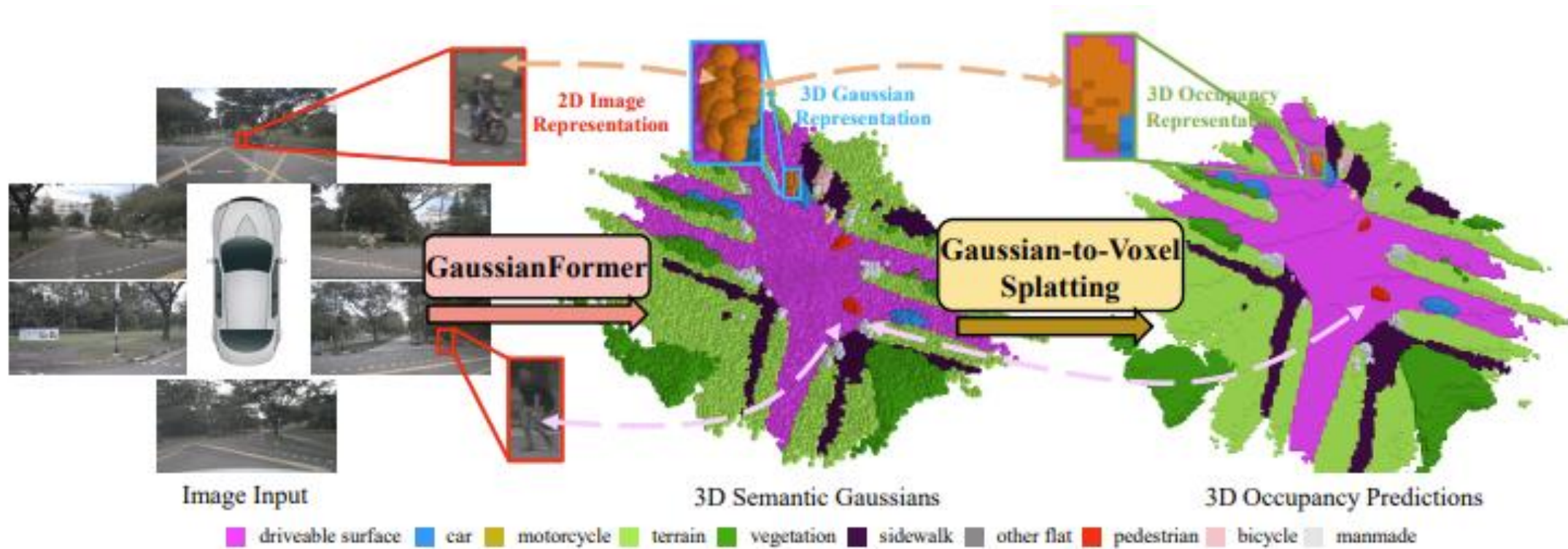


GaussianFormer: Scene as Gaussians for Vision-Based 3D Semantic Occupancy Prediction

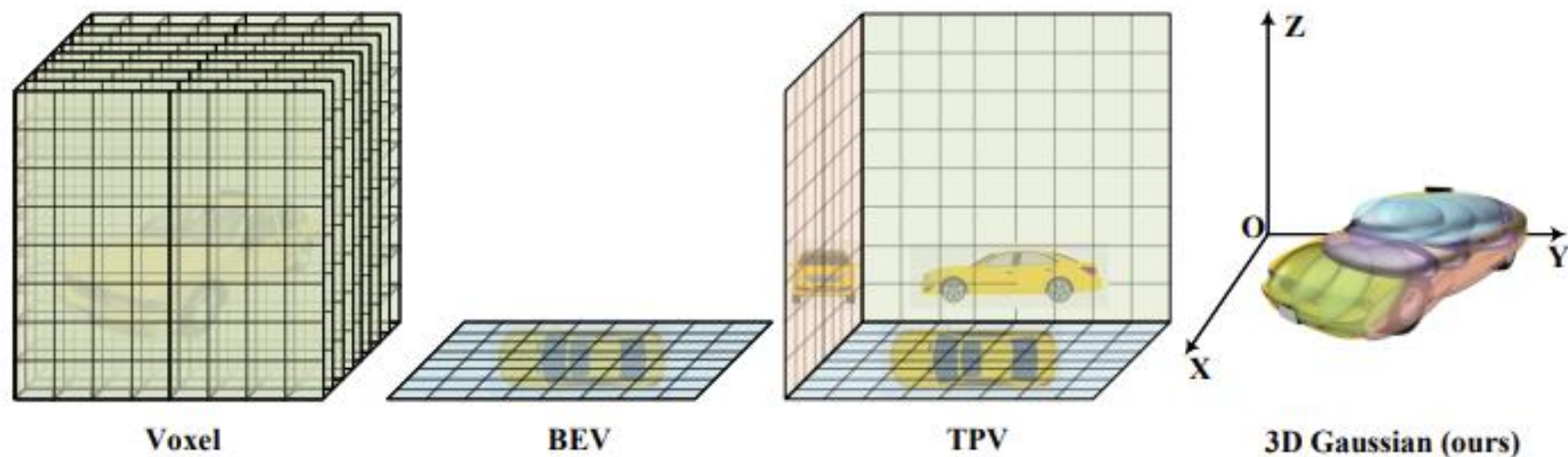
Yuanhui Huang, Wenzhao Zheng[†], Yunpeng Zhang, Jie Zhou, Jiwen Lu[‡]
Tsinghua University, UC Berkeley

[Paper (Arxiv)] [Code (GitHub)]

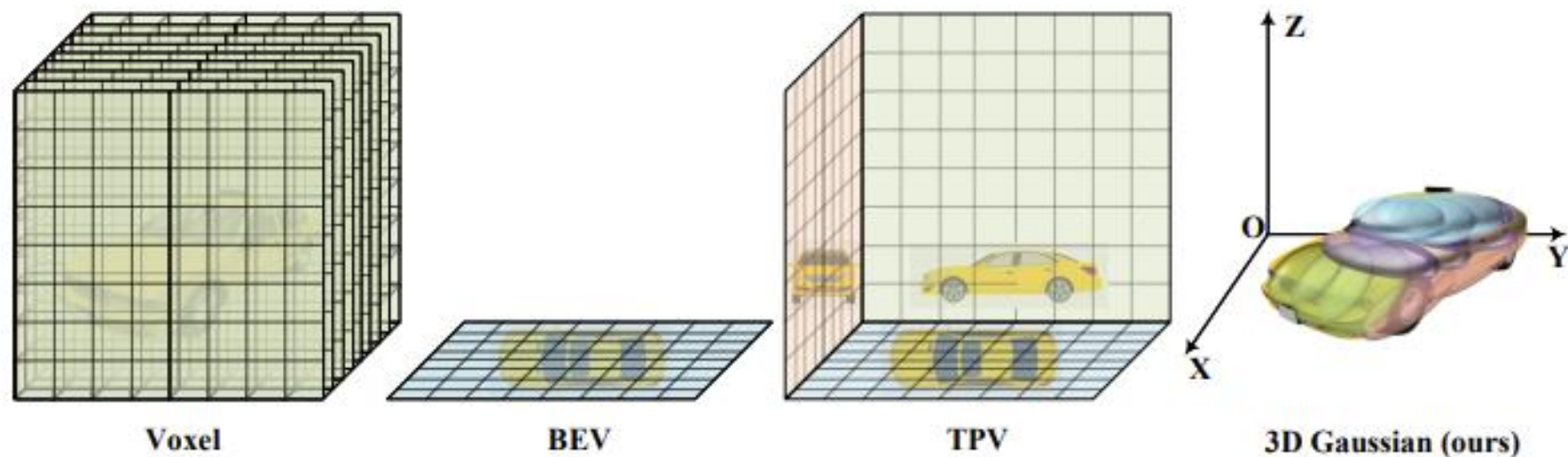
[†] Project Leader. [‡] Corresponding author.



- 提出一种面向对象的三维语义高斯表示
- 结合稀疏卷积和跨注意力机制，高效的将二维图像转换为三维高斯表示
- 设计了一个可通过cuda高效实现高斯到体素的泼溅模块



- 体素表示：将三维空间划分为规则的立方体网格，每个体素相当于一个像素立方块，对于每个像素，分配一个长度固定的特征向量，用于刻画该位置的几何占据/语义类别
- BEV(鸟瞰图)：将三维点/图像特征沿着高度方向(Z轴)投影/压缩到地平面上，得到一个二维网格图，该平面每个像素对应一定范围的水平位置，通道上编码高度、密度、语义、特征等信息
- TPV(三视角)：在BEV的基础上，增加前视角(FV)、测视角(SV)等额外的二维平面，将三维空间从多个正交方向进行投影。BEV捕获平面布局，FV补充垂直高度与立面细节，SV进一步捕获深度和高度的联合信息。



- 体素表示：为三维空间中每个体素分配一个特征，但是由于三维空间的稀疏性，这会导致大量冗余。
- BEV和TPV：采用2D平面来描述三维空间，虽然能够在一定程度上缓解冗余问题，但是会丢失部分细节。
- 提出的面向对象的三维高斯表示，通过混合高斯的强大逼近能力，能够自适应的聚焦于灵活的兴趣区域，同时能够刻画场景的细粒度结构，有效兼顾效率和表达能力。

基于视觉的三维语义占据预测旨在利用多视角相机图像，预测某一范围内每个体素网格的密集占据状态以及其语义类别，形式化的，给定多视角图像集合：

$$I = \{I_i \in \mathbb{R}^{3 \times H \times W} \mid i = 1, 2, 3 \dots N\}$$

相机内参数集合：

$$K = \{K_i \in \mathbb{R}^{3 \times 3} \mid i = 1, 2, 3 \dots N\}$$

以及相机外参数集合：

$$T = \{T_i \in \mathbb{R}^{4 \times 4} \mid i = 1, 2, 3 \dots N\}$$

目标是：预测三维语义占据体积：

$$O \in \mathbb{R}^{C \times X \times Y \times Z}$$

其中N为视角数量，{H, W}为图像分辨率，C为语义类别，{X, Y, Z}为目标体素分辨率

由于体素、BEV、TPV三种表达方式的弊端，本文提出了一种面向对象的三维表示方式，用可变形的三维高斯分布来描述场景中的各个区域，而非固定的网格，如图所示，我们用P个三维高斯构成的集合 $\mathcal{G} = \{G_i\}_{i=1}^P$ ，每个高斯由一个d维的向量表示($\mathbf{m} \in \mathbb{R}^3$, $\mathbf{S} \in \mathbb{R}^3$, $\mathbf{r} \in \mathbb{R}^4$, $\mathbf{c} \in \mathbb{R}^{|c|}$)，其中 \mathbf{m} 为高斯中心坐标， \mathbf{S} 为尺度向量(协方差矩阵)， \mathbf{r} 为四元数， \mathbf{c} 为语义特征。

设点 $\mathbf{p}=(x, y, z)$ ，单个语义高斯在该位置的分布值为：

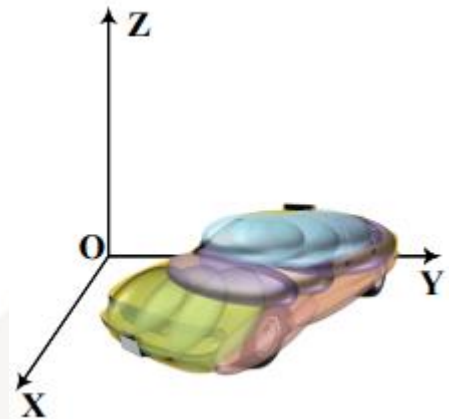
$$g(\mathbf{p}; \mathbf{m}, \mathbf{s}, \mathbf{r}, \mathbf{c}) = \exp\left(-\frac{1}{2}(\mathbf{p} - \mathbf{m})^T \Sigma^{-1}(\mathbf{p} - \mathbf{m})\right) \mathbf{c}$$

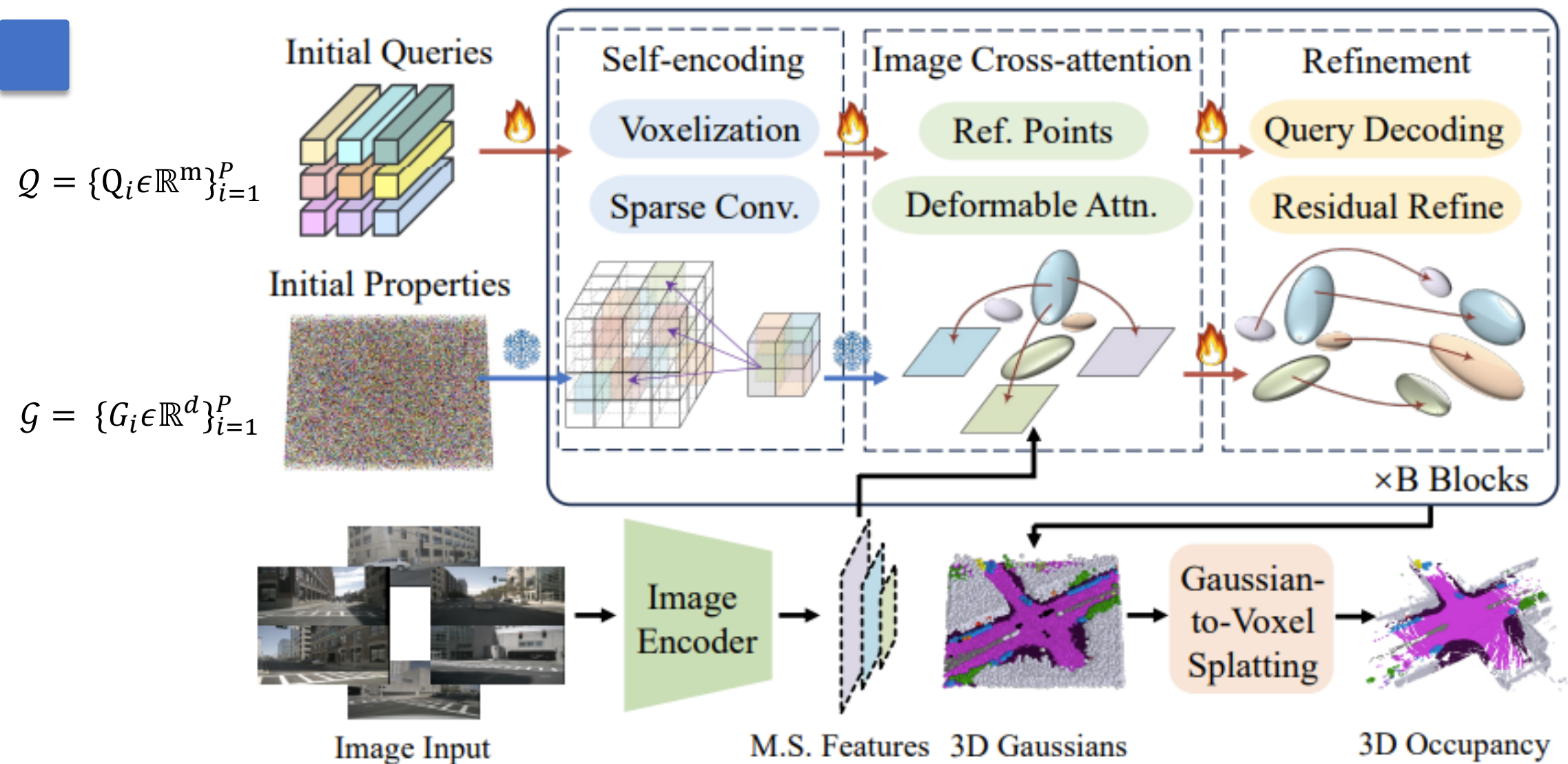
其中

$$\Sigma = \mathbf{R} \mathbf{S} \mathbf{S}^T \mathbf{R}^T, \quad \mathbf{S} = \text{diag}(\mathbf{s}), \quad \mathbf{R} = \mathbf{q}2\mathbf{r}(\mathbf{r})$$

将所有高斯在 \mathbf{p} 处的贡献累加，即得到该点的预测占据及其语义向量：

$$\hat{o}(\mathbf{p}, \mathcal{G}) = \sum_{i=1}^P g_i(\mathbf{p}; \mathbf{m}_i, \mathbf{s}_i, \mathbf{r}_i, \mathbf{c}_i) = \sum_{i=1}^P \exp\left(-\frac{1}{2}(\mathbf{p} - \mathbf{m}_i)^T \Sigma_i^{-1}(\mathbf{p} - \mathbf{m}_i)\right) \mathbf{c}_i$$

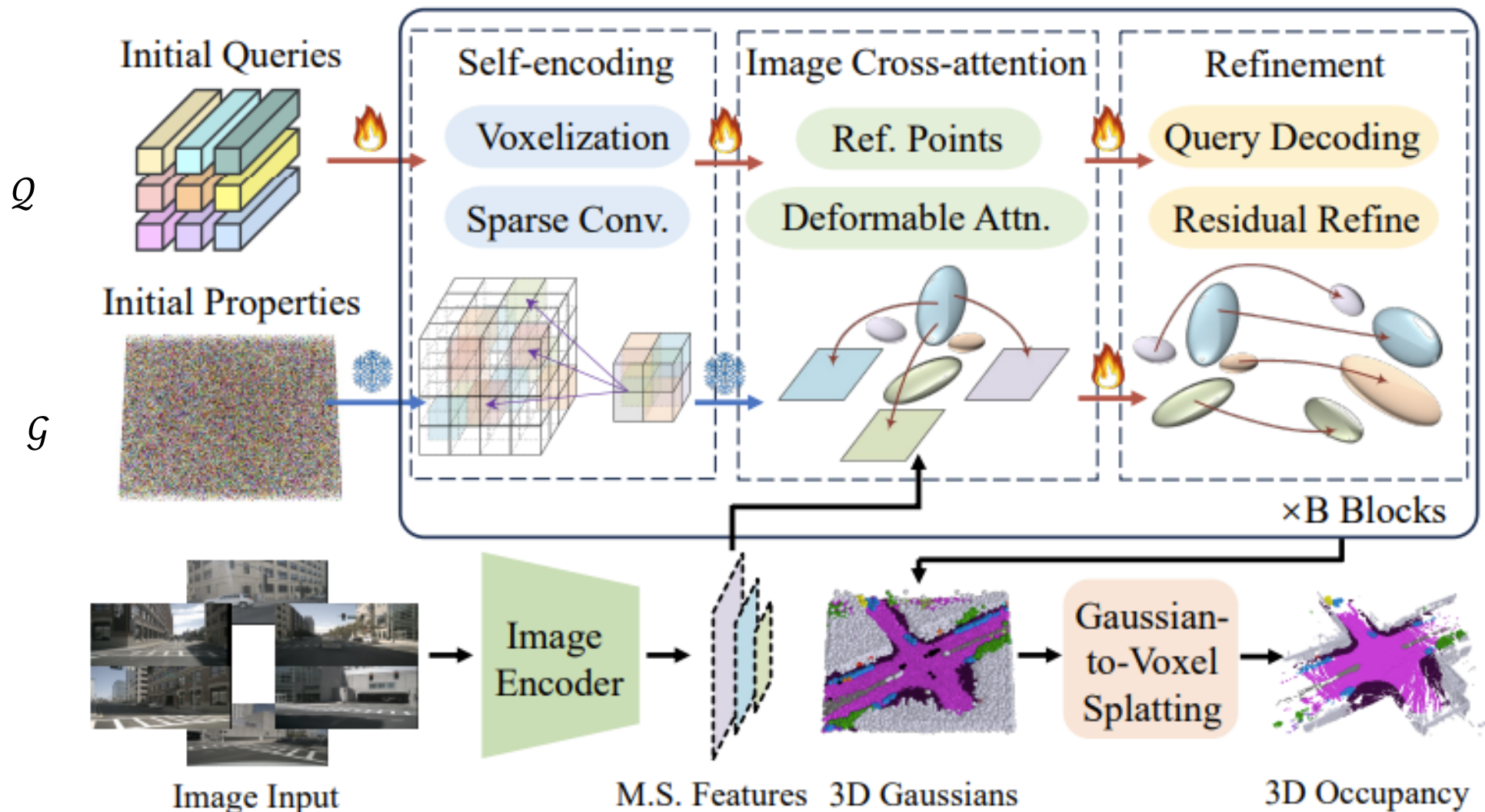




初始化高斯属性和查询向量

将每个三维高斯的属性定义为集合 $G = \{G_i \in \mathbb{R}^d\}_{i=1}^P$

同时初始化一组高维的查询向量 $Q = \{Q_i \in \mathbb{R}^m\}_{i=1}^P$ 用于在后续模块中隐式编码和传递三维信息



优化迭代B个GaussianFormer块

Self-encoding block: 为了使高斯分布之间能够高效交互, 模型首先将每个高斯中心视为一个点, 按其均值 m 生成点云, 并通过体素化, 映射到稀疏网格上, 再在该网格上施加三维稀疏卷积, 由于高斯数量 P 远少于常规模型的体素数量, 稀疏卷积既能利用稀疏性, 又与可变形注意力保持线性计算复杂度。

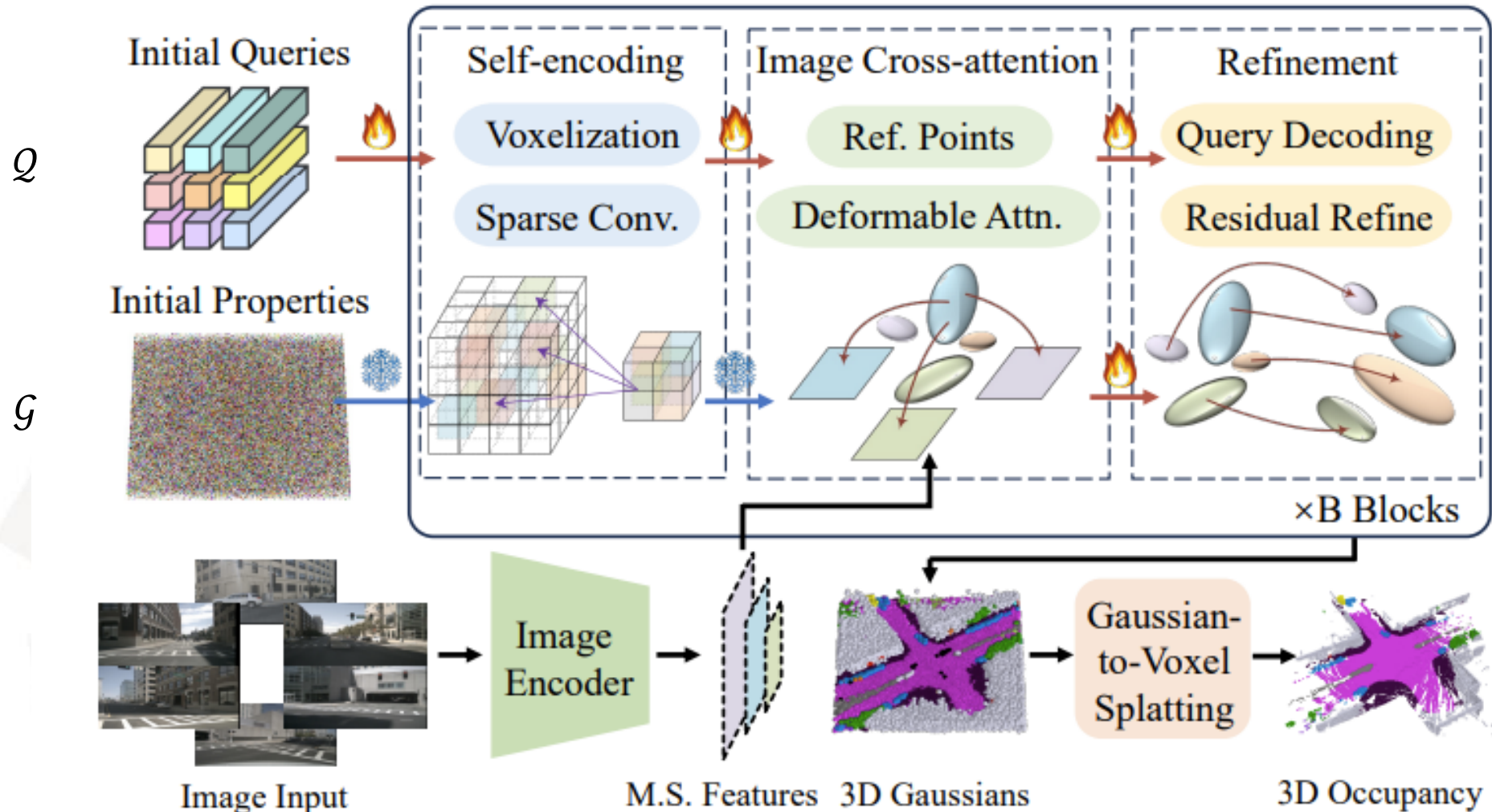
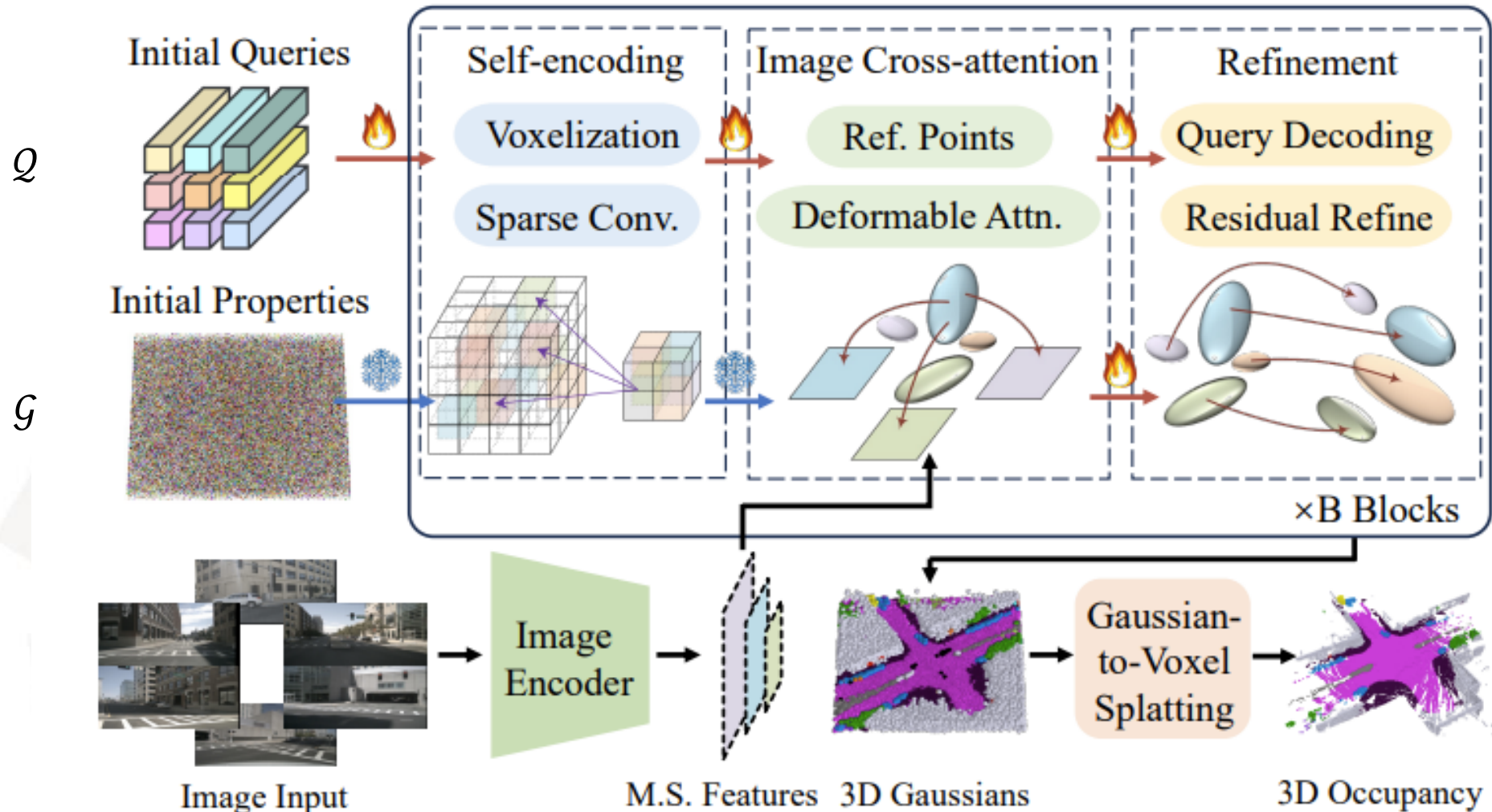


Image Cross-attention block: 用于从多视角图像中提取视觉特征:

1. 对于每个高斯G, 根据其协方差在其均值m周围生成R个参考点集合, $\mathcal{R} = \{m + \Delta m_i\}_{i=1}^R$, 偏移量由协方差决定, 以反映高斯分布形状。
2. 将这些三维参考点利用相机外参T和内参K投影到各视角的图像特征图 F_n 上
3. 通过可变形注意力, 将查询向量Q与投影后的参考点位置 $\pi(\mathcal{R}, T, K)$ 以及对应的图像特征 F_n 交互, 最后对所有视角与参考点聚合求平均: $ICA(\mathcal{R}, Q, \{F_n\}; T, K) = \frac{1}{N} \sum_{n=1}^N \sum_{i=1}^R DA(Q, \pi(\mathcal{R}, T, K), F_n)$



Refinement block: 在自编码和跨注意力更新后，利用更新后的查询向量Q来微调三维高斯的物理属性，以逼近真实的场景分布。

对于高斯 $G=(m,s,r,c)$ ，先通过一个MLP，从对应的查询向量Q解码出一组中间属性。 $\hat{G}=(\hat{m},\hat{s},\hat{r},\hat{c})=MLP(Q)$ 使用中间属性更新原有属性时，将中间的均值 \hat{m} 视为增量残差加到老的均值m上，而对于其他属性则直接替换
 $G_{new}=(m+\hat{m},\hat{s},\hat{r},\hat{c})$

高斯到体素的泼溅

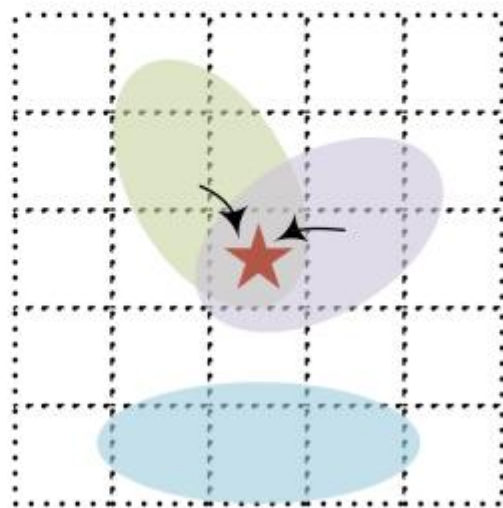
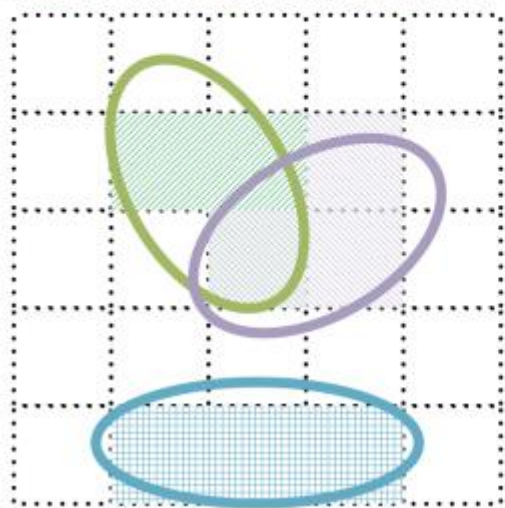
设计了一个高效的“高斯到体素”投溅模块，仅通过局部聚合操作，将三维高斯表示转换为三维占据预测。

$$\hat{o}(\mathbf{p}, \mathcal{G}) = \sum_{i=1}^P g_i(\mathbf{p}; \mathbf{m}_i, s_i, r_i, c_i) = \sum_{i=1}^P \exp\left(-\frac{1}{2}(\mathbf{p} - \mathbf{m}_i)^T \Sigma_i^{-1}(\mathbf{p} - \mathbf{m}_i)\right) c_i$$

上式展示了对高斯的贡献做加权求和的思想，但是如果对每个体素都查询所有的高斯，其计算的复杂度和存储复杂度将达到 $O(XYZ \times P)$ ，这是十分庞大的。

鉴于高斯的分布权重，其实远处的高斯贡献较小，因此我们只需要考虑体素领域内的高斯，忽略远处的高斯。

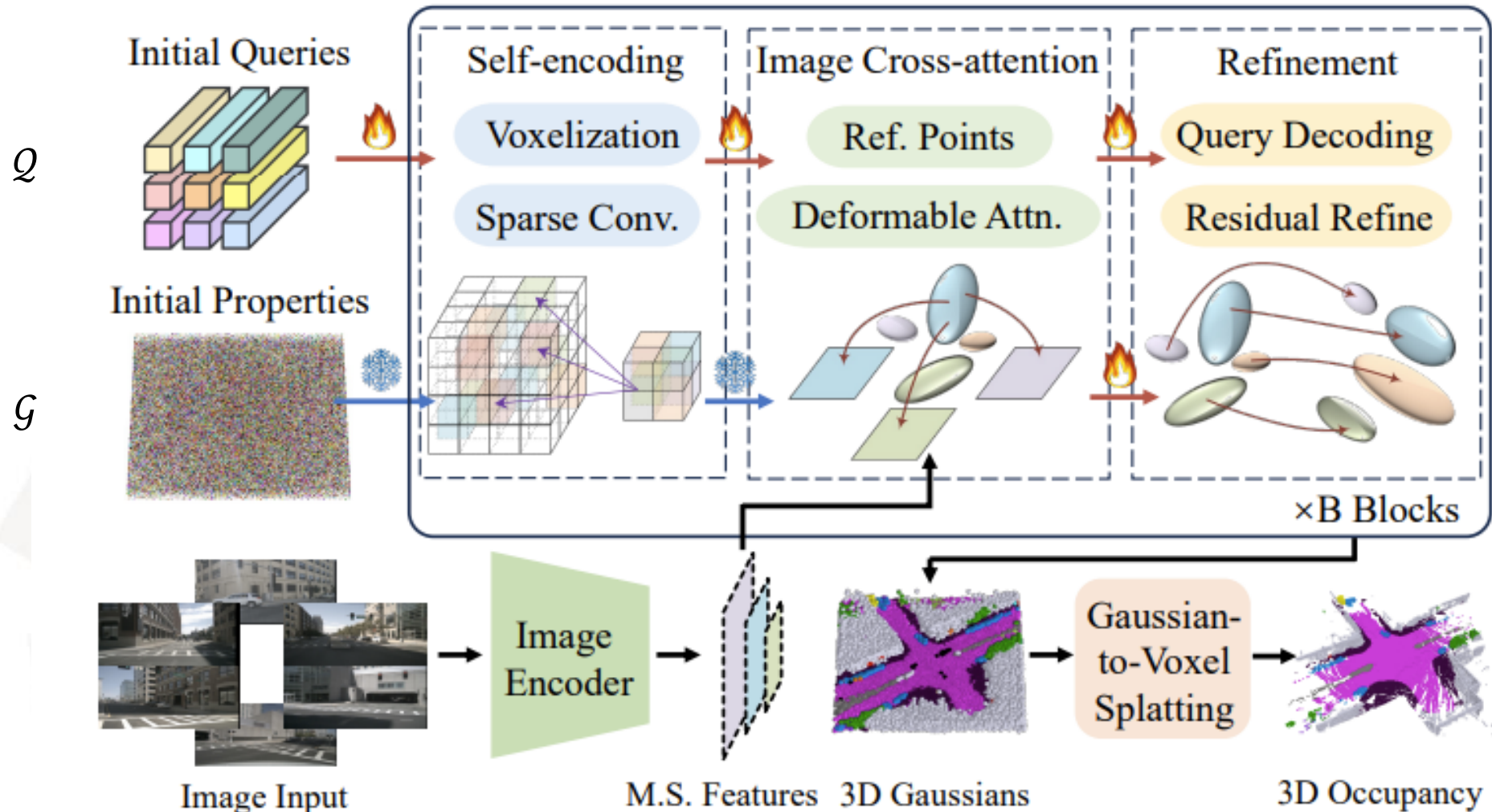
$[(G1,7),(G1,8),(G1,13),$
 $(G2,9),(G2,13),(G2,14),$ $\xrightarrow{\text{Sort}}$ $[\dots,(G1,13),(G2,13),\dots]$
 $(G3,22),(G3,23),(G3,24)]$



首先根据各个高斯的均值 \mathbf{m} ，将其签入到目标的体素网格中(大小为 XYZ)，然后对每个高斯，按照其尺度属性 s 计算领域半径，在该领域内将高斯索引 g 与体素索引 v 组合成元组 (g, v) 加入列表，按照体素索引，对该列表进行排序，可得到每个体素应聚合的高斯索引序列。

$$\text{sort}_{\text{vox}}([(g, v_{g_1}), \dots, (g, v_{g_k})]_{g=1}^P) = [(g_{v_1}, v), \dots, (g_{v_l}, v)]_{v=1}^{XYZ},$$

$$\hat{o}(\mathbf{p}; \mathcal{G}) = \sum_{i \in \mathcal{N}(\mathbf{p})} g_i(\mathbf{p}; \mathbf{m}_i, s_i, r_i, c_i).$$




由于整个GaussianFormer可端到端的训练，因此采用了TPVFormer的训练损失 L :

$$L = \sum_{i=1}^B (L_{ce}^i + L_{lov}^i)$$

nuScenes

Method																		
	SC IoU	SSC mIoU	barrier	bicycle	bus	car	const. veh.	motorcycle	pedestrian	traffic cone	trailer	truck	drive. suf.	other flat	sidewalk	terrain	manmade	vegetation
MonoScene [4]	23.96	7.31	4.03	0.35	8.00	8.04	2.90	0.28	1.16	0.67	4.01	4.35	27.72	5.20	15.13	11.29	9.03	14.86
Atlas [38]	28.66	15.00	10.64	5.68	19.66	24.94	8.90	8.84	6.47	3.28	10.42	16.21	34.86	15.46	21.89	20.95	11.21	20.54
BEVFormer [26]	30.50	16.75	14.22	6.58	23.46	28.28	8.66	10.77	6.64	4.05	11.20	17.78	37.28	18.00	22.88	22.17	13.80	22.21
TPVFormer [16]	11.51	11.66	16.14	7.17	22.63	17.13	8.83	11.39	10.46	8.23	9.43	17.02	8.07	13.64	13.85	10.34	4.90	7.37
TPVFormer* [16]	30.86	17.10	15.96	5.31	23.86	27.32	9.79	8.74	7.09	5.20	10.97	19.22	38.87	21.25	24.26	23.15	11.73	20.81
OccFormer [57]	31.39	19.03	18.65	10.41	23.92	30.29	10.31	14.19	13.59	10.13	12.49	20.77	38.78	19.79	24.19	22.21	13.48	21.35
SurroundOcc [50]	31.49	20.30	20.59	11.68	28.06	30.86	10.70	15.14	14.09	12.06	14.38	22.26	37.29	23.70	24.49	22.77	14.89	21.86
Ours	29.83	19.10	19.52	11.26	26.11	29.78	10.47	13.83	12.58	8.67	12.74	21.57	39.63	23.28	24.46	22.99	9.59	19.12

SSCBench-KITTI-360

Method	Input	SC IoU	SSC mIoU																			
				car	bicycle	motorcycle	truck	other-veh.	person	road	parking	sidewalk	other-grnd	building	fence	vegetation	terrain	pole	traf.-sign	other-struct.	other-object	
LMSCNet [43]	L	47.53	13.65	20.91	0	0	0.26	0	0	62.95	13.51	33.51	0.2	43.67	0.33	40.01	26.80	0	0	3.63	0	
SSCNet [44]	L	53.58	16.95	31.95	0	0.17	10.29	0.58	0.07	65.7	17.33	41.24	3.22	44.41	6.77	43.72	28.87	0.78	0.75	8.60	0.67	
MonoScene [4]	C	37.87	12.31	19.34	0.43	0.58	8.02	2.03	0.86	48.35	11.38	28.13	3.22	32.89	3.53	26.15	16.75	6.92	5.67	4.20	3.09	
Voxformer [23]	C	38.76	11.91	17.84	1.16	0.89	4.56	2.06	1.63	47.01	9.67	27.21	2.89	31.18	4.97	28.99	14.69	6.51	6.92	3.79	2.43	
TPVFormer [16]	C	40.22	13.64	21.56	1.09	1.37	8.06	2.57	2.38	52.99	11.99	31.07	3.78	34.83	4.80	30.08	17.51	7.46	5.86	5.48	2.70	
OccFormer [57]	C	40.27	13.81	22.58	0.66	0.26	9.89	3.82	2.77	54.30	13.44	31.53	3.55	36.42	4.80	31.00	19.51	7.77	8.51	6.95	4.60	
Symphonies [18]	C	44.12	18.58	30.02	1.85	5.90	25.07	12.06	8.20	54.94	13.83	32.76	6.93	35.11	8.58	38.33	11.52	14.01	9.57	14.44	11.28	
Ours	C	35.38	12.92	18.93	1.02	4.62	18.07	7.59	3.35	45.47	10.89	25.03	5.32	28.44	5.68	29.54	8.62	2.99	2.32	9.51	5.14	