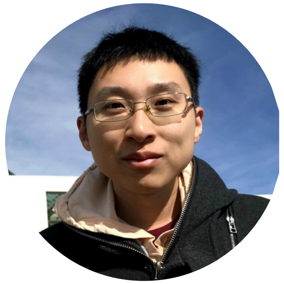
ZPressor: Bottleneck-Aware Compression for Scalable Feed-Forward 3DGS

Weijie Wang[1]✉ Donny Y. Chen[2]✉ Zeyu Zhang[3] Duochao Shi[1] Akide Liu[3] Bohan Zhuang[1]

✉ Corresponding authors.
[1]Zhejiang University [2]ByteDance Seed [3]Monash University
Weijie Wang (王伟杰)

College of Computer Science and Technology, Zhejiang University

2025/6/16

About Me

Weijie Wang (王伟杰)

First-year Ph.D. student@ **ZIP Lab, State Key Lab of CAD&CG, ZJU**

Supervised by **Prof. Bohan Zhuang**

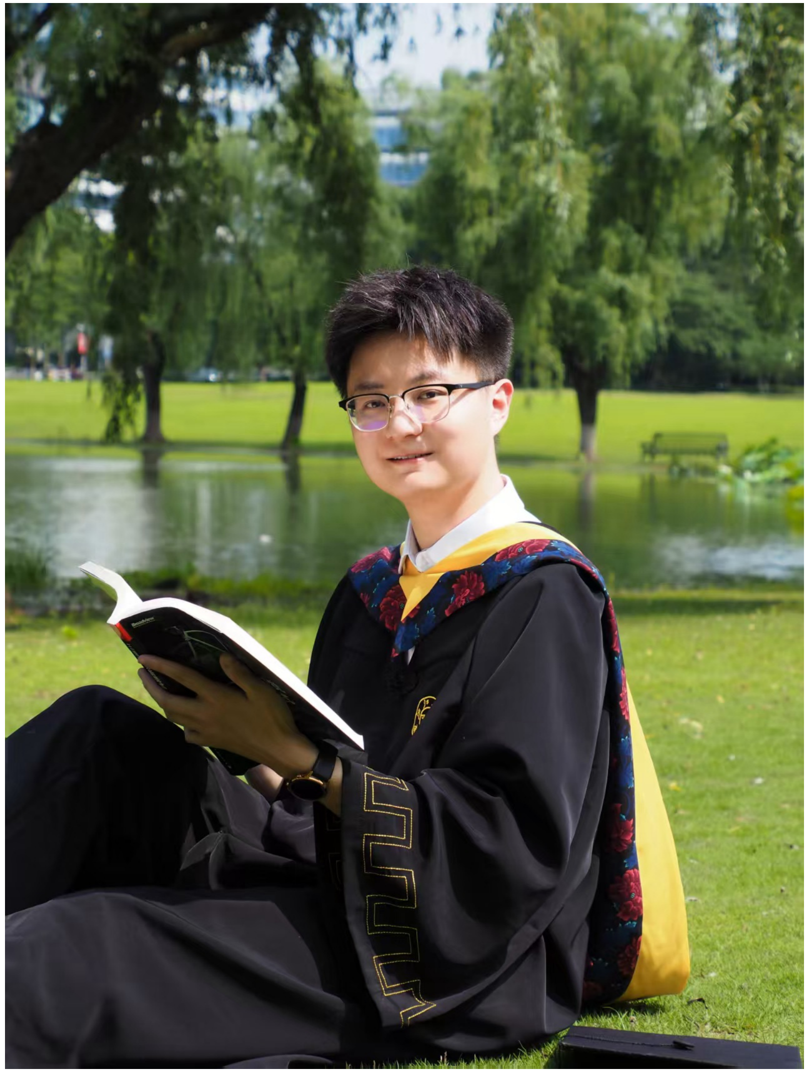
Research Interest:

- **Feed-Forward Reconstruction**: ZPressor, PM-Loss, VolSplat
- **Dynamic Reconstruction**: Street Gaussians, DriveGen3D
- **Interactive Generation**: WonderTurbo

Webpage: https://lhmd.top

Email: wangweijie@zju.edu.cn

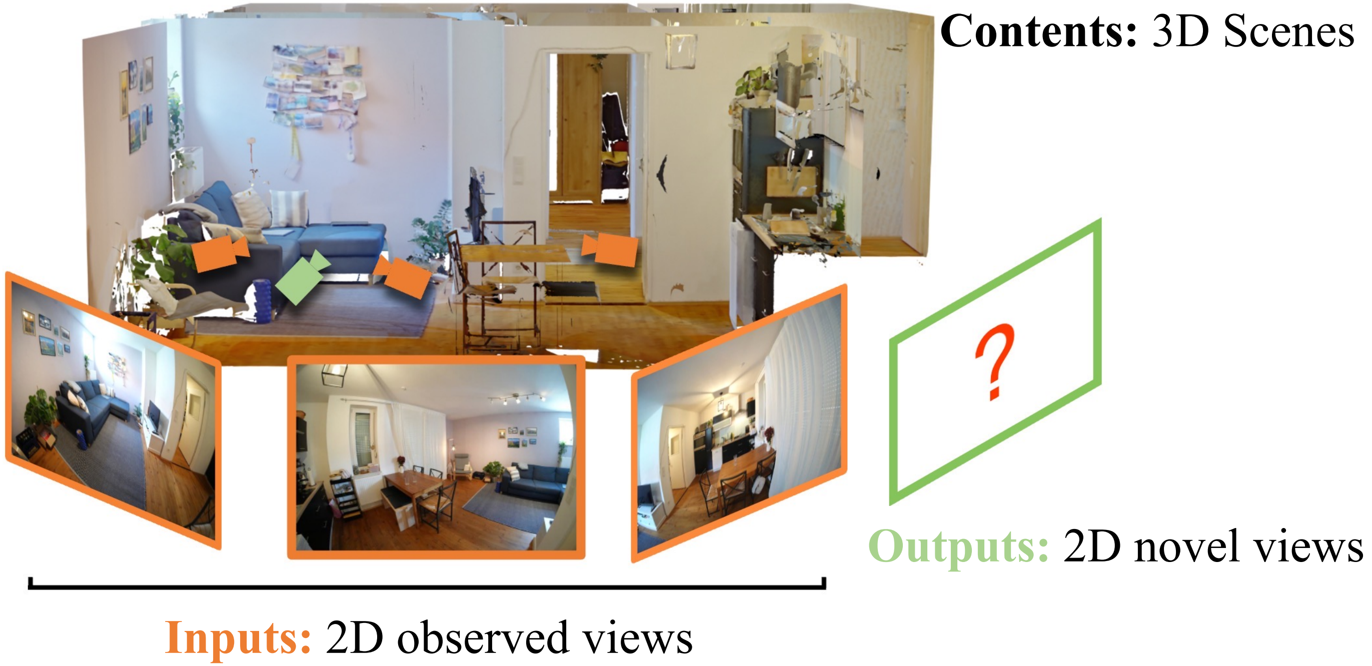
Wechat: zju-lhmd



Background

Tasks

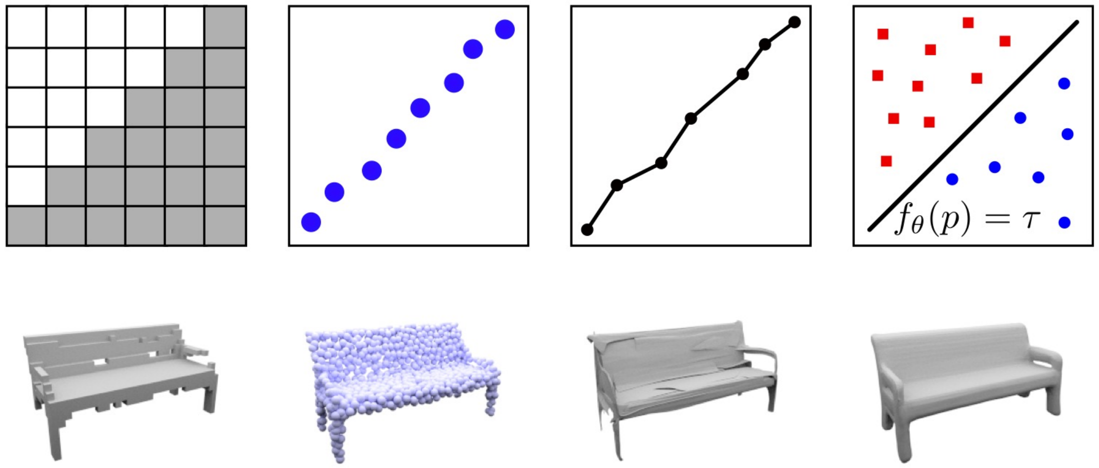
3D Reconstruction

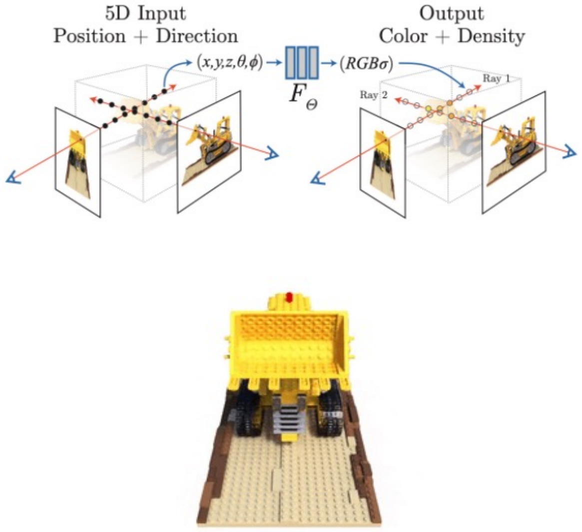


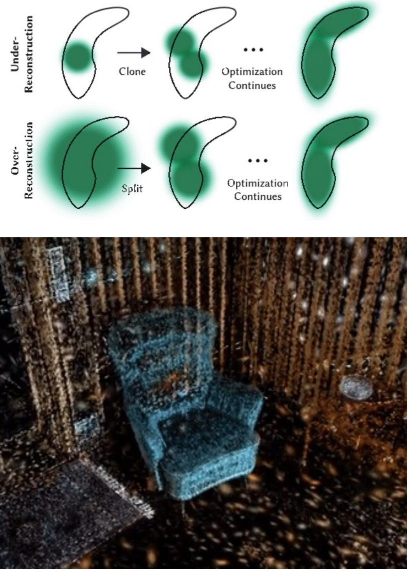
Novel View Synthesis

https://kaldir.vc.in.tum.de/scannetpp/benchmark/nvs
Fast3R: Towards 3D Reconstruction of 1000+ Images in One Forward Pass. CVPR 2025.

3D Representations



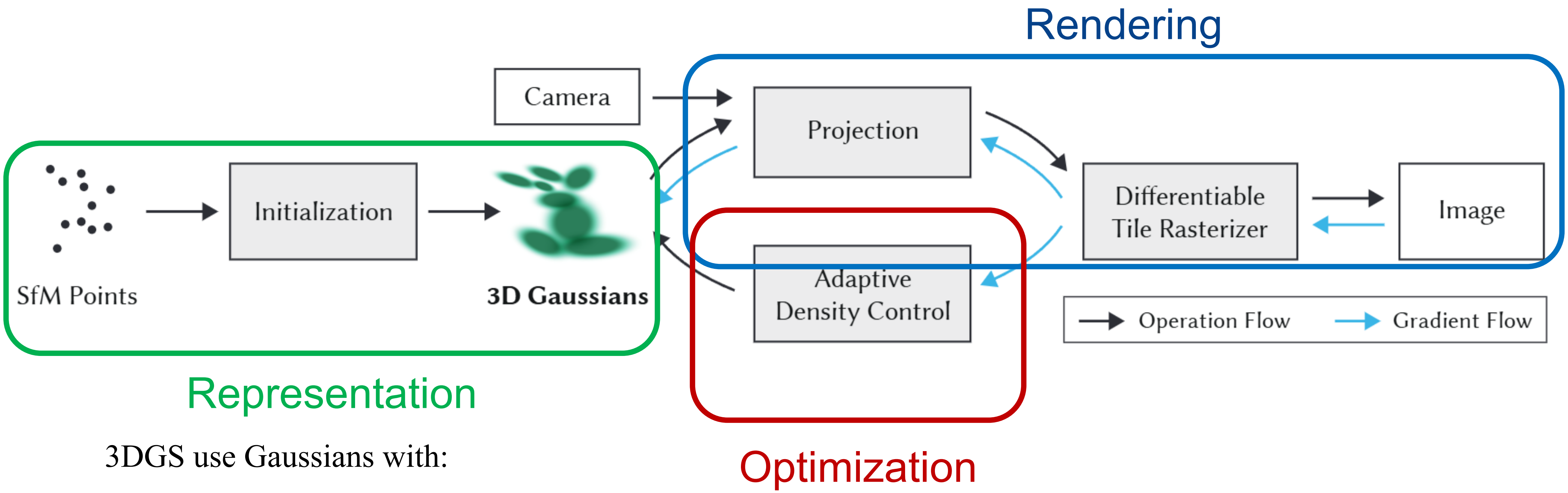




Voxel | Point Cloud | Mesh | Occupancy Networks | Neural Radiance Field (NeRF) | **3D Gaussian Splatting (3DGS)**

There is no canonical representation in 3D. We chose 3DGS since it performs the best for NVS in general.

3D Gaussian Splatting (3DGS)



3DGS use Gaussians with:

μ: Gaussian center position (xyz)
α: opacity; (how transparent)
Σ: covariance; (scale, rotation)
c: color; (spherical harmonic)

3D Gaussian Splatting for Real-Time Radiance Field Rendering. ACM Transactions on Graphics.

Limitations of Per-Scene based 3DGS

1. **Time:** requires applying the optimization process to *each scene* (20+ mins)

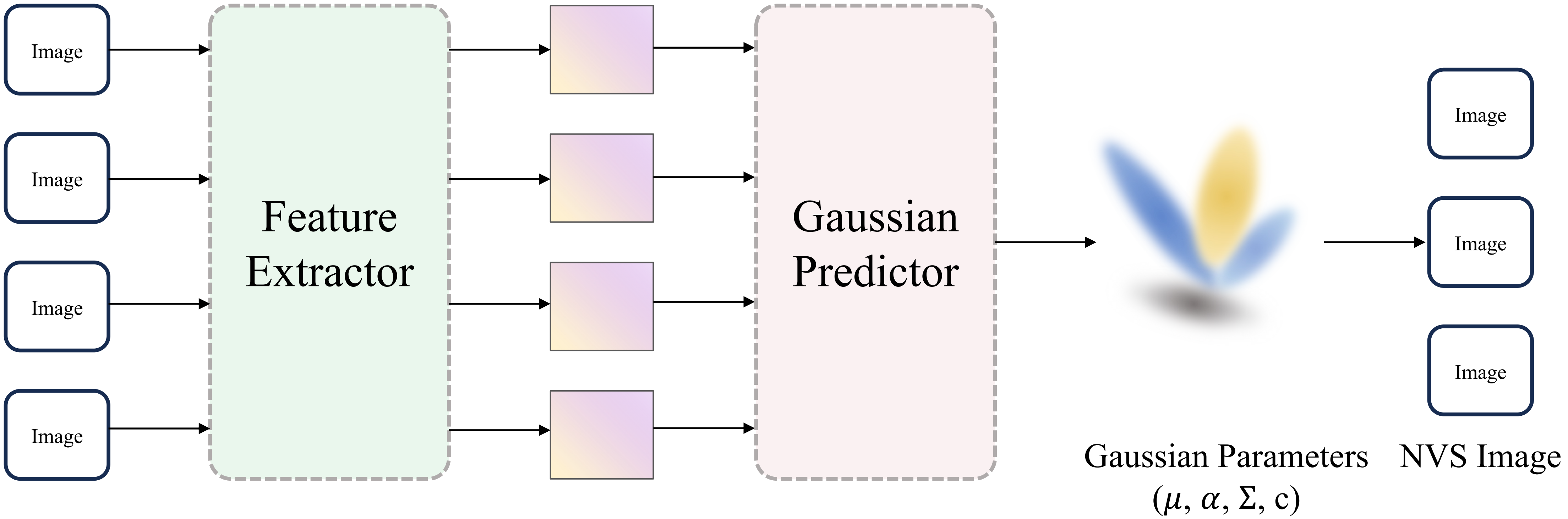
2. **Space:** requires additional permanent storage for the 3D representation of *each scene* (10+ M)

The bicycle scene takes: ~50 mins, ~100 M

Note: Here , we refer to the inria's version of 3DGS;
NOT those improved models such as sparse-view 3DGS, fast-training 3DGS, 3DGS compression, *etc*.

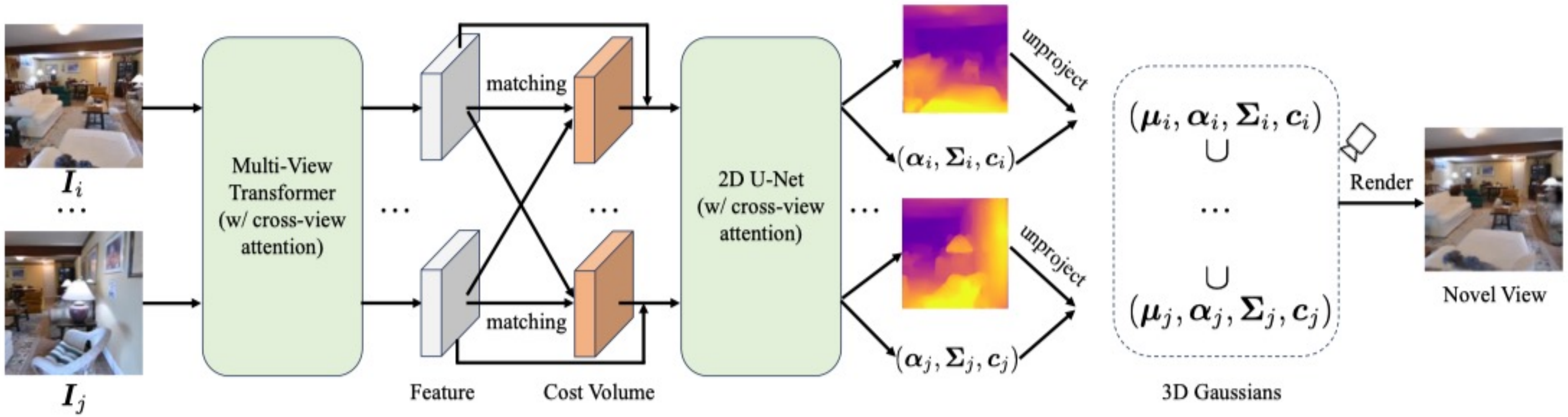
3D Gaussian Splatting for Real-Time Radiance Field Rendering. ACM Transactions on Graphics.

Pipeline of Feed-Forward 3DGS



Almost all feed-forward 3DGS networks use this paradigm.

Example: MVSplat



$$f_{\theta} : \{(I^i, P^i)\}_{i=1}^{K} \mapsto \{(\mu_j, \alpha_j, \Sigma_j, c_j)\}_{j=1}^{H\times W\times K}$$

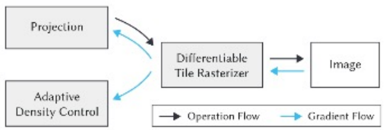
Inputs: Multi-view images, with corresponding camera poses

Outputs: Pixel-align 3D Gaussians for the scenes

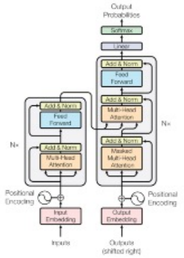
NVS: Render the predicted 3DGS from novel viewpoints

Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images. ECCV 2024.

Per-Scene VS Feed-Forward



Per-Scene Optimization

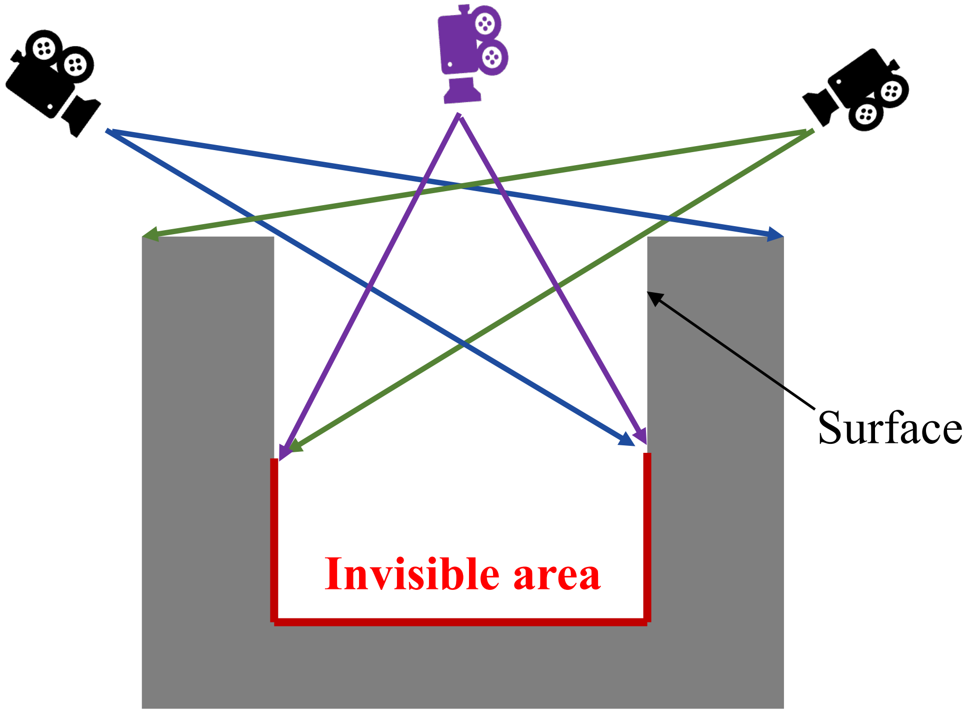


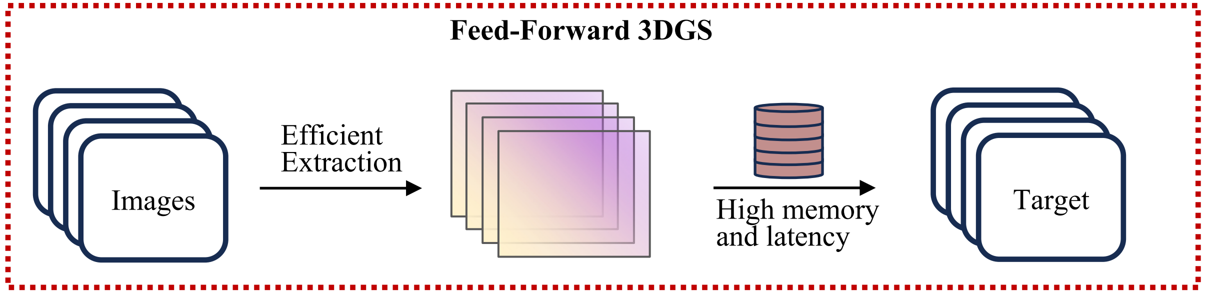
Feed-Forward

Time: 10+ mins
Space: 10+ M

Time: Real-time or a few seconds
Space: Direct inference, no permanent storage

Challenges in Feed-Forward 3DGS





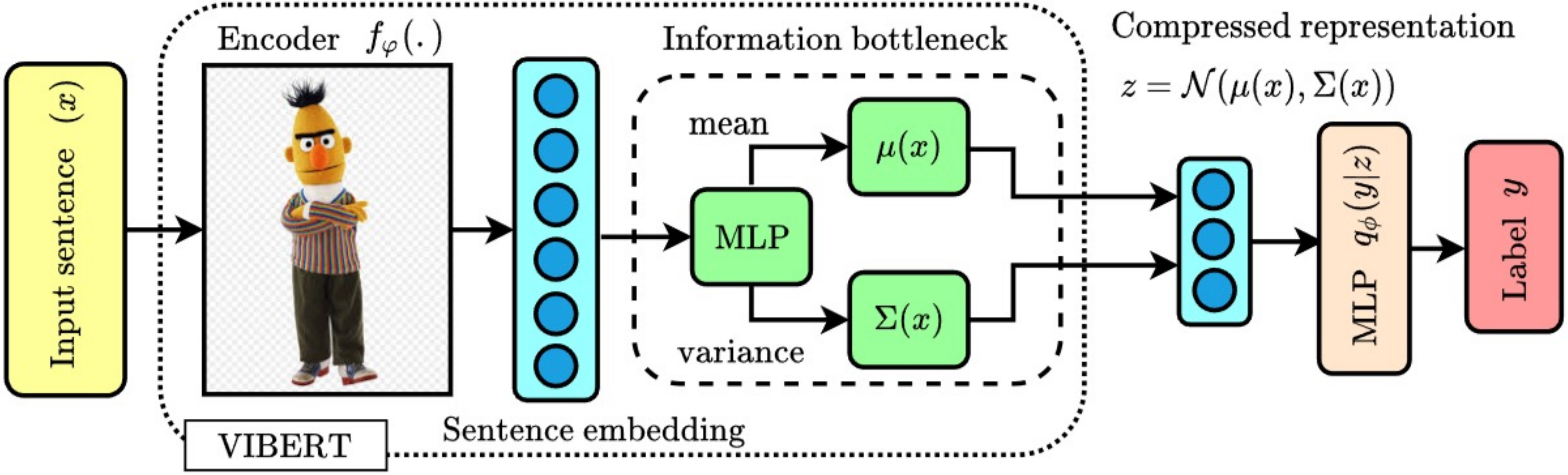
We need denser views to **provide more information**, but at the same time not be influenced by **redundancy**.

The scalability of feed-forward 3DGS is fundamentally constrained by the **limited capacity** of their encoders.

Fast3R: Towards 3D Reconstruction of 1000+ Images in One Forward Pass. CVPR 2025.

ZPressor: Bottleneck-Aware Compression for Scalable Feed-Forward 3DGS

Information Bottleneck Theory

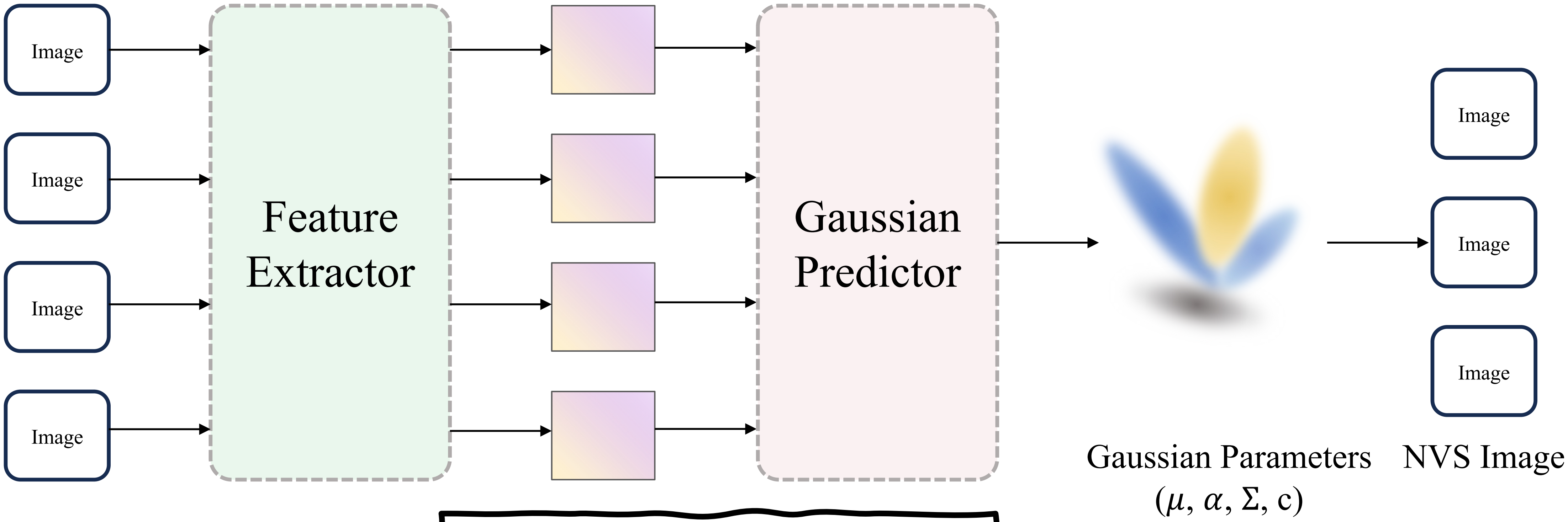


$$I(Z,Y;\boldsymbol{\theta}) = \int dx\, dy\, p(z,y|\boldsymbol{\theta}) \log \frac{p(z,y|\boldsymbol{\theta})}{p(z|\boldsymbol{\theta})p(y|\boldsymbol{\theta})}.^2$$

$$\min_{\mathcal{Z}} IB = \underbrace{\beta\, I(\mathcal{X},\ \mathcal{Z})}_{\text{Compression Score}} - \underbrace{I(\mathcal{Z},\ \mathcal{Y})}_{\text{Prediction Score}}$$

Deep Variational Information Bottleneck. ICLR 2017.
Variational Information Bottleneck for Effective Low-Resource Fine-Tuning. ICLR 2021.

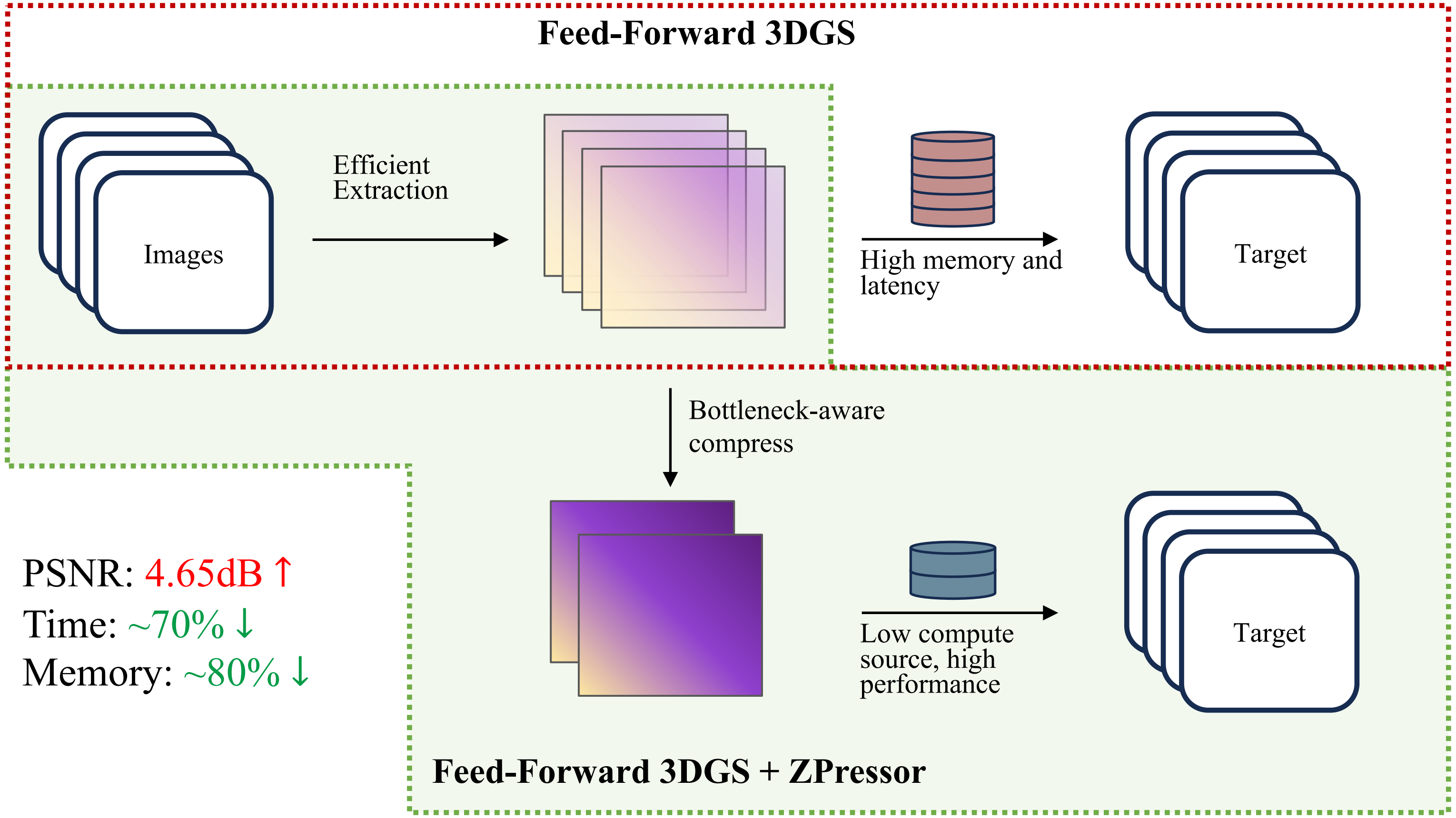
Information Flow in FF 3DGS



$$\sum_{i=1}^{k} H(F_i) \quad \xleftrightarrow{\textbf{Need Latent Space } Z\textbf{!!!}} \quad H(F_1, F_2, \dots, F_k)$$

Significant amount of redundant information

Bottleneck-Aware Compression



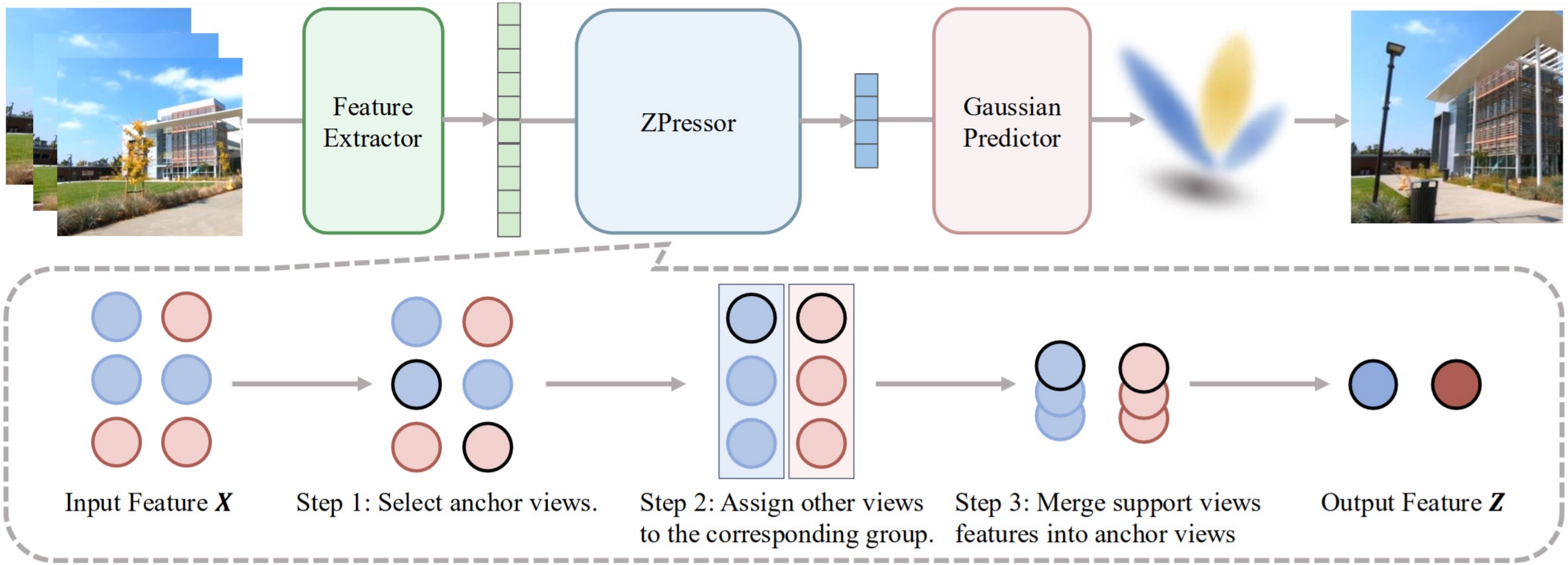
PSNR: 4.65dB ↑
Time: ~70% ↓
Memory: ~80% ↓

$$\min_{\mathcal{Z}} IB = \underbrace{\beta\, I(\mathcal{X}, \mathcal{Z})}_{\text{Compression Score}} - \underbrace{I(\mathcal{Z}, \mathcal{Y})}_{\text{Prediction Score}}$$

1. **Compression Score:** Minimizing I($\mathcal{X}$, $\mathcal{Z}$)
2. **Prediction Score:** Maximizing I($\mathcal{Z}$, $\mathcal{Y}$)

Note: The mutual information (MI) of two random variables I(·,·) is a measure of the mutual dependence between the two variables.

Zpressor: Overview



Anchor View Selection

Support-to-anchor Assignment

Views Information Fusion

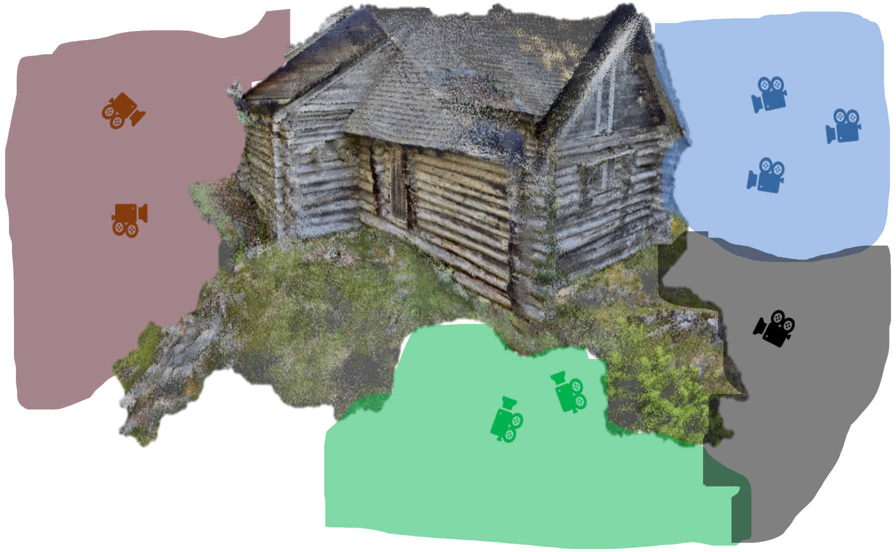
Anchor View Selection

Algorithm 2 Farthest Point Sampling for Anchor View Selection

Input: Set of view camera positions $\mathcal{T} = \{\mathbf{T}_1, \mathbf{T}_2, ..., \mathbf{T}_K\}$, Number of anchor views N
Output: Indices of the selected anchor views $\mathcal{S} = \{\mathbf{T}_{a_1}, \mathbf{T}_{a_2}, ..., \mathbf{T}_{a_n}\}$

```
Initialize the set of anchor view indices S ← ∅
Randomly select a random anchor view T_{a_1} ∈ T, where T_{a_1} ~ Uniform(T)
Add T_{a_1} to S: S ← {T_{a_1}}
for j ← 2 to N do
    Initialize a dictionary to store minimum distances D ← {}
    for k ← 1 to K do
        if k ∉ S then
            Calculate the minimum distance d_k ← min_{i∈S} ||T_k − T_i||_2
            Store the distance: D[k] ← d_k
        end if
    end for
    Find the view position T_{a_j} with the maximum minimum distance: T_{a_j} ← arg max_{k∉S} D[k]
    Add a_j to S: S ← S ∪ {T_{a_j}}
end for
return S
```

Support-to-anchor Assignment



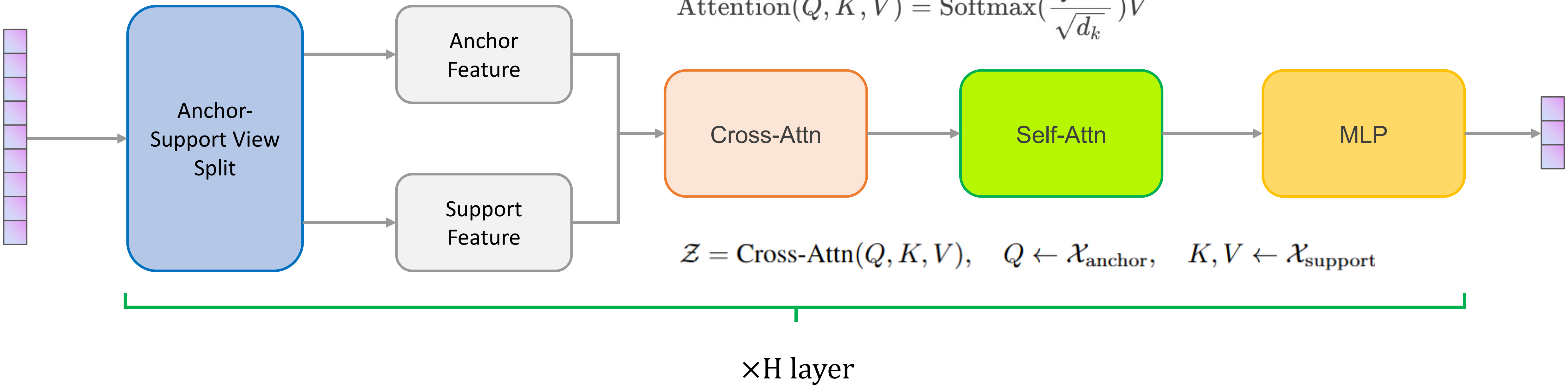
View Groups after Step 1 and Step 2

- Once anchor views are selected, each support view is assigned to its nearest anchor based on **camera position**.
- This grouping ensures that support views, which capture complementary scene details, are paired with **the most spatially relevant** anchor views.
- This pairing thereby ensures the effectiveness of information fusion.
- Formally, the cluster assignment to the i-th anchor view can be denoted as:

$$\mathcal{C}_i = \{f(\mathbf{T}) \in \mathcal{X}_{\text{support}} \mid \|\mathbf{T} - \mathbf{T}_{a_i}\| \leq \|\mathbf{T} - \mathbf{T}_{a_j}\|, \forall j \neq i\}$$

Views Information Fusion

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



$$\mathcal{Z} = \text{Cross-Attn}(Q, K, V), \quad Q \leftarrow \mathcal{X}_{\text{anchor}}, \quad K, V \leftarrow \mathcal{X}_{\text{support}}$$

Design of Feature Fusion Networks. Feature Fusion by Cross-Attention, Self-Attention and MLP.

Results on DL3DV with DepthSplat

Views	Methods	PSNR↑	SSIM↑	LPIPS↓
36 views	DepthSplat	19.23	0.666	0.286
	DepthSplat + ZPressor	**23.88**+4.65	**0.815**+0.149	**0.150**-0.136
24 views	DepthSplat	20.38	0.711	0.253
	DepthSplat + ZPressor	**24.26**+3.88	**0.820**+0.109	**0.147**-0.106
16 views	DepthSplat	22.07	0.773	0.195
	DepthSplat + ZPressor	**24.25**+2.18	**0.819**+0.046	**0.147**-0.047
12 views	DepthSplat	23.32	0.807	0.162
	DepthSplat + ZPressor	**24.30**+0.97	**0.821**+0.014	**0.146**-0.017

Results on RE10K with MVSplat

Views	Methods	PSNR↑	SSIM↑	LPIPS↓
36 views	pixelSplat	OOM	OOM	OOM
	pixelSplat + ZPressor	**26.59**	**0.849**	**0.225**
	MVSplat	24.19	0.851	0.155
	MVSplat + ZPressor	**27.34**+3.15	**0.893**+0.042	**0.113**-0.042
24 views	pixelSplat	OOM	OOM	OOM
	pixelSplat + ZPressor	**26.72**	**0.851**	**0.223**
	MVSplat	25.00	0.871	0.137
	MVSplat + ZPressor	**27.49**+2.49	**0.895**+0.024	**0.111**-0.026
16 views	pixelSplat	OOM	OOM	OOM
	pixelSplat + ZPressor	**26.81**	**0.853**	**0.221**
	MVSplat	25.86	0.888	0.120
	MVSplat + ZPressor	**27.60**+1.74	**0.896**+0.008	**0.110**-0.010
8 views	pixelSplat	26.19	0.852	**0.215**
	pixelSplat + ZPressor	**26.86**+0.67	**0.854**+0.002	0.219+0.004
	MVSplat	26.94	**0.902**	**0.107**
	MVSplat + ZPressor	**27.72**+0.78	0.897-0.005	0.109+0.002

Qualitative comparison

Visualization on DL3DV (36 Input Views)

a62c330f5403e2e41a82a74c4e865b705c5706843b992fae2fe2e538b122d984

63798f5c6fbfcb4eb686268248b8ecbc8d87d920b2bcce967eeaedfd3b3b6d82

Analysis of model efficiency

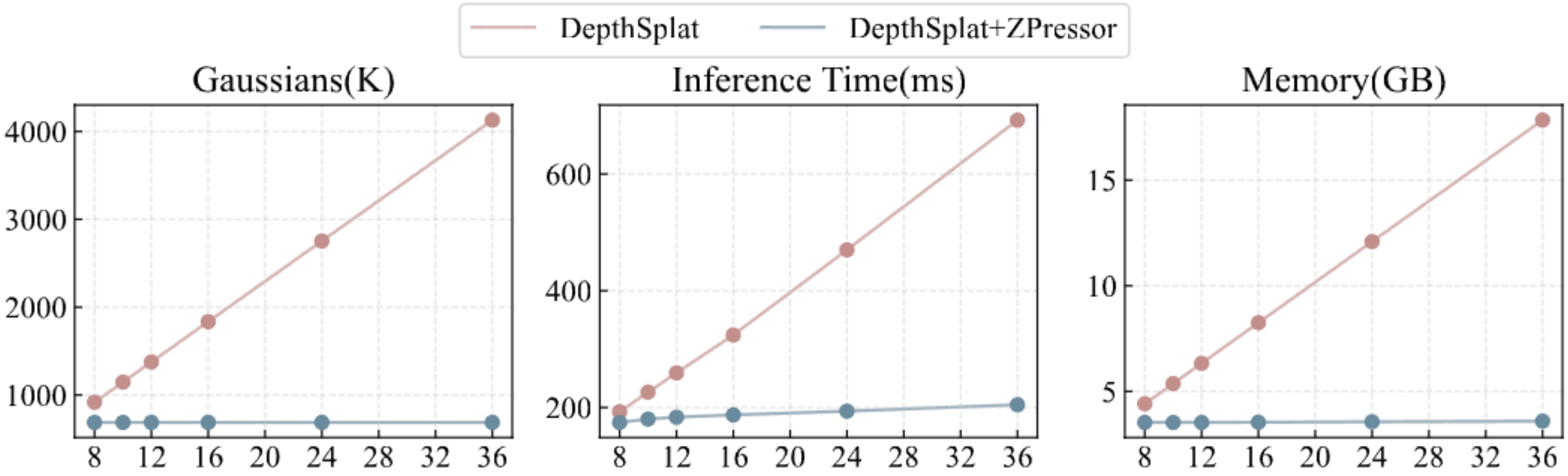


Figure 5: **Efficiency analysis.** We report the number of Gaussians (K), inference time (ms) and peak memory (GB) of DepthSplat [12] and DepthSplat with ZPressor.

Analysis of the Information Bottleneck

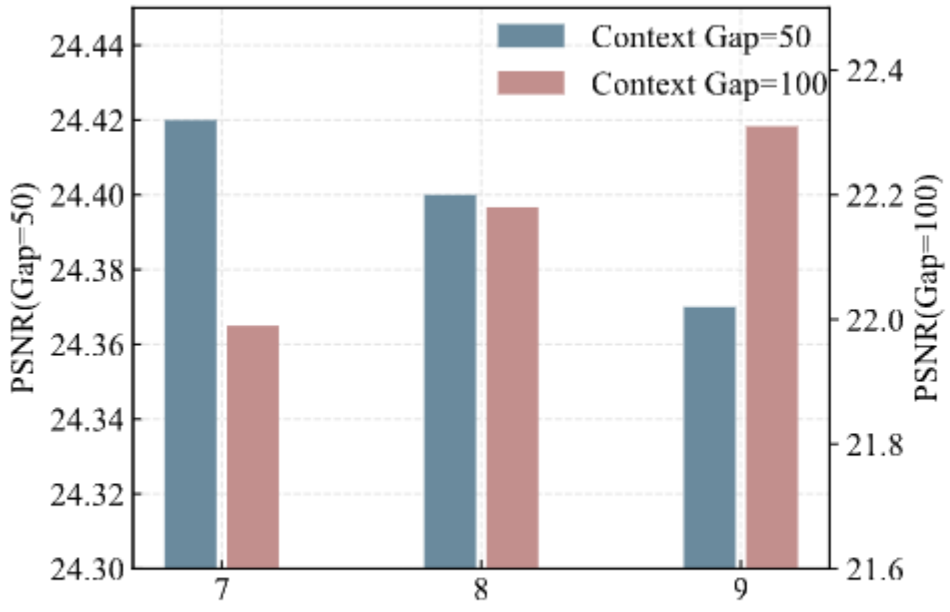
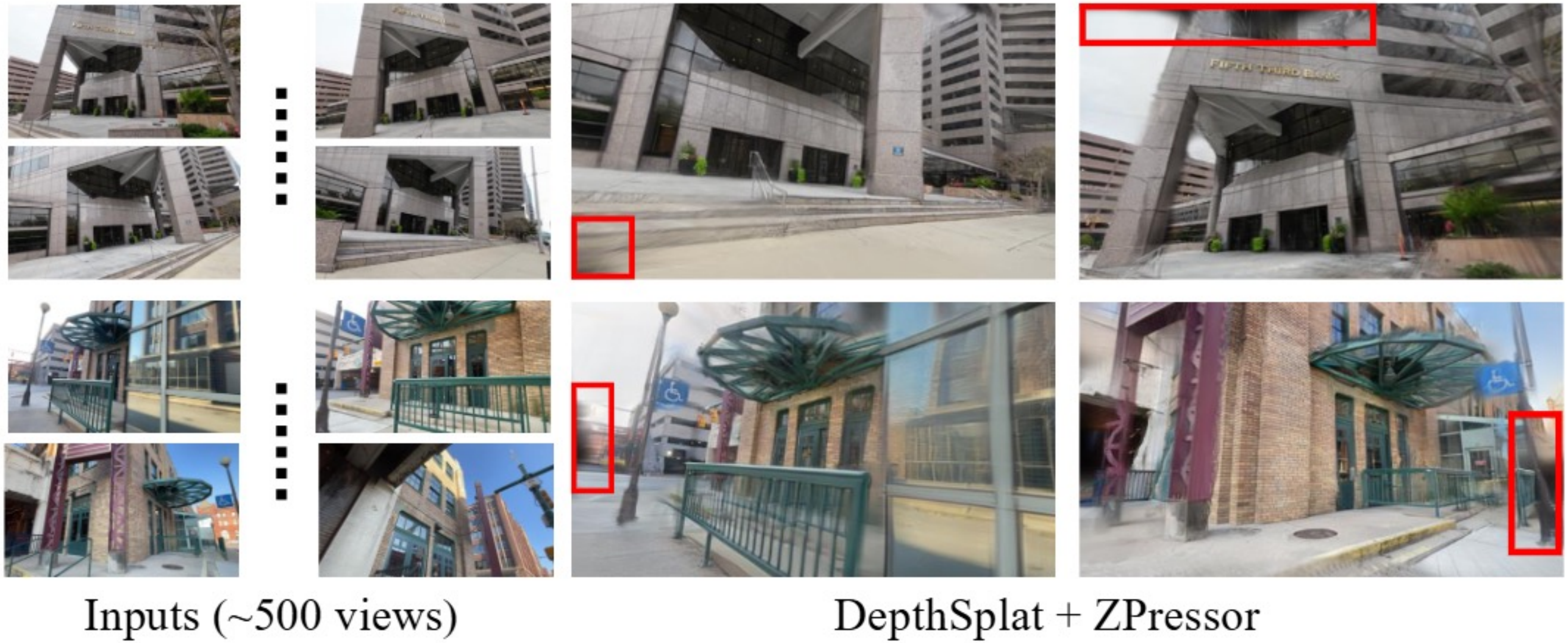


Figure 6: **Analysis of the bottleneck constraint.** We compare the performance of ZPressor in different scale of scene coverage.

Limitations



ZPressor exhibits limitations when processing scenarios with an **extremely high** density of input views.

More Information

Paper, code and model will be available on our project page

ZIP Lab. We are currently recruiting research assistants for 3D LM topic

Weijie Wang's homepage. Actively seeking cooperation

THANK YOU

Weijie Wang (王伟杰)

College of Computer Science and Technology, Zhejiang University

2025/6/16

Cross Dataset Generalization on ACID

Views	Methods	PSNR↑	SSIM↑	LPIPS↓
36 views	pixelSplat	OOM	OOM	OOM
	pixelSplat + Ours	**27.78**	**0.823**	**0.238**
	MVSplat	24.89	0.812	0.179
	MVSplat + Ours	**28.16**+3.27	**0.853**+0.041	**0.145**-0.034
24 views	pixelSplat	OOM	OOM	OOM
	pixelSplat + Ours	**27.91**	**0.825**	**0.235**
	MVSplat	25.46	0.829	0.167
	MVSplat + Ours	**28.33**+2.87	**0.856**+0.027	**0.142**-0.025
16 views	pixelSplat	OOM	OOM	OOM
	pixelSplat + Ours	**27.97**	**0.826**	**0.234**
	MVSplat	26.08	0.844	0.156
	MVSplat + Ours	**28.42**+2.34	**0.858**+0.014	**0.141**-0.015
8 views	pixelSplat	26.69	0.807	0.260
	pixelSplat + Ours	**28.05**+1.36	**0.828**+0.021	**0.234**-0.026
	MVSplat	27.89	**0.864**	**0.140**
	MVSplat + Ours	**28.60**+0.71	0.860-0.004	**0.140**-0.000

Ablation Studies

Table 4: **Ablation study of our method with DepthSplat [12] on the DL3DV dataset [17].** Models are evaluated by rendering eight novel views using 12 input views.

Methods	PSNR↑	SSIM↑	LPIPS↓	Time (s)	Peak Memory (GB)
DepthSplat + ZPressor	**24.30**	**0.821**	**0.146**	0.184	3.80
w/o multi-blocks	24.18	0.817	0.149	**0.140**	**3.79**
w/o self-attention	23.85	0.810	0.156	0.183	3.80
DepthSplat	23.32	0.808	0.162	0.260	6.80

Note: All ablation models and training settings will be available on our GitHub project.