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

ECCV 2024

Yuanhui Huang[1], Wenzhao Zheng[1,2*], Yunpeng Zhang[3],
Jie Zhou[1], and Jiwen Lu[1†]

[1]Tsinghua University [2]University of California, Berkeley [3]PhiGent Robotics

- Problem/Objective
  - 3D occ + GS
- Contribution/Key Idea
  - *Object-centric* 3D Gaussian Representation
  - Transformation 모델(GaussianFormer) 설계
  - Efficient Gaussian-to-Voxel Splatting 모듈

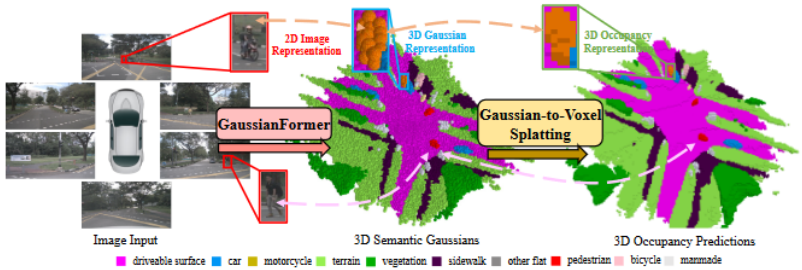
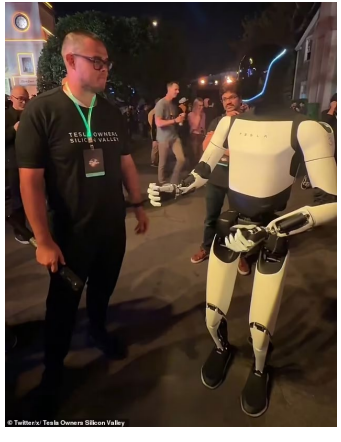


Fig. 1: Considering the universal approximating ability of Gaussian mixture [8, 11], we propose an object-centric 3D semantic Gaussian representation to describe the fine-grained structure of 3D scenes without the use of dense grids. We propose a GaussianFormer model consisting of sparse convolution and cross-attention to efficiently transform 2D images into 3D Gaussian representations. To generate dense 3D occupancy, we design a Gaussian-to-voxel splatting module that can be efficiently implemented with CUDA. With comparable performance, our GaussianFormer reduces memory consumption of existing 3D occupancy prediction methods by **75.2% - 82.2%**.

김범준

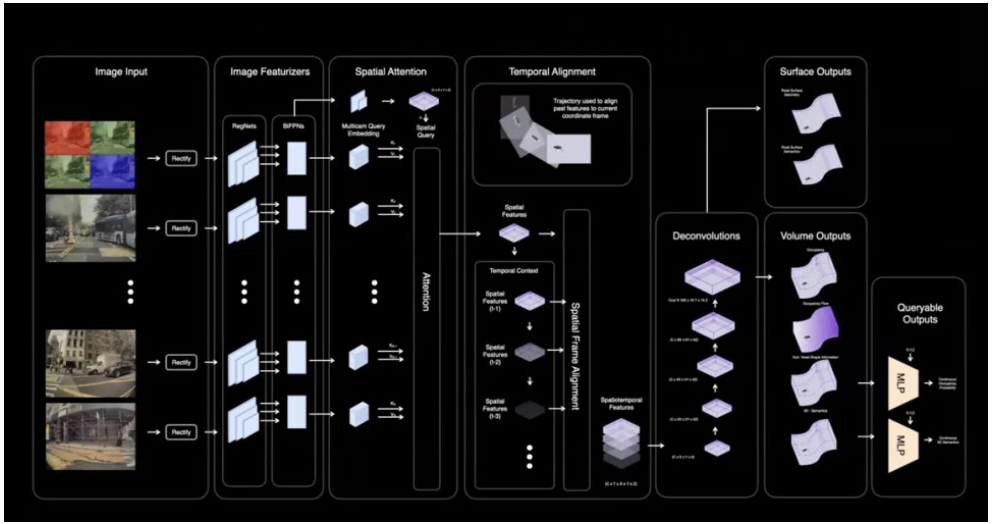
- **3D occupancy prediction이란?**

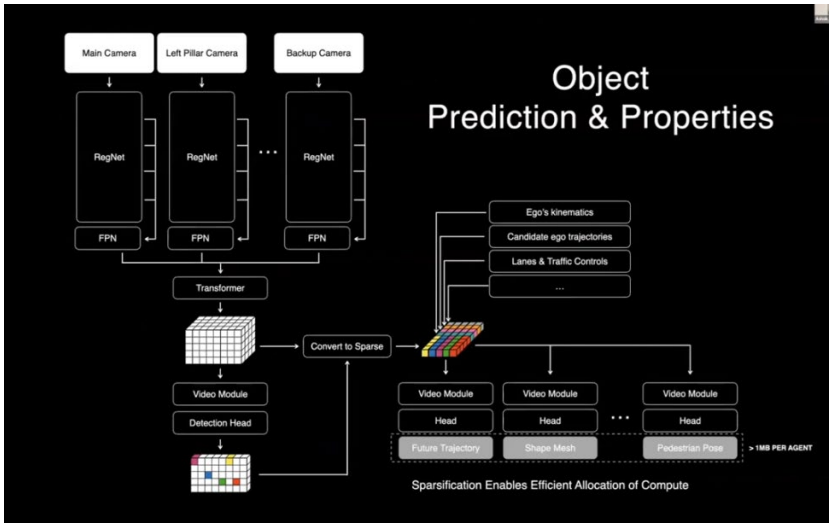
- Tesla가 2022년 CVPR에서 공개한 내용
- 이후 CVPR challenge 등 큰 관심
- 2023, 2024년 CVPR에서는 tesla 휴머노이드 로봇 옵티머스에도 같은 방법이 적용중이라고 밝힘



- Voxel 단위 3차원 BEV segmentation
  - 공간에 대한 점유 여부 + 점유한 공간이 어떤 class인지 분류하는 task

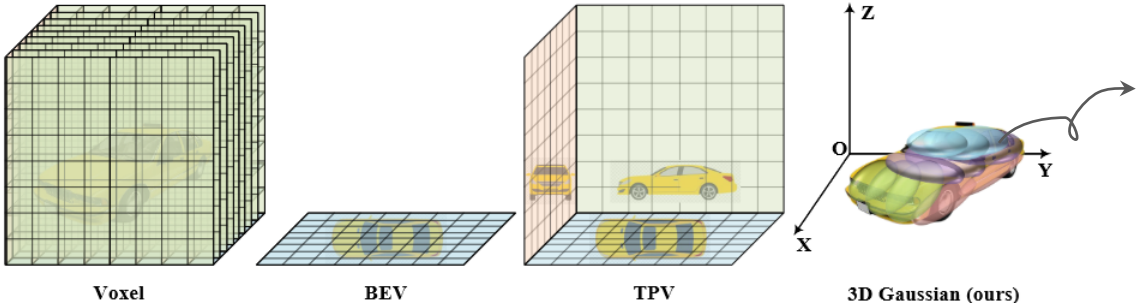
- **OCC by tesla**





[1] 유튜브 링크 - [CVPR'23 WAD] Keynote - Ashok Elluswamy, Tesla
[2] 한글 번역 - [CVPR'23 WAD] Keynote - Ashok Elluswamy, Tesla

- **Introduction - 기존 3D Occ**



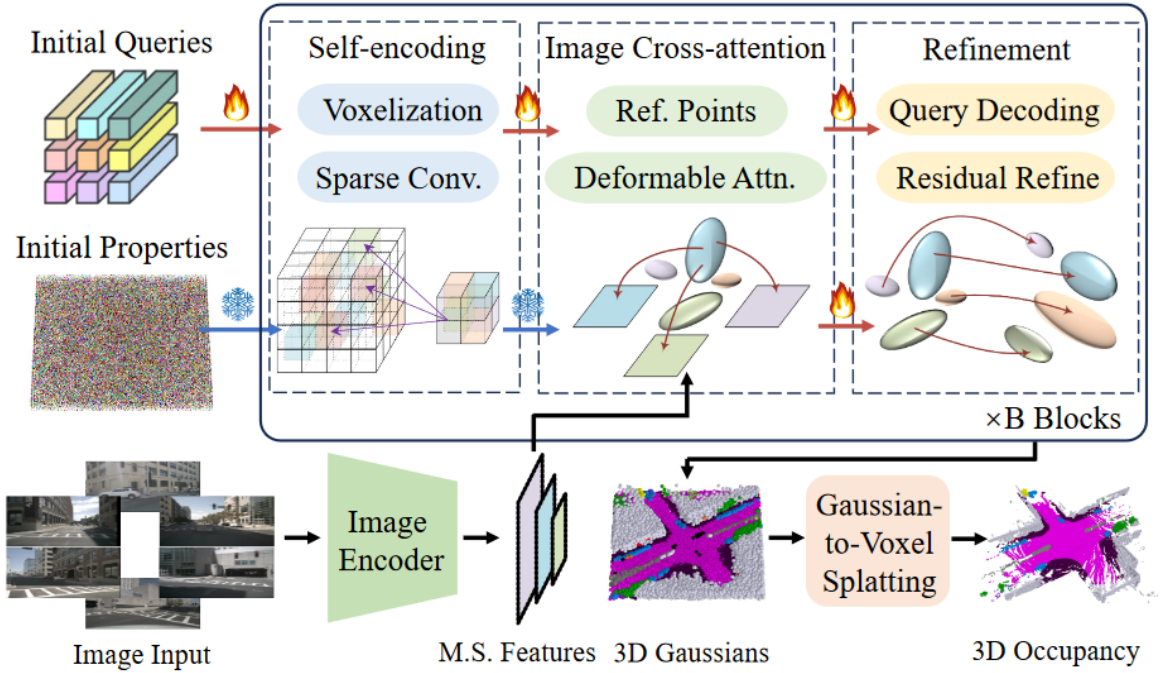
각 Gaussian
: mean, covariance + semantic category

- Voxel/Grid-based Methods (Dense 3D Grids)
  - 대부분의 voxel이 비어있음
    - resource allocation이 매우 비효율적 / empty grid redundancy가 심함
    - 모든 위치에 동일한 자원 할당 → 불균형적이고 낭비가 큼

- Planar-based Methods (BEV, TPV 등)
  - 3D 정보를 2D plane(BEV, TPV 등)으로 투영 → 연산 **효율 높임**
    - 차원을 줄이는 과정에서 정보 loss
    - **본질적으로 grid 기반이기 때문에 빈 영역 redundancy 문제에서 완전히 자유롭지 못함(한계점 극복 x)**

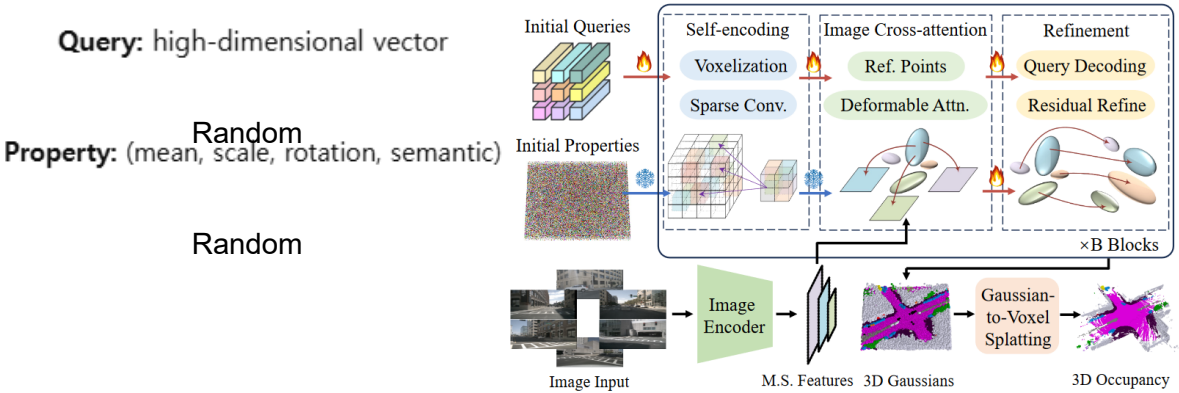
→ 실제로는 grid가 아닌 **object 단위**로 움직임이 일어나기에 이에 집중 (그래서 본 연구에서는 object-centric gaussian을 사용)

- **Method - Object-centric 3D Gaussian Representation**



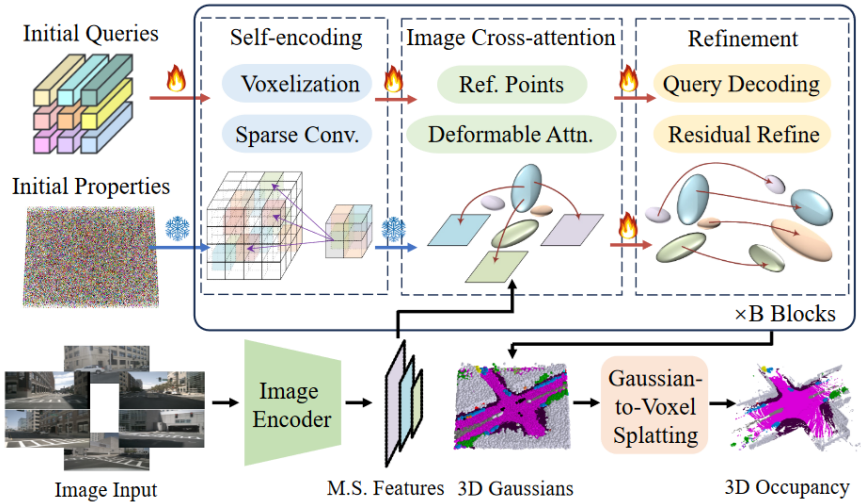
- Scene 내 object-centric으로 sparse하고 유연한 gs 사용
  - Dense grid 대비 훨씬 적은 수의 element 사용
  - object scale과 complexity에 유연하게 표현력 동적 조절

- **Method**



- Self-encoding / Image Cross-attention
  - **Query vector**들이 서로 상호작용 → 이미지에서 정보 get + 다른 가우시안과도 정보를 주고받음
  - query vector가 계속 업데이트 → 이 gaussian은 scene의 어디쯤 있고 환경이 이렇다를 학습
- Refinement
  - **Refinement 모듈**에서 현재 업데이트된 query vector를 **MLP에 넣어서** property들을 "예측"함
  - 여기서 얻어진 예측값을 기존 property를 사용
    - mean은 residual로 더함
    - scale, rotation, semantic은 계속 대체(update)
- Supervision
  - 각 block마다 property를 이용하여 supervision함 (block은 scene마다 4번 iteration)

- **Method**



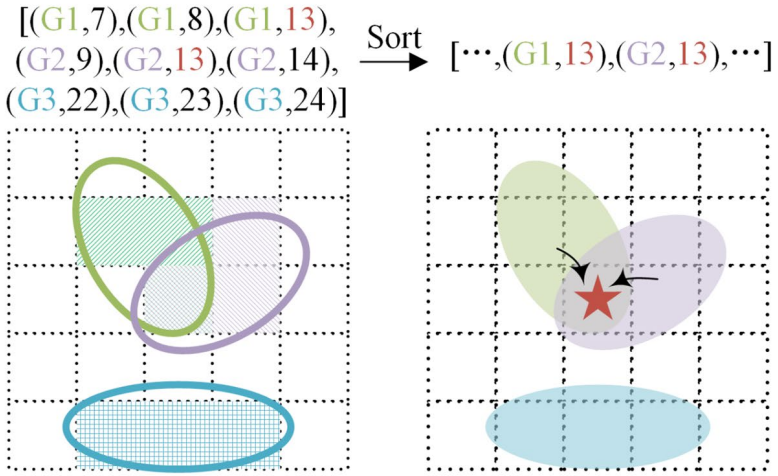
- Voxelization / Sparse-Conv
  - 현재 Gaussian들의 **mean 위치**를 3D voxel grid에 맞춰 배치
  - → 3D sparse Conv 적용을 위해 pointcloud처럼 변환
  - → Gaussian(object단위)의 상호 작용, 정보 교환

- Reference Point / Deform-Attn
  - extrinsic/intrinsic 이용해서 2D 이미지에 투영
  - 각 Gaussian에 대해 mean 기준 reference point 생성(4개)
  - $$\mathrm{ICA}(\mathcal{R}, \mathbf{Q}, \mathbf{F}; \mathcal{T}, \mathcal{K}) = \frac{1}{N} \sum_{n=1}^{N} \sum_{i=1}^{R} \mathrm{DA}(\mathbf{Q}, \pi(\mathcal{R}; \mathcal{T}, \mathcal{K}), \mathbf{F}_n)$$

- Query Decoding
  - query → property 변환 단계
  - update된 Gaussian query에서 mlp로 property 추출

$$\hat{\mathbf{G}} = (\hat{\mathbf{m}}, \hat{\mathbf{s}}, \hat{\mathbf{r}}, \hat{\mathbf{c}}) = \mathrm{MLP}(\mathbf{Q}), \quad \mathbf{G}_{new} = (\mathbf{m} + \hat{\mathbf{m}}, \hat{\mathbf{s}}, \hat{\mathbf{r}}, \hat{\mathbf{c}}).$$

- **Method - Gaussian to Voxel splatting**



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

- 기존 방식대비 효율적인 방법 제시 - *Local aggregation*
  - 기존: 모든 voxel마다 모든 Gaussian의 영향을 다 더함 → 연산량 high
  - → 실제로 영향이 있는 gaussian만 더함
  - → 각 voxel $\text{sort}_{vox}\left([(g, v_{g_1}), ..., (g, v_{g_k})]_{g=1}^{P}\right) = [(g_{v_1}, v), ..., (g_{v_l}, v)]_{v=1}^{XYZ},$

$$\hat{o}(\mathbf{p}; \mathcal{G}) = \sum_{i \in \mathcal{N}(\mathbf{p})} \mathbf{g}_i(\mathbf{p}; \mathbf{m}_i, \mathbf{s}_i, \mathbf{r}_i, \mathbf{c}_i),$$

- **Experiment**

Table 1: 3D semantic occupancy prediction results on nuScenes validation set. While the original TPVFormer [16] is trained with LiDAR segmentation labels, TPVFormer* is supervised by dense occupancy annotations. Our method achieves comparable performance with state-of-the-art methods.

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

- **Experiment**

Table 2: 3D semantic occupancy prediction results on SSCBench-KITTI-360 validation set. Our method achieves performance on par with state-of-the-art methods, excelling at some smaller and general categories (i.e. motorcycle, other-veh.).

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**

- **Experiment**

Table 3: Efficiency comparison of different representations on nuScenes. The latency and memory consumption for GaussianFormer are tested on one NVIDIA 4090 GPU with batch size one, while the results for other methods are reported in OctreeOcc [33] tested on one NVIDIA A100 GPU. Our method demonstrates significantly reduced memory usage compared to other representations.

Methods	Query Form	Query Resolution	Latency ↓	Memory ↓
BEVFormer [26]	2D BEV	200×200	**302** ms	25100 M
TPVFormer [16]	2D tri-plane	200×(200+16+16)	341 ms	29000 M
PanoOcc [49]	3D voxel	100×100 ×16	502 ms	35000 M
FBOCC [27]	3D voxel & 2D BEV	200×200×16 & 200×200	463 ms	31000 M
OctreeOcc [33]	Octree Query	91200	386 ms	26500 M
GaussianFormer	3D Gaussian	144000	372 ms	**6229** M

- **Experiment**

Table 4: Ablation on the components of GaussianFormer. Deep Supervision represents supervising the output of each refinement module. Residual Refine means on which properties of Gaussian to apply residual refinement as opposed to substitution.

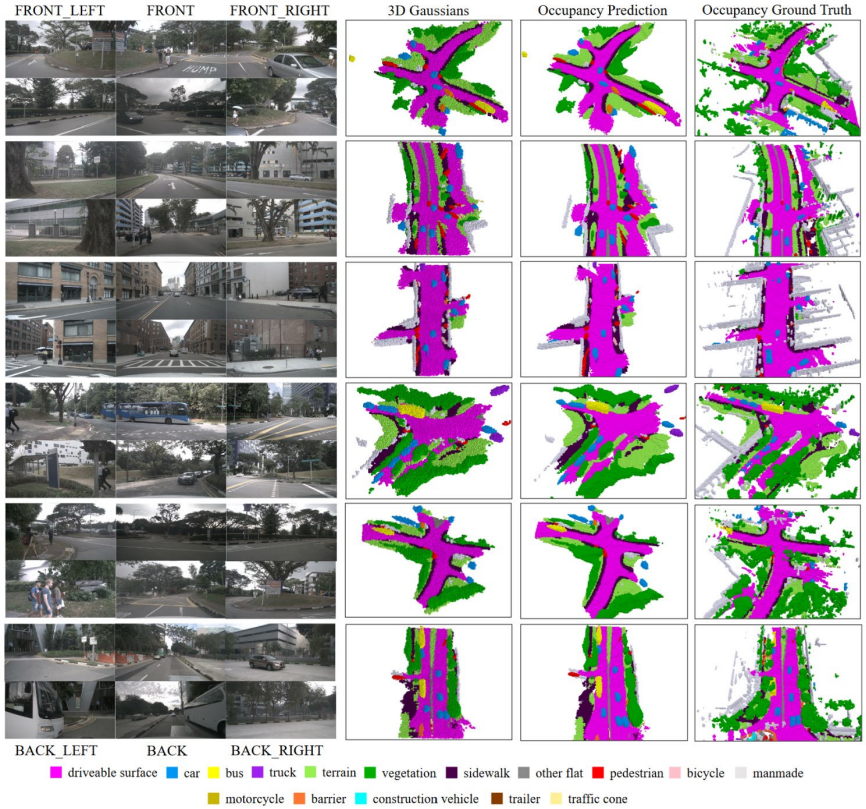
Deep Supervision	Sparse Conv.	Residual Refine	mIoU	IoU
	✓	mean	16.36	29.32
✓		mean	15.93	28.99
✓	✓	none	-	-
✓	✓	all except semantics	16.24	29.30
✓	✓	mean	**16.41**	**29.37**

- **Experiment**

Table 5: Ablation on the number of Gaussians. The latency and memory are tested on an NVIDIA 4090 GPU with batch size one during inference. The performance improves consistently with more Gaussians while taking up more time and memory.

Number of Gaussians	Latency	Memory	mIoU	IoU
25600	**227** ms	**4850** M	16.00	28.72
38400	249 ms	4856 M	16.04	28.72
51200	259 ms	4866 M	16.41	29.37
91200	293 ms	5380 M	18.31	27.48
144000	372 ms	6229 M	**19.10**	**29.83**

- **Experiment**



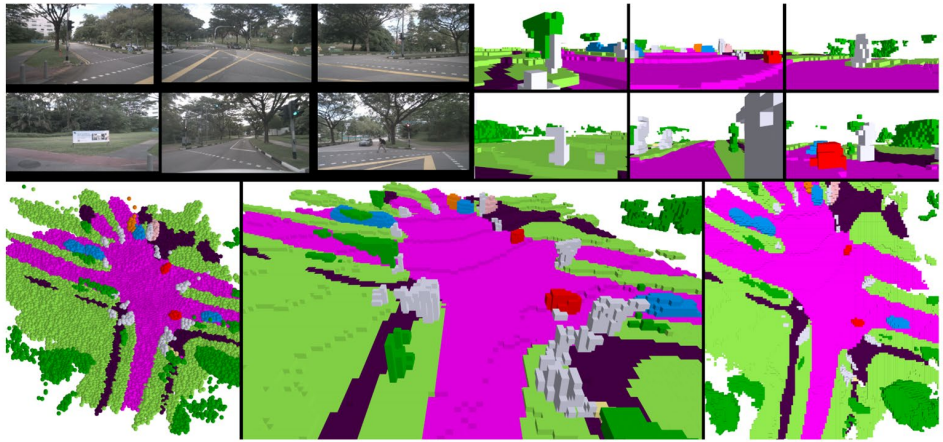


Fig. 6: Visualizations of the proposed GaussianFormer method for 3D semantic occupancy prediction on the nuScenes [3] validation set. We visualize the six input surrounding images and the corresponding predicted semantic occupancy in the upper part. The lower row shows the predicted 3D Gaussians (left), the predicted semantic occupancy in the global view (middle) and the bird's eye view (right).