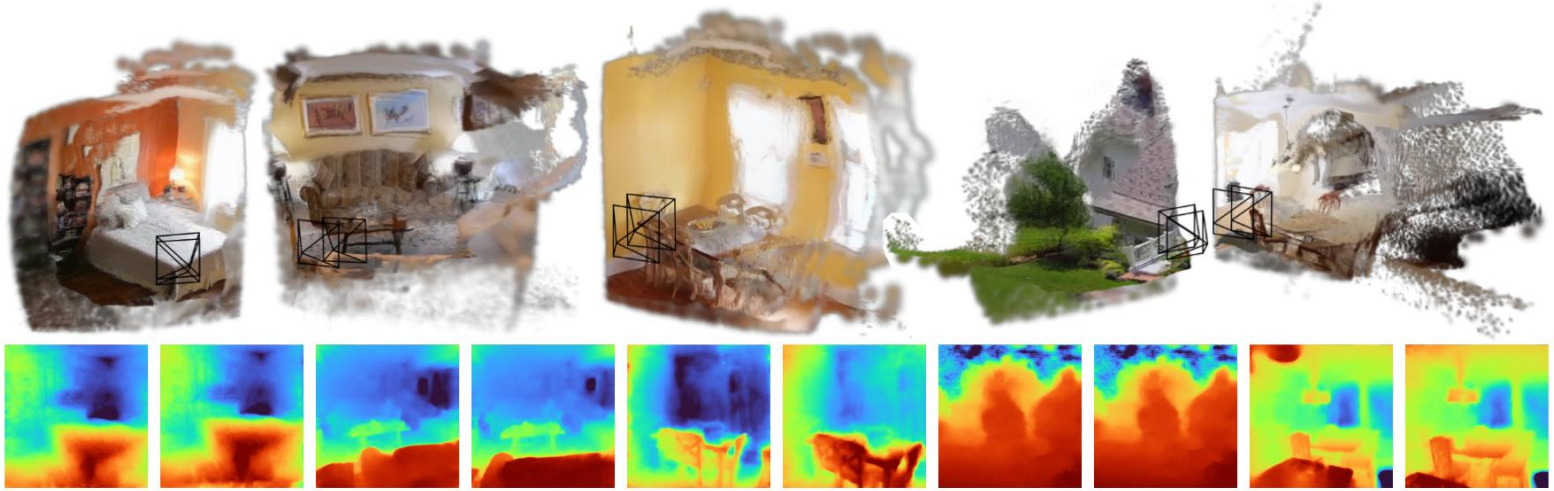
pixelSplat: 3D Gaussian Splats from Image Pairs for Scalable Generalizable 3D Reconstruction

(CVPR 2024 Best Paper Runner-Up)



한국과학기술연구원(KIST)

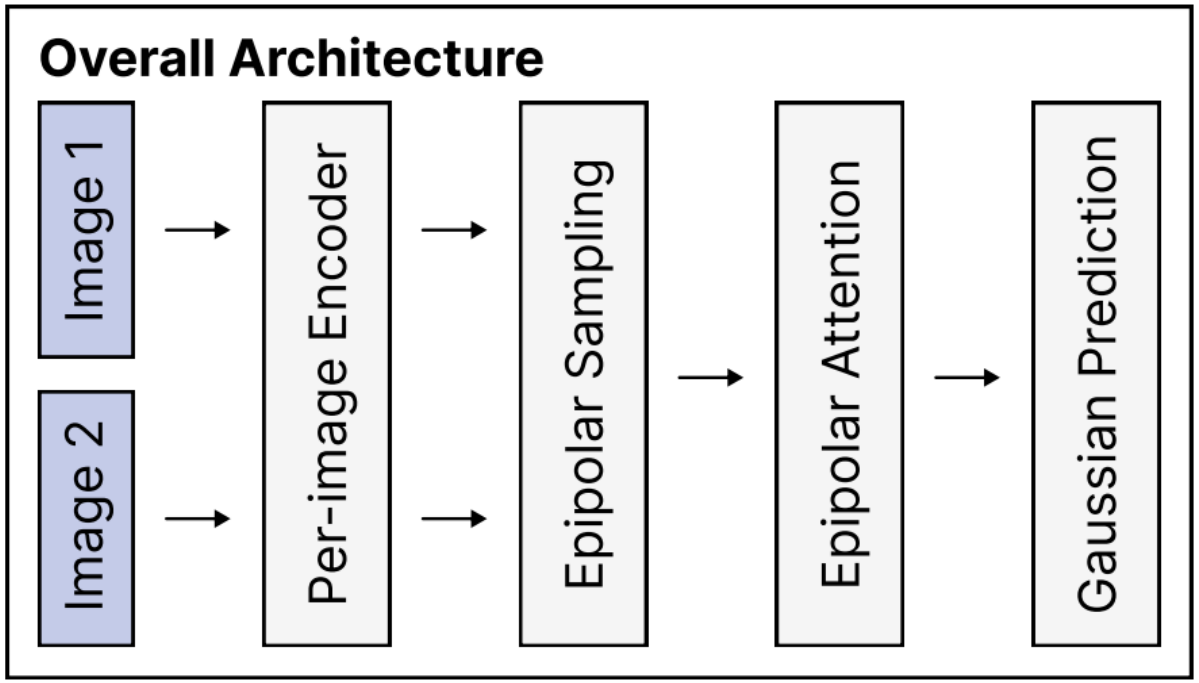
CVIPL 학생연구원 **김연욱**

KIST 한국과학기술연구원
Korea Institute of Science and Technology

Contents

OverView

PixelSplat: Image 쌍으로부터 **3D Gaussian primitives**로 표현된 **3D radiance field**를 재구성하는 **generalizable feed-forward** model



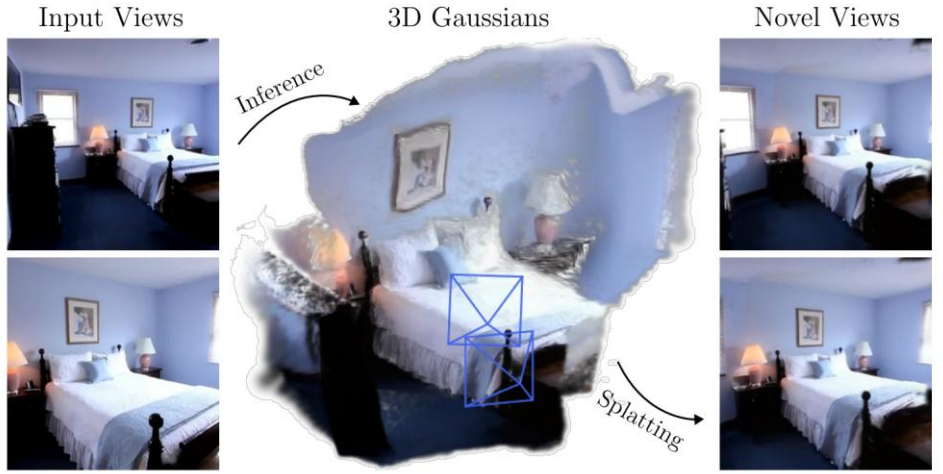
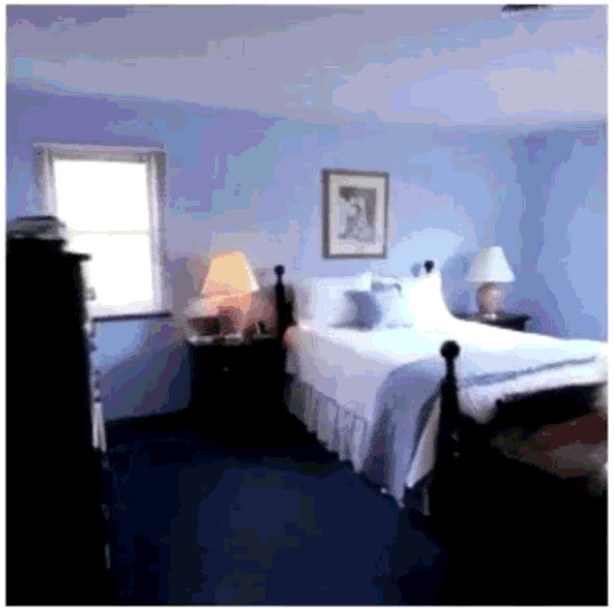


Figure 1. **Overview.** Given a pair of input images, pixelSplat reconstructs a 3D radiance field parameterized via 3D Gaussian primitives. This yields an explicit 3D representation that is renderable in real time, remains editable, and is cheap to train.

OverView

3D Gaussians

Unseen Views



Introduction

Generalizable Novel View Synthesis

- Sparse(handful) images로부터 새로운 시점을 합성하는 일반화 문제를 다룸.
- NeRF, 3D-GS처럼 Per-scene optimization이 아닌, 새로운 장면에 대해서도 새로운 시점 합성.

Current Limitations: Differentiable Volumetric Rendering

- High quality, but notoriously memory and time-intensive
- Differentiable rendering requires evaluating dozens or hundreds of points along each camera ray

Introduction

Light Field Transformers (LFTs)

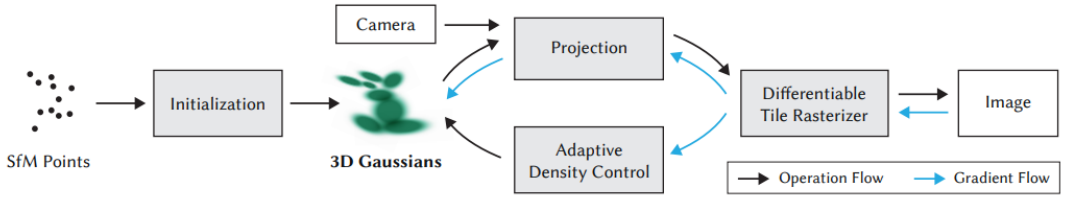
- Ray를 query token으로 embedding
- Color를 image tokens의 cross-attention을 통해 획득
- Faster than NeRF

But Real-time rendering, Interpretable and editable X

3D Gaussian Splatting (3D-GS)

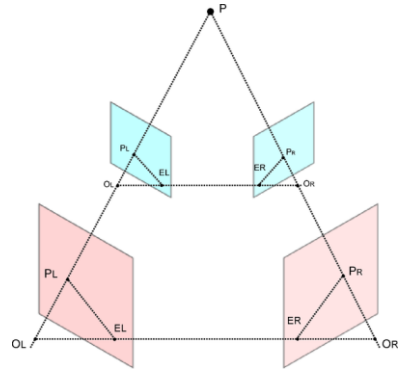
- Rasterization 기반으로 Real-time rendering
- Gaussian primitives기반으로 Interpretable and editable

But single-scene novel view synthesis, lots of input images



Introduction

PixelSplat: Generalizable, real-time rendering



Challenges

- **Scale Ambiguity**

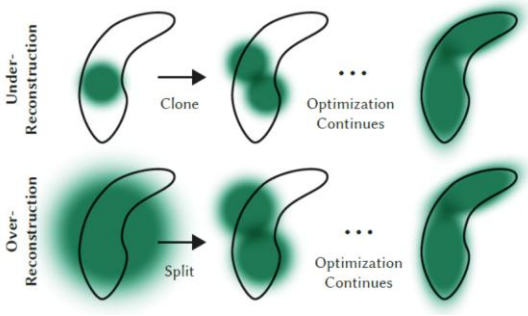
  Real-world datasets의 카메라는 SfM으로 얻어져 임의적 Scale (Up-to-scale).
  Single image로는 깊이의 절대적인 스케일을 추론 불가.

  → **Multi-view encoder**도입

- **Local Minima**

  Sparse and Locally supported Gaussian primitives.
  Gradient Descent으로는 정답 위치로 이동하지 못하고 local minima에 빠지기 쉬움.
  3D-GS의 Pruning/Splitting은 non-differentiable하여 End-to-End 부적합.

  → propose a **differentiable method**



Preliminaries

3D Gaussian Splatting

Rendering technique that represents 3D scenes using a set of learnable 3D Gaussian primitives.

- can be rendered via an **inexpensive rasterization** operation. → cheaper in terms of time and memory
- Has a key challenge a.k.a. **Local minima**

- $\boldsymbol{\mu}_k$: mean
- $\boldsymbol{\Sigma}_k$: covariance
- $\boldsymbol{\alpha}_k$: opacity
- $\mathbf{S}_k$: coefficient

$$\{\mathbf{g}_k = (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \boldsymbol{\alpha}_k, \mathbf{S}_k)\}_k^K$$

전체 가우시안 수

K번째 gaussian primitive

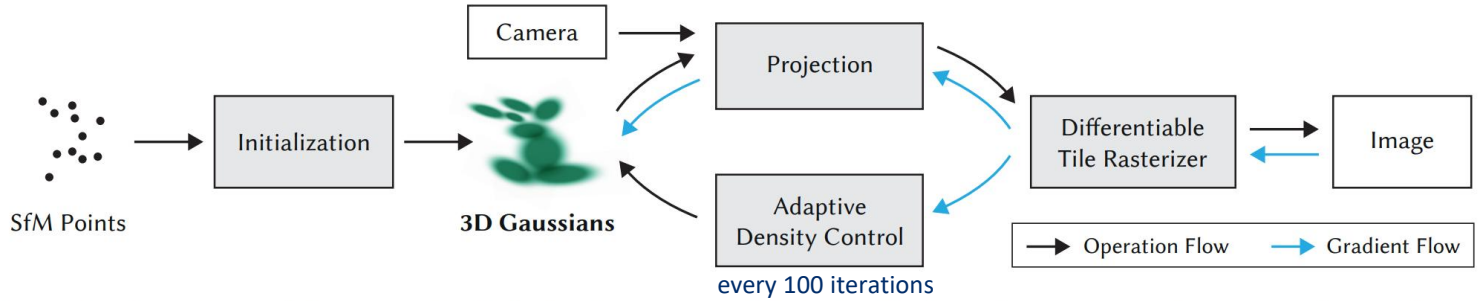


Fig. 2. Optimization starts with the sparse SfM point cloud and creates a set of 3D Gaussians. We then optimize and adaptively control the density of this set of Gaussians. During optimization we use our fast tile-based renderer, allowing competitive training times compared to SOTA fast radiance field methods. Once trained, our renderer allows real-time navigation for a wide variety of scenes.

PipeLine

Preliminaries

Local minima

local minima **arise when** Gaussian primitives **initialized** at random locations have to **move through** space to arrive at their **final location**.

Two issues prevent this:

First:

- Gaussian primitives have "**local support"** that **gradients vanish** if the distance to the "correct" location **exceeds** more than a few standard **deviations**.

Second:

- Even if a Gaussian is **close enough** to a "correct" location, there still **needs a "path"** to its final location along which loss decreases monotonically.

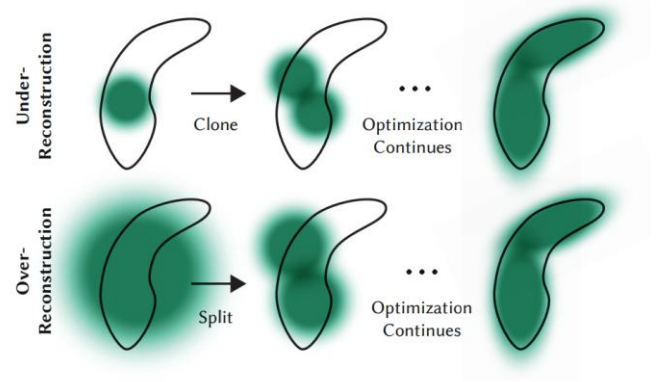


Fig. 4. Our adaptive Gaussian densification scheme. *Top row (under-reconstruction)*: When small-scale geometry (black outline) is insufficiently covered, we clone the respective Gaussian. *Bottom row (over-reconstruction)*: If small-scale geometry is represented by one large splat, we split it in two.

→ 3D-GS relies on **non-differentiable** pruning and splitting operations dubbed "**Adaptive Density Control**"

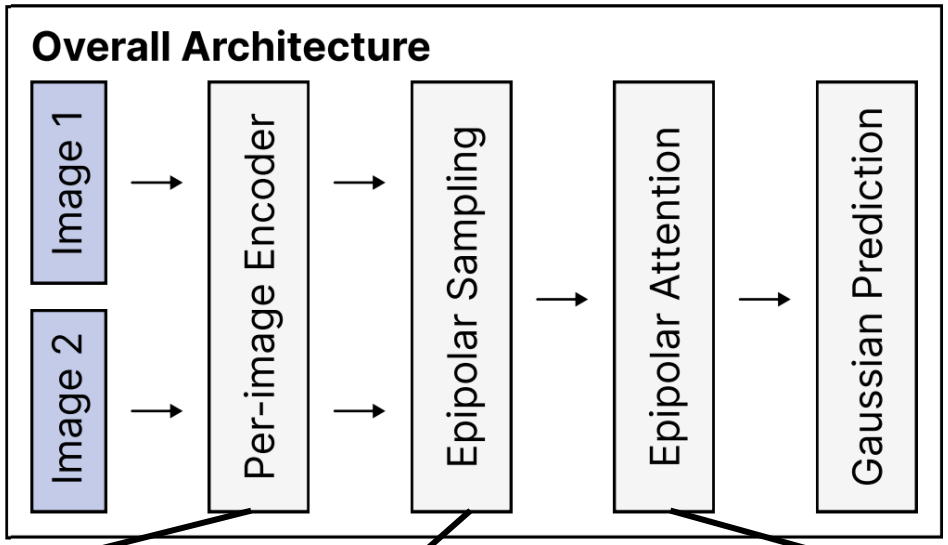
Imcompatible with the generalizable setting

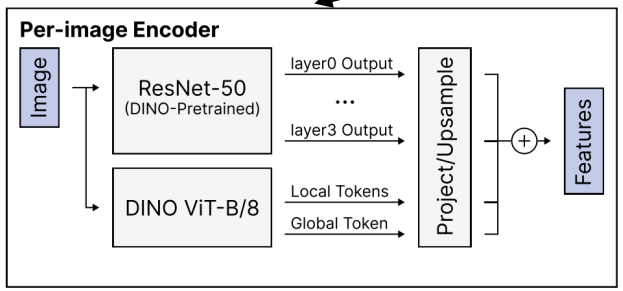
Method – Network Architectures

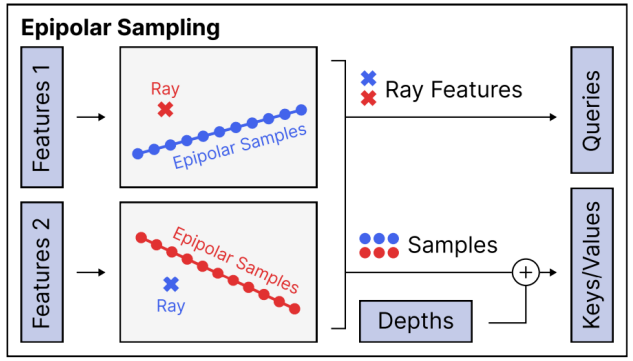
Network Architectures

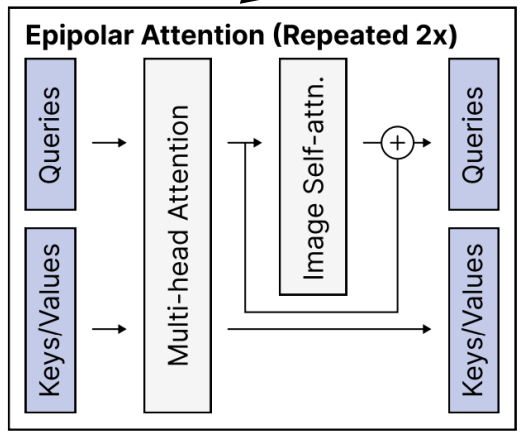
Four main parts:

- Per-image Encoder
- Epipolar Sampling
- Epipolar Attention
- Gaussian Prediction









Method

Resolving Scale Ambiguity: (실제 크기와 SfM으로 나온 scale이 안맞음)

- In an **ideal world**, datasets contain camera poses that are **metric.**

$$\mathcal{C}_i^{\mathrm{m}} = \{(\mathbf{I}_j, \mathbf{T}_j^{\mathrm{m}})\}_j$$

C : Scene, I: image, T: real-world-scale poses

- In **practice**, datasets provides poses from **SfM. (up to scale)**
- Different scenes → Different scale factor
- $\mathcal{C}_i = \{(\mathbf{I}_j, s_i \mathbf{T}_j^{\mathrm{m}})\}_j$ **s: scale factor**

+ **recovering** s_i from **single image** is **impossible** due to **scale ambiguity**.

→ 최소 2장의 image 필요 → propose a **two-view encoder**

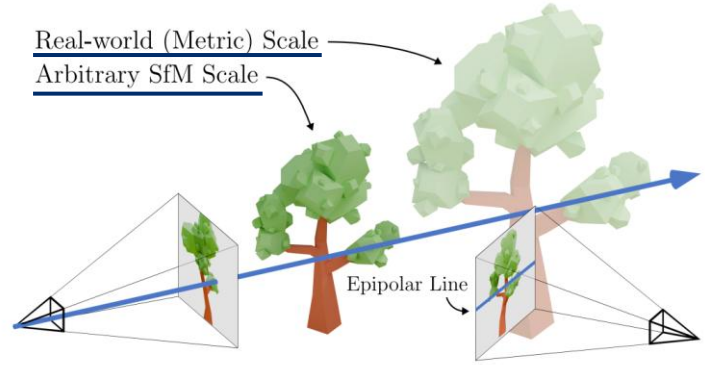
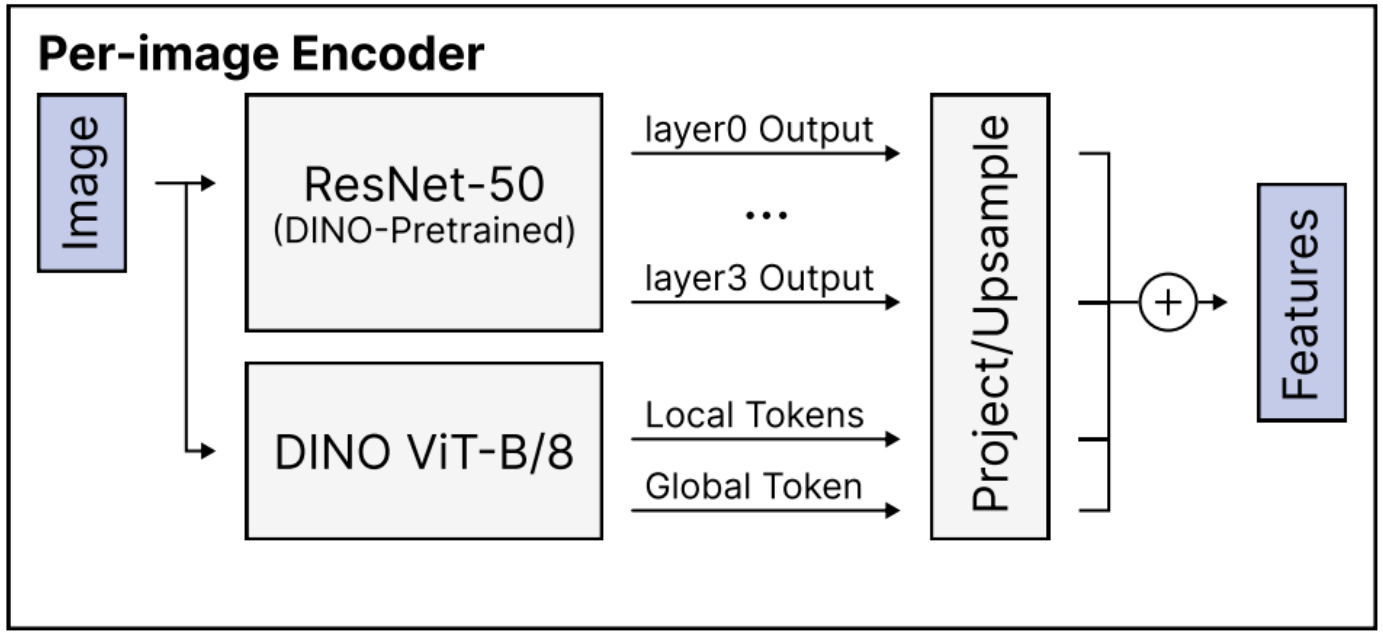


Figure 2. **Scale ambiguity.** SfM does not reconstruct camera poses in real-world, metric scale—poses are scaled by an arbitrary scale factor that is different for each scene. To render correct views, our model's 3D reconstruction needs to be consistent with this arbitrary scale. We illustrate how our epipolar encoder solves this problem. Features belonging to the ray's corresponding pixel on the left are compared with features sampled along the epipolar line on the right. Epipolar samples are augmented with their positionally-encoded depths along the ray, which allows our encoder to record correct depths. Recorded depths are later used for depth prediction.

Method

Resolving Scale Ambiguity

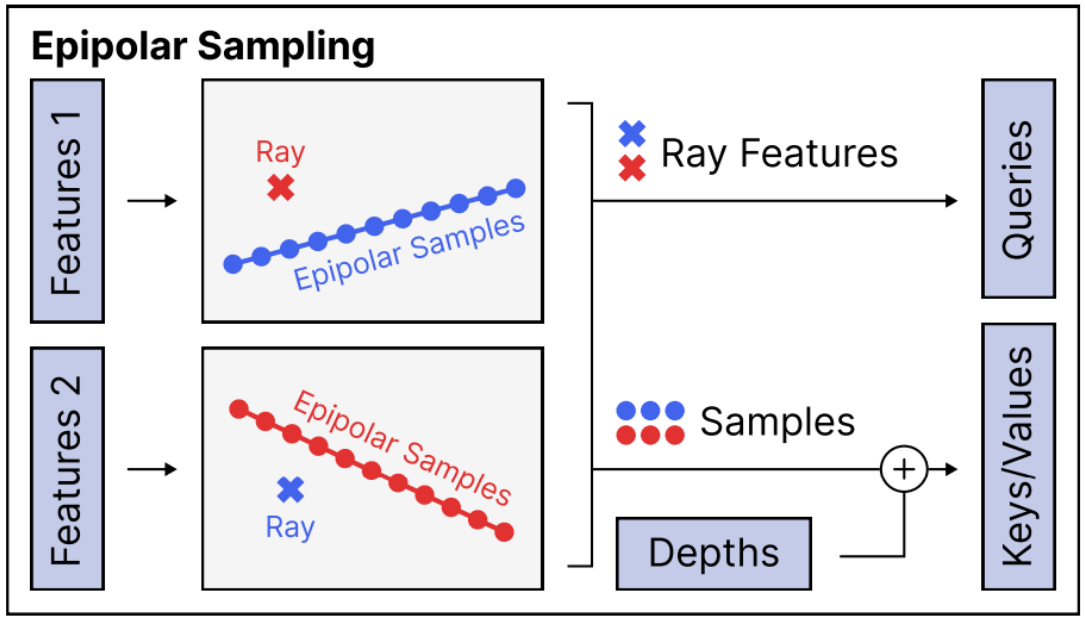
1. Encode each view into feature volumes $\mathbf{F}$ and $\tilde{\mathbf{F}}$ via **per-image feature encode**



Method

Resolving Scale Ambiguity

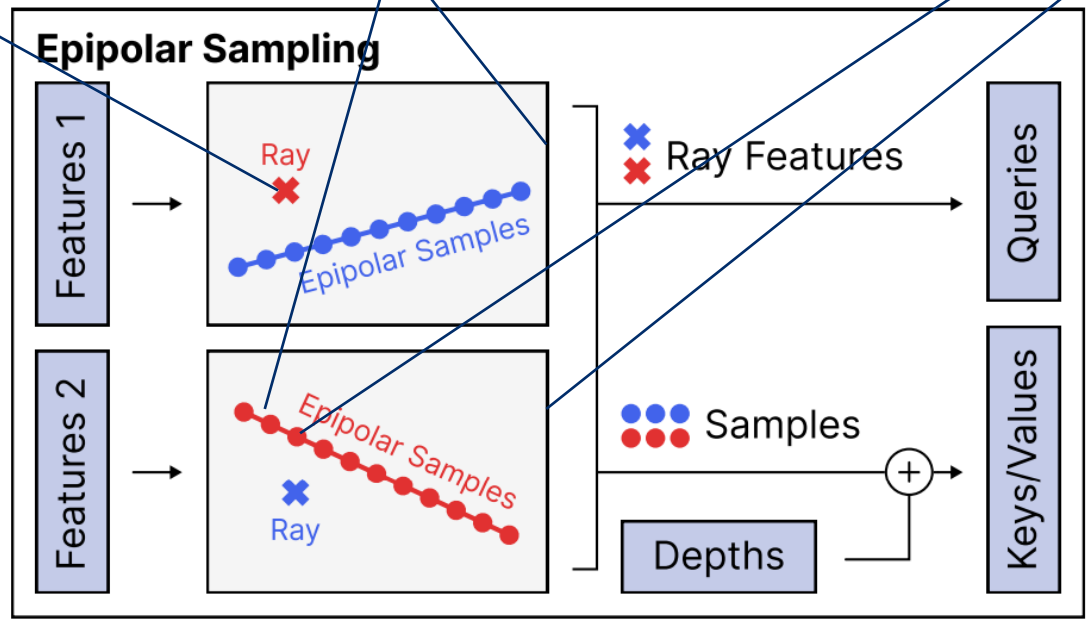
2. Let **u** be **pixel coordinates** from image **I**, and **ℓ** be the **epipolar line** induced by its ray in **˜I**.
3. Along **ℓ**, sample pixel coordinates.
4. compute its **distance** to I's camera origin $\tilde{d}_{\tilde{\mathbf{u}}_l}$ by **triangulation** of u and $\tilde{\mathbf{u}}_l$. (각 샘플까지의 거리 구하기)



Method

Resolving Scale Ambiguity

2. Let **u** be **pixel coordinates** from image **I**, and **ℓ** be the **epipolar line** induced by its ray in **˜I**.
3. Along **ℓ**, sample pixel coordinates.
4. compute its **distance** to I's camera origin $\tilde{d}_{\tilde{\mathbf{u}}_l}$ by **triangulation** of u and $\tilde{\mathbf{u}}_l$.



Method

Resolving Scale Ambiguity

2. Let **u** be **pixel coordinates** from image **I**, and **ℓ** be the **epipolar line** induced by its ray in **˜I**.
3. Along **ℓ**, sample pixel coordinates.
4. compute its **distance** to I's camera origin $\tilde{d}_{\tilde{\mathbf{u}}_l}$ by **triangulation** of u and $\tilde{\mathbf{u}}_l$.
5. Then, **cross-attention** and update per-pixel feature F[u] (어떤 sample이 진짜 정답인지)

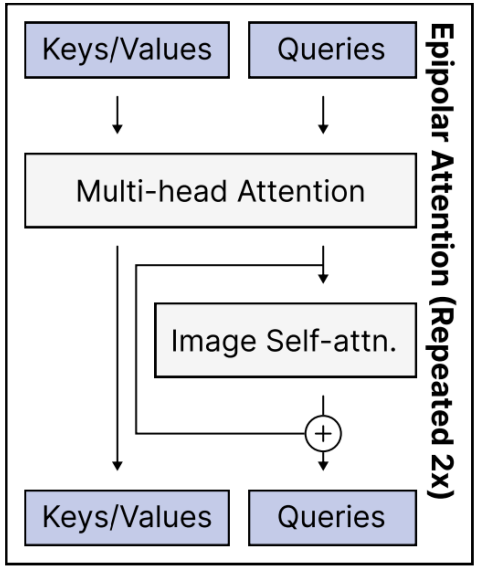
~I 이미지에 위치한 l번째 sample의 픽셀 좌표

l번째 sample의 카메라까지 거리

Source token

$$\mathbf{s} = \tilde{\mathbf{F}}[\tilde{\mathbf{u}}_l] \oplus \gamma(\tilde{d}_{\tilde{\mathbf{u}}_l}) \qquad (1)$$

$$\mathbf{q} = \mathbf{Q} \cdot \mathbf{F}[\mathbf{u}], \quad \mathbf{k}_l = \mathbf{K} \cdot \mathbf{s}, \quad \mathbf{v}_l = \mathbf{V} \cdot \mathbf{s}, \qquad (2)$$



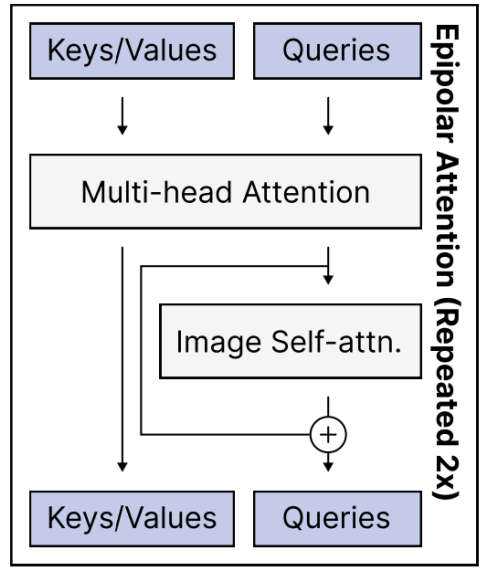
Method

Resolving Scale Ambiguity

6. After this, each pixel feature F[u] encodes the **scaled depth** that is **consistent** with the arbitrary **scale factor** s_i of the camera poses.

$$\mathbf{F}[\mathbf{u}] \mathrel{+}= \mathrm{Att}(\mathbf{q}, \{\mathbf{k}_l\}, \{\mathbf{v}_l\}), \qquad (3)$$

Skip connection



7. This enables our encoder to **propagate scaled depth estimates to parts** of the image feature maps that may **not have any epipolar correspondences** in the opposite image.

$$\mathbf{F} \mathrel{+}= \mathrm{SelfAttention}(\mathbf{F}). \qquad (4)$$

(Scale Ambiguity 해결한 Feature 생성)

- Mechanism can be **extended** to **more than two** input views

Method

Gaussian Parameter Prediction

Predict the parameters of a set of Gaussian primitives

- $\boldsymbol{\mu}_k$: mean
- $\boldsymbol{\Sigma}_k$: covariance
- $\boldsymbol{\alpha}_k$: opacity
- $\mathbf{S}_k$: coefficient

$$\{\mathbf{g}_k = (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \boldsymbol{\alpha}_k, \mathbf{S}_k)\}_k^K$$

Pixel-aligned Gaussian Prediction Module:

- For each pixel **at coordinate u**, take the corresponding feature F[u] as input and predict **Gaussian primitives** (각 Pixel 마다 하나의 Gaussian primitive)
- Predict **Gaussians** from **both reference views** and **simply union** them.

Method

Gaussian Parameter Prediction

Rather than predicting the **depth** d of Gaussian,
predict the **probability distribution** of the **likehood** that a **Gaussian exists** at a depth d along the ray u
(depth 값인 숫자 하나로 예측하기 보단 확률 분포가 가우시안과 overlap되기 쉬우므로)

Discrete probability density

- Set **near** and **far** planes d_{near} and d_{far}
- Discretize depth into **Z bins**

- Z-th element b_z(Bucket Depth): 카메라 원점으로부터 z번째 버킷(깊이 구간)의 중심까지의 거리

$$\mathbf{b}_z = \left(\left(1 - \frac{z}{Z}\right) \left(\frac{1}{d_{\text{near}}} - \frac{1}{d_{\text{far}}} \right) + \frac{1}{d_{\text{far}}} \right)^{-1}$$

전체 구간 중 몇 번째인지

전체 구간: 가장 가까운 곳 의 역수- 가장 먼 곳의 역수

시작 기준점(가장 먼 점의 역수 = 가장 작은 값)

Method

Gaussian Parameter Prediction

- define a discrete probability distribution $p_{\phi}(z)$
- ϕ : vector of discrete probabilities (이산확률벡터)
- ϕ_z : probability that a surface exists in depth bucket b_z (표면이 b_z에 존재할 확률)
- δ : a per-bucket center offset (각 구간에서의 offset)
- **o** : 카메라 렌즈의 위치(원점)
- $\mathbf{d}_u$: ray direction(vector)

$$\phi, \delta, \Sigma, \mathbf{S} = f(\mathbf{F}[\mathbf{u}]).$$

FCNN

$$\mu = \mathbf{o} + (\mathbf{b}_z + \delta_z)\,\mathbf{d}_\mathbf{u},\ z \sim p_{\phi}(z),\ (\phi, \delta) = f(\mathbf{F}[\mathbf{u}]) \quad (7)$$

네트워크가 예측한 확률 분포에 따라, 실제 가우시안을 심을 위치(인덱스 z)를 하나 '샘플링'

Method

Gaussian Parameter Prediction

- Sampling operation $z \sim p_{\phi}(z)$ is **not differentiable.**
- Need a **reparameterization trick**

Reparameterization trick

set the **opacity α** of a Gaussian **to be equal** to the **probability of the bucket** that it was sampled from.

Consider that we **sampled** $z \sim p_{\phi}(z)$.
Then set $\alpha = \phi_z$. **: α to the z-th entry of the vector of probabilities ϕ.** **(불투명도를 그 위치가 뽑힐 확률값과 똑같이 설정)**

→ $\nabla_{\phi}\mathcal{L} = \nabla_{\alpha}\mathcal{L}$

At an intuitive level, the case where **sampling** produces a **correct depth**.
In this case, **gradient descent increases** the **opacity** of the Gaussian, leading it to be **sampled more often**.

Method

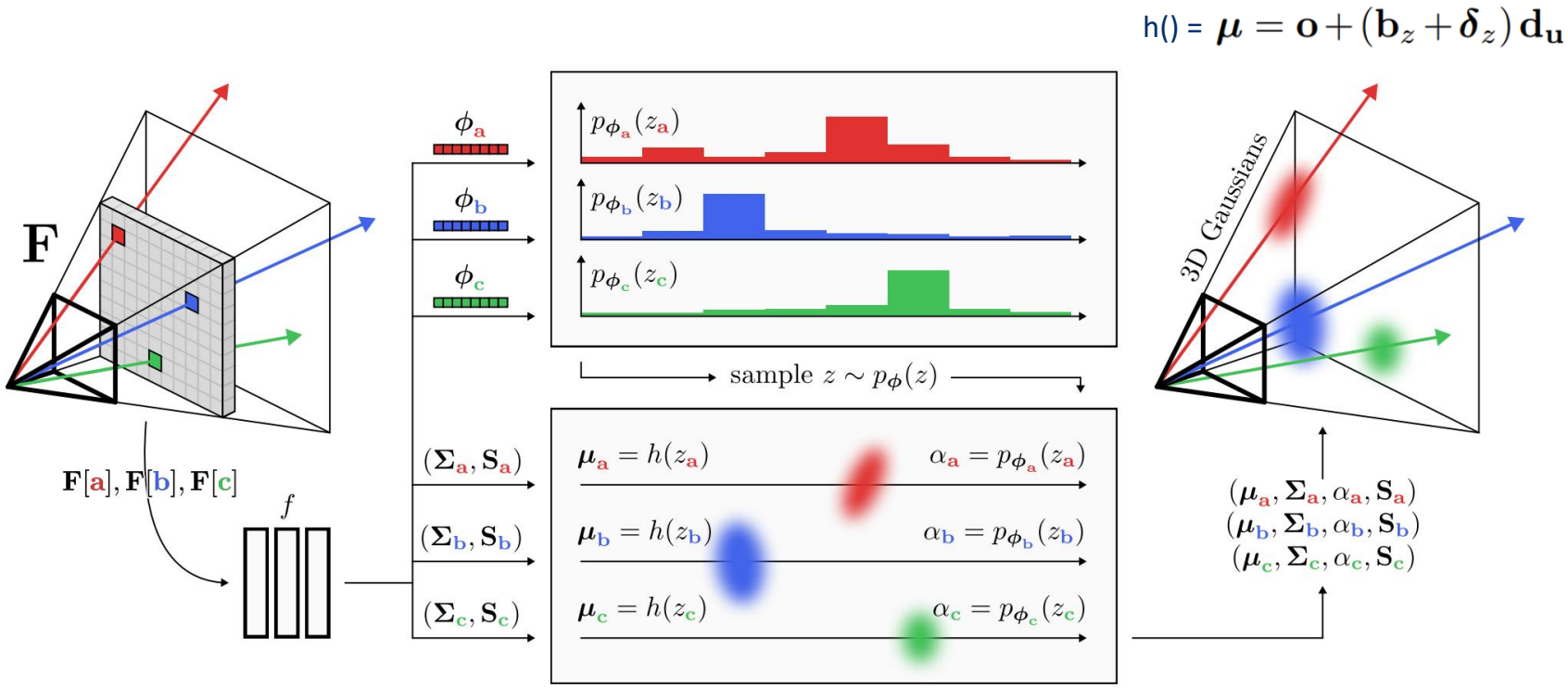


Figure 3. **Proposed probabilistic prediction of pixel-aligned Gaussians.** For every pixel feature $\mathbf{F}[\mathbf{u}]$ in the input feature map, a neural network f predicts Gaussian primitive parameters $\boldsymbol{\Sigma}$ and $\mathbf{S}$. Gaussian locations $\boldsymbol{\mu}$ and opacities $\boldsymbol{\alpha}$ are not predicted directly, which would lead to local minima. Instead, f predicts per-pixel discrete probability distributions over depths $p_{\boldsymbol{\phi}}(z)$, parameterized by $\boldsymbol{\phi}$. Sampling then yields the locations of Gaussian primitives. The opacity of each Gaussian is set to the probability of the sampled depth bucket. The final set of Gaussian primitives can then be rendered from novel views using the splatting algorithm proposed by Kerbl et al. [19]. Note that for brevity, we use h to represent the function that computes depths from bucket indices (see equations 6 and 7).

Method

Pseudo code

Algorithm 1 Probabilistic Prediction of a Pixel-Aligned Gaussian.

Require: Depth buckets $\mathbf{b} \in \mathbb{R}^Z$, feature $\mathbf{F}[\mathbf{u}]$ at pixel coordinate $\mathbf{u}$, camera origin of reference view $\mathbf{o}$, ray direction $\mathbf{d_u}$.

1: $(\boldsymbol{\phi}, \boldsymbol{\delta}, \boldsymbol{\Sigma}, \mathbf{S}) = f(\mathbf{F}[\mathbf{u}])$ ▷ predict depth probabilities $\boldsymbol{\phi}$ and offsets $\boldsymbol{\delta}$, covariance $\boldsymbol{\Sigma}$, spherical harmonics coefficients $\mathbf{S}$

2: $z \sim p_{\boldsymbol{\phi}}(z)$ ▷ Sample depth bucket index z from discrete probability distribution parameterized by $\boldsymbol{\phi}$

3: $\boldsymbol{\mu} = \mathbf{o} + (\mathbf{b}_z + \boldsymbol{\delta}_z)\mathbf{d_u}$ ▷ Compute Gaussian mean $\boldsymbol{\mu}$ by unprojecting with depth $\mathbf{b}_z$ adjusted by bucket offset $\boldsymbol{\delta}_z$

4: $\boldsymbol{\alpha} = \boldsymbol{\phi}_z$ ▷ Set Gaussian opacity $\boldsymbol{\alpha}$ according to probability of sampled depth (Sec. 4.2).

5: **return** $(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\alpha}, \mathbf{S})$

Experiment

Evaluate the method on **wide-baseline novel view synthesis** from image pairs

Datasets

RealEstate10k : a dataset of **home walkthrough videos** downloaded from YouTube
ACID : a dataset of **aerial landscape videos** downloaded from YouTube

RealEstate10k

ACID

Experiment

Quantitative Comparison

Input을 넣고, 3d Gaussian primitives Output이 나오는 시간

	ACID			RealEstate10k			Inference Time (s)		Memory (GB)	
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	Encode ↓	Render ↓	Training ↓	Inference ↓
Ours	**28.27**	**0.843**	**0.146**	**26.09**	**0.863**	**0.136**	0.102	**0.002**	**14.4**	**3.002**
Du et al. [10]	26.88	0.799	0.218	24.78	0.820	0.213	0.016	1.309	314.3	19.604
GPNR [46]	25.28	0.764	0.332	24.11	0.793	0.255	**N/A**	13.340	3789.9	19.441
pixelNeRF [58]	20.97	0.547	0.533	20.43	0.589	0.550	0.005	5.294	436.7	3.962

Table 1. **Quantitative comparisons.** We outperform all baseline methods in terms PSNR, LPIPS, and SSIM for novel view synthesis on the real-world RealEstate10k and ACID datasets. In addition, our method requires less memory during both inference and training and renders images about 650 times faster than the next-fastest baseline. In the memory column, we report memory usage for a single scene and 256×256 rays, extrapolating from the smaller number of rays per batch used to train the baselines where necessary. Note that we report GPNR's encoding time as N/A because it has no encoder. We **bold first-place** results and underline second-place results in each column.

Experiment

Qualitative Comparison

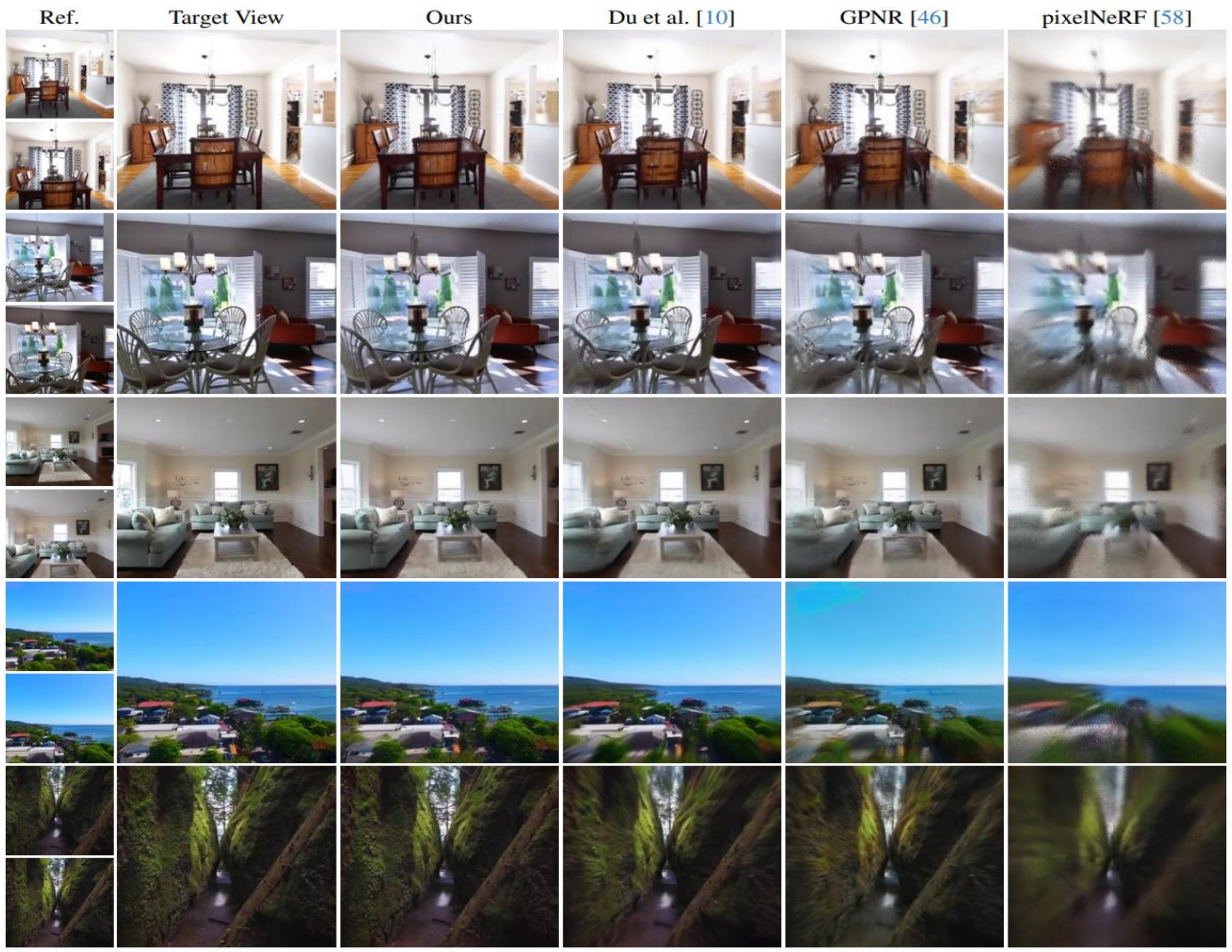


Figure 5. **Qualitative comparison of novel views on the RealEstate10k (top) and ACID (bottom) test sets.** Compared to the baselines, our approach not only produces more accurate and perceptually appealing images, but also generalizes better to out-of-distribution examples like the creek in the bottom row.

Experiment

Attention visualization

Figure 6. **Attention visualization.** We visualize the epipolar cross-attention weights between the rays on the left and the corresponding epipolar lines on the right to confirm that our model learns to find the correct correspondence along each ray.

Experiment

Ablation Study

Method	PSNR ↑	SSIM ↑	LPIPS ↓
Ours	**26.09**	**0.863**	**0.136**
No Epipolar Encoder	19.89	0.639	0.295
No Depth Encoding	24.97	0.830	0.158
No Probabilistic Sampling	24.62	0.828	0.171
Plus Depth Regularization	25.94	0.862	0.137

Table 2. **Ablations.** Both our epipolar encoder and our probabilistic sampling scheme are essential for high-quality novel view synthesis. Depth regularization slightly impacts rendering quality.

Experiment

Ablation Study

Figure 7. **Ablations.** Without the epipolar transformer, our model is unable to resolve scale ambiguity, leading to ghosting artifacts. Without our sampling approach, our model falls into local minima that manifest themselves as speckling artifacts. Regularizing our model's predicted depths minimally affects rendering quality.

Experiment

Input images: 2 vs 3

Method	PSNR ↑	SSIM ↑	LPIPS ↓
Ours	26.09	0.863	0.136
Ours (3 Views)	28.31	0.908	0.100

Table 3. **Quantitative comparison of 2 and 3 view Real Estate 10k results.** Given a third reference view located halfway between the two existing reference views, a 3-view variant of pixelSplat produces slightly better results.

Conclusion

Contribution

- Proposes a generalizable model that infers 3D Gaussian primitives directly from image pairs in a single forward pass.
- By leveraging epipolar constraints, it solves scale ambiguity.
- Predicts depth as a pixel-wise probability distribution instead of a point estimate to avoid local minima.

Conclusion

Limitation

- Simply unions of Gaussians predicted from each view rather than fusing → Redundancy.
- Does not address generative modeling to fill in unseen or occluded parts of the scene.
- The epipolar attention mechanism becomes prohibitively expensive in terms of memory when extended to many reference views.