Survey for 3DGS-SLAM

Guan Weipeng
Email: wpguan@connect.hku.hk

The University of Hong Kong

Department of Mechanical Engineering

Adaptive Robotic Controls Lab (ArcLab)


香 港 大 學
THE UNIVERSITY OF HONG KONG
目录
CONTENT

What is 3DGS?

- Gaussian Splatting is an effective method for representing 3D scenes with novel-view synthesis capability. This approach is notable for its speed, without compromising on the rendering quality. Originally, 3D Gaussians are initialized from a sparse SfM point cloud of a scene. Having a set of images observing the scene from different angles, the Gaussian parameters are optimized using differentiable rendering while 3D Gaussians are adaptively added or removed to the representation based on a set of heuristics.
- 高斯抛雪球是一种用于表示具有新视角3D场景合成的方法。该方法以其速度而著称，同时不损害渲染质量。3D高斯是从场景的稀疏SfM点云中初始化的。在一组从不同角度观察场景的图像的情况下，使用可微分渲染(前向渲染，光栅化)来优化高斯参数，同时根据一组启发式规则自适应地添加或删除3D高斯以进行更新。

$$f^{3\mathrm{D}}(p) = \mathrm{sigmoid}(o)\exp\left(-\frac{1}{2}(p-\mu)^T\Sigma^{-1}(p-\mu)\right)$$

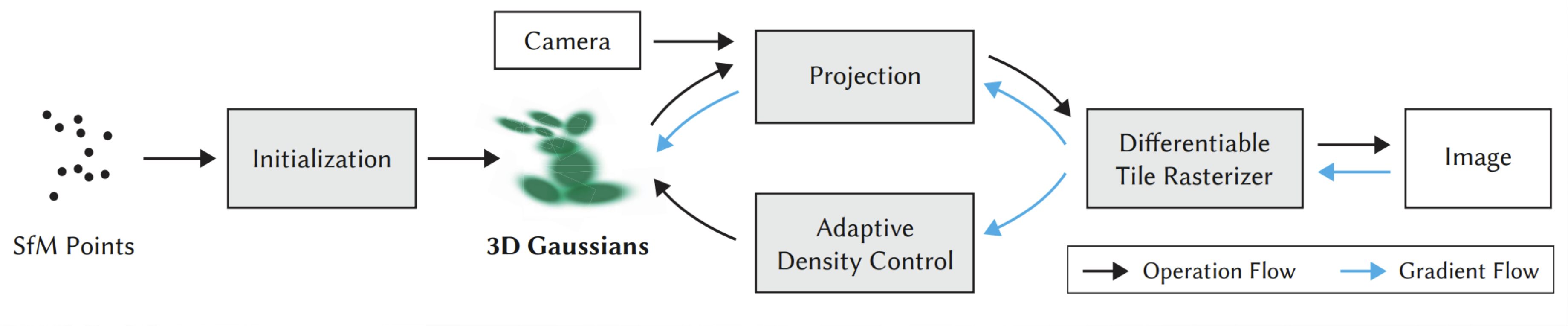
$$C_{\mathrm{pix}} = \sum_{i\in V} c_i f^{2\mathrm{D}}_{i,\mathrm{pix}} \prod_{j=1}^{i-1}\left(1 - f^{2\mathrm{D}}_{j,\mathrm{pix}}\right)$$

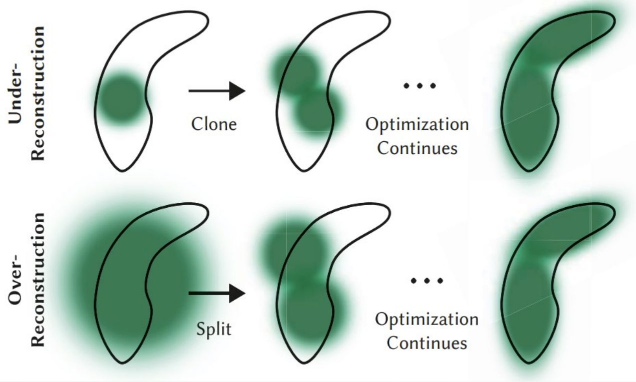
What is 3DGS?

- Representing the scene with 3D Gaussians using the sparse point cloud from SfM;
- Performing interleaved optimization/density control of the 3D Gaussians, optimizing anisotropic covariance;
- Developing a fast visibility-aware rendering algorithm (tile-based splatting);

Limitations:

- Artifacts in the regions where the scene is **not well observed**;
- **Memory consumption** is significantly higher than NeRF-based solutions;
- Having **popping artifacts** when our optimization creates large Gaussians;
- Elongated artifacts or "splotchy" Gaussians



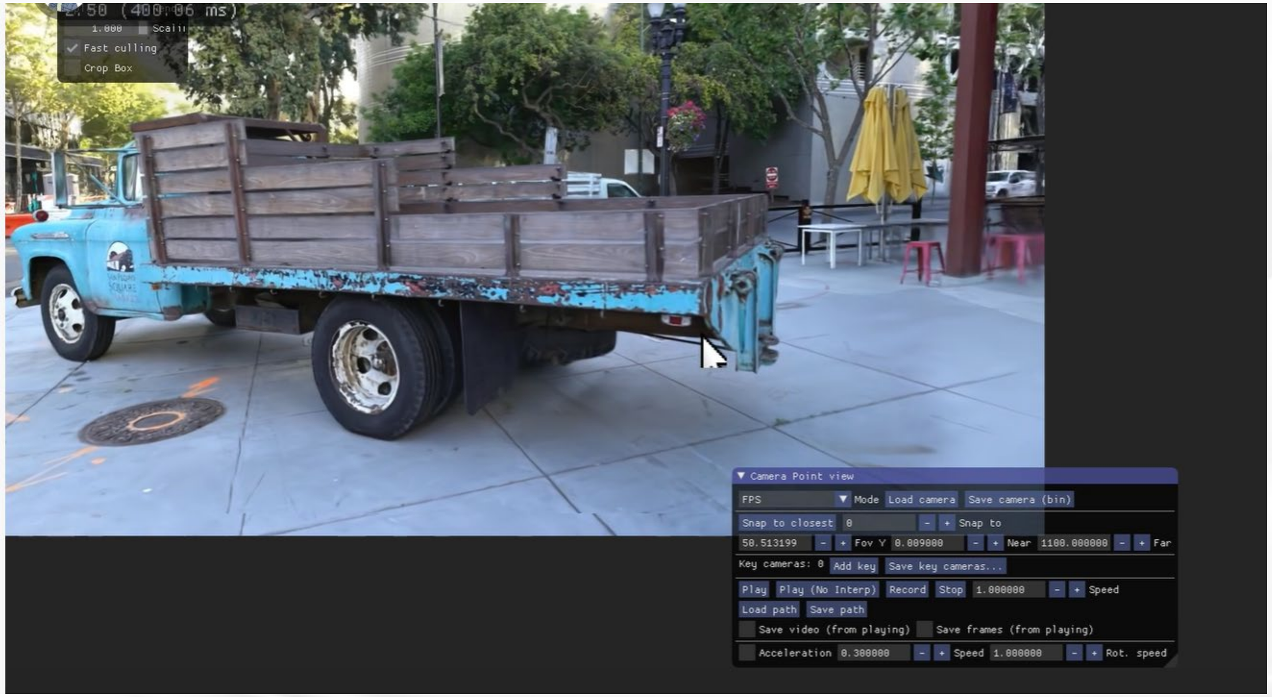


Implementation 3DGS in C++

香 港 大 學
THE UNIVERSITY OF HONG KONG

- Fully in C++ and CUDA 11.7;
- LibTorch Framework;
- Python with development headers;
- Cmake 3.22 or higher;
- SIBR_viewers for visualization;

(6000 Iters)	GPU3090 (cpp)	GPU3060 (cpp)	GPU3090 (py)
Train	265s / psnr:21.28	277s / psnr:21.30	452s / psnr: 20.13
Truck	189s / psnr:23.79	343s / psnr:23.86	535s / psnr: 23.89



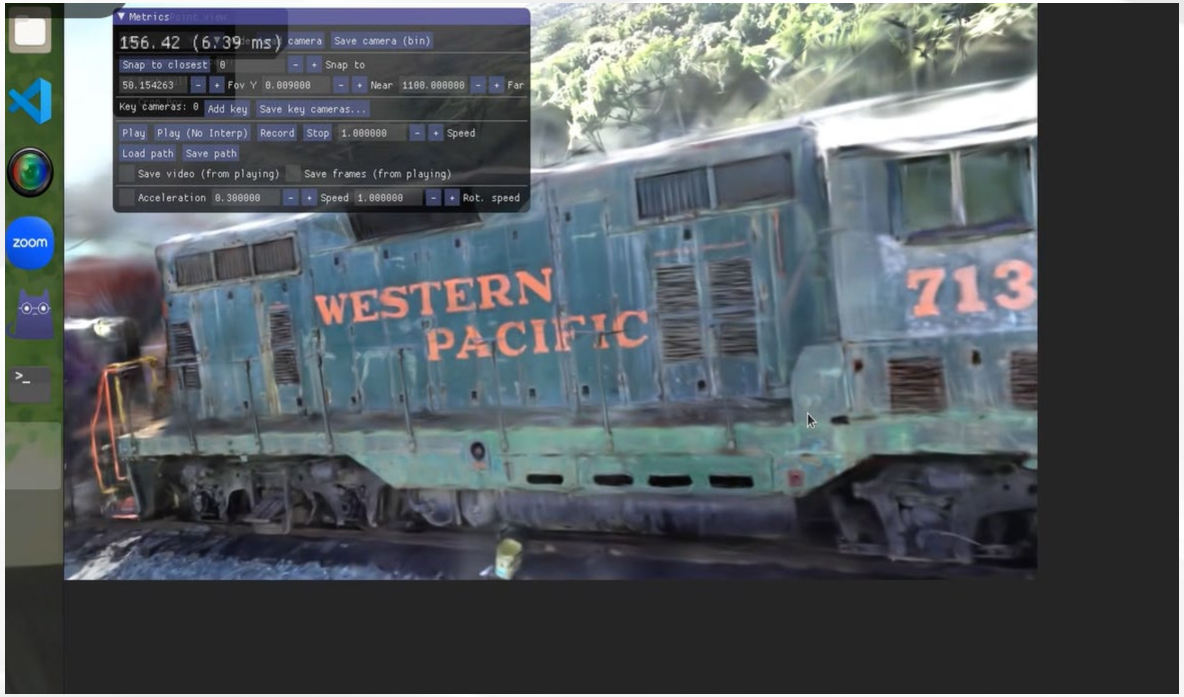
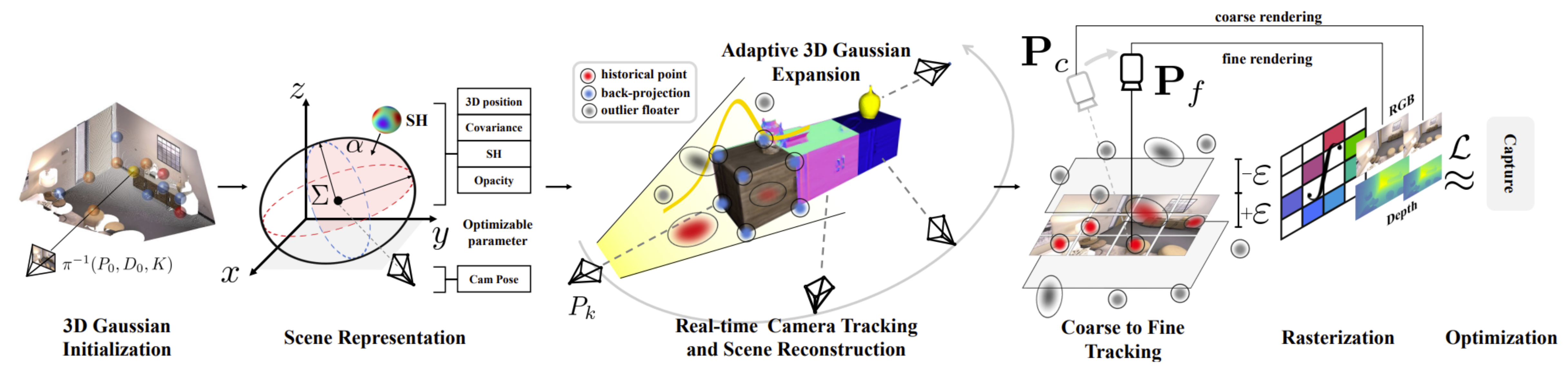


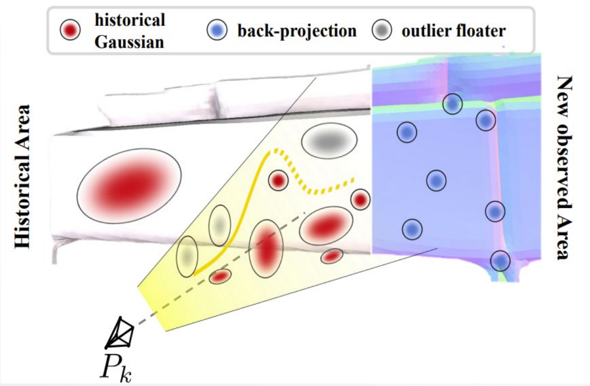
Image-based 3DGS

GS-SLAM: Dense Visual SLAM with 3D Gaussian Splatting

香 港 大 學
THE UNIVERSITY OF HONG KONG

- The first one utilizes 3DGS for SLAM for pose tracking (coarse-to-fine) and RGB-D rendering;
- Adaptive expansion strategy that adds / deletes 3D Gaussian;





GS-SLAM: Dense Visual SLAM with 3D Gaussian Splatting

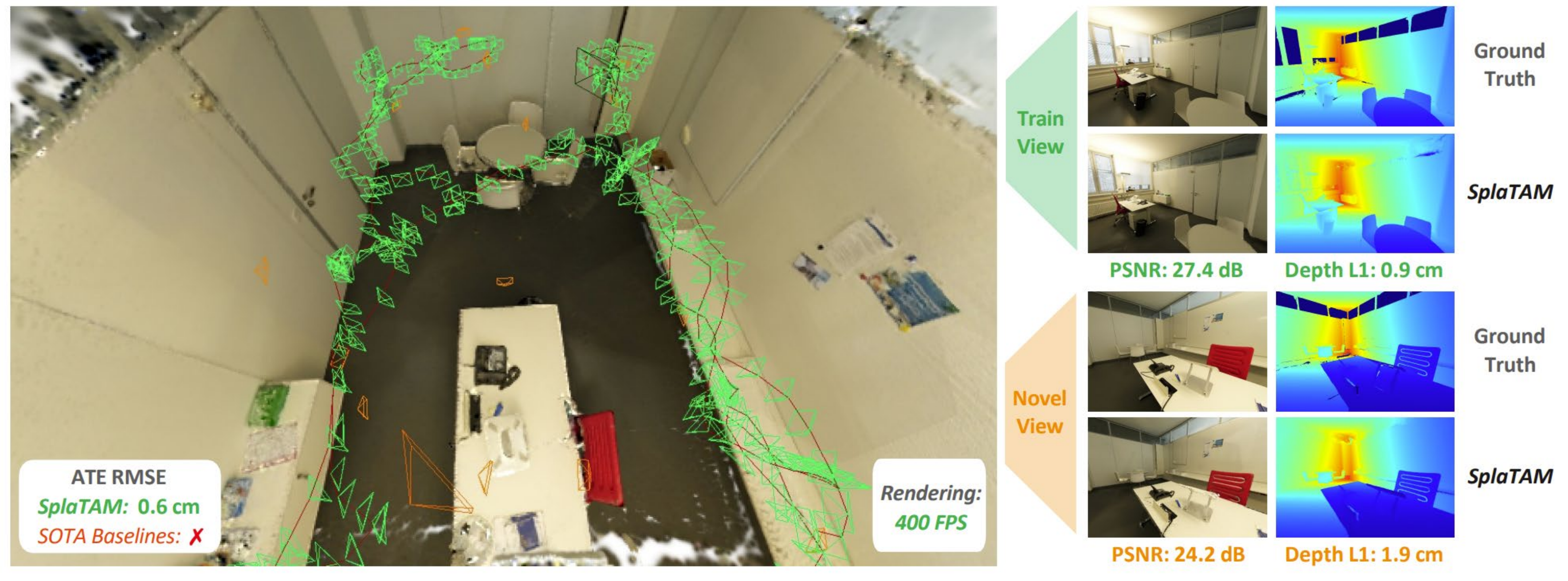
- The rendering speed has reached 386 FPS, but the overall performance of the SLAM framework, with a running speed of 8.34 FPS, and the localization accuracy, hasn't seen a significant improvement;
- Memory consumption is too large!

Table 5. **Runtime and Memory Usage on Replica** `Room 0`. The decoder parameters and embedding denote the parameter number of MLPs and the memory usage of the scene representation.

Method	Tracking [ms×it] ↓	Mapping [ms×it] ↓	System FPS ↑	Decoder param ↓	Scene Embedding↓
Point-SLAM [24]	0.06 × 40	34.81 × 300	0.42	0.127 M	55.42 MB
NICE-SLAM [51]	6.64 × 10	28.63 × 60	2.91	0.06 M	48.48 MB
Vox-Fusion [45]	0.03 × 30	66.53 × 10	1.28	0.054 M	1.49 MB
CoSLAM [45]	6.01 × 10	13.18 × 10	16.64	1.671 M	—
ESLAM [45]	6.85 × 8	19.87 × 15	13.42	0.003 M	27.12 MB
GS-SLAM	11.9 × 10	12.8 × 100	8.34	**0 M**	198.04 MB

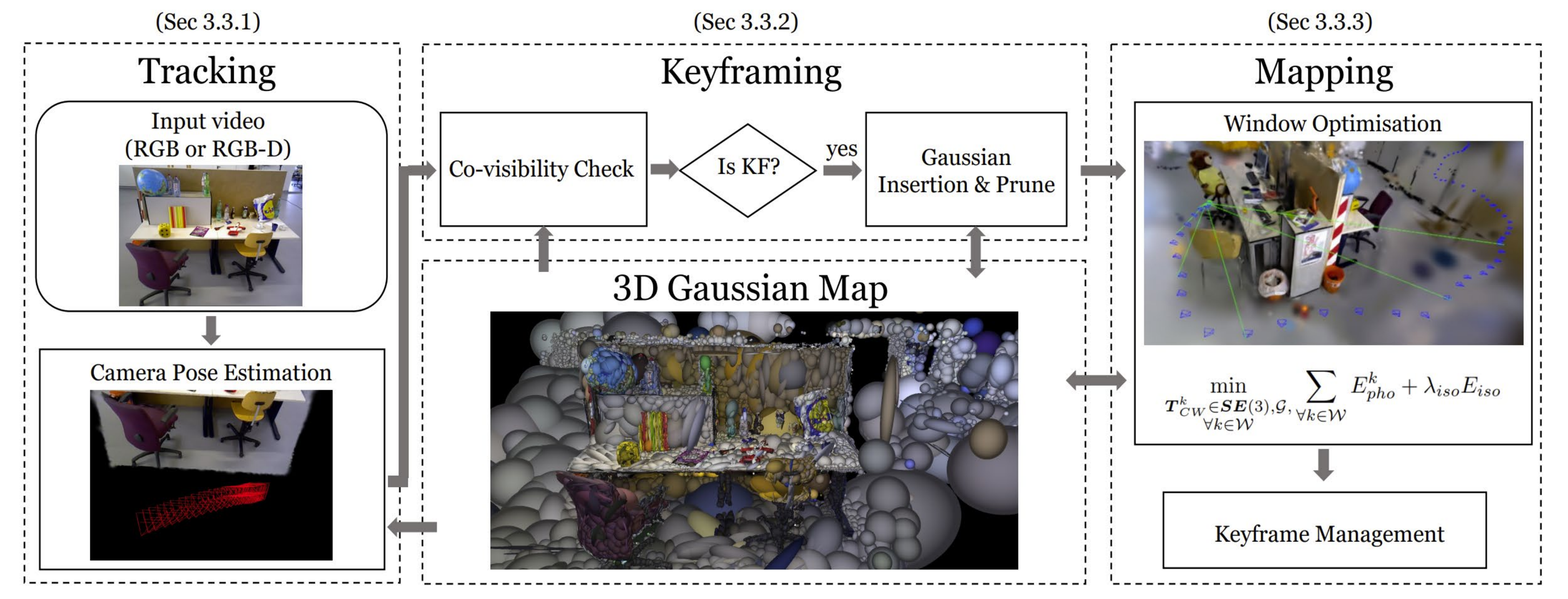
SplaTAM: Splat, Track & Map 3D Gaussians for Dense RGB-D SLAM

- Introducing several simple modifications that make splatting even faster; while the contribution 2~4 is the advantages of using 3DGS;
- Unobserved/novel camera viewpoint—— new evaluation metrics;



Gaussian Splatting SLAM

- The first application of 3D Gaussian Splatting to incremental 3D reconstruction using a single moving monocular or RGB-D camera; Can reconstruct the tiny and even **transparent objects**;
- **Formulating the analytic Jacobian of camera pose with respect to a 3D Gaussians map**, enable camera poses to be optimized alongside scene geometry;
- Introducing the novel Gaussian shape **regularization** to ensure geometric consistency;
- Propose a novel Gaussian resource **allocation and pruning** method to keep the geometry clean and enable accurate camera tracking



Gaussian Splatting SLAM

- In the monocular scenario, its performance is comparable to most of RGB-D methods, while in the RGB-D case, it is nearly on par with ORB-SLAM2.

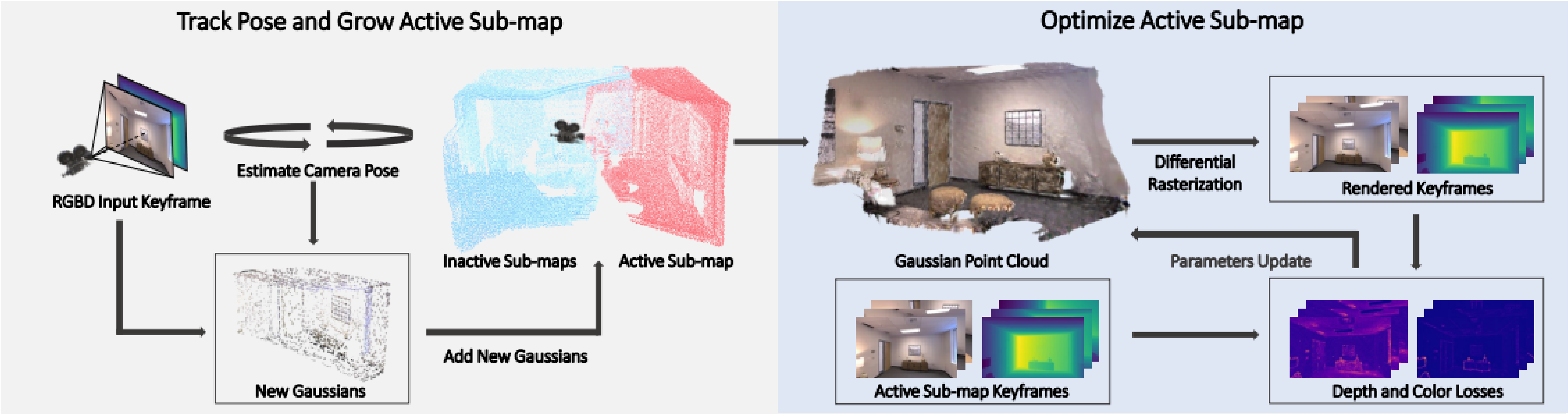
Memory Usage				
iMAP [33]	NICE-SLAM [46]	Co-SLAM [39]	Ours (Mono)	Ours (RGB-D)
0.2M	101.6M	1.6M	2.6MB	3.97MB

Input	Loop-closure	Method	fr1/desk	fr2/xyz	fr3/office	Avg.
Monocular	w/o	DSO [4]	22.4	**1.10**	9.50	11.0
		DROID-VO [36]	5.20	10.7	7.30	7.73
		DepthCov [3]	5.60	1.20	68.8	25.2
		Ours	**4.15**	4.79	**4.39**	**4.44**
	w/	DROID-SLAM [36]	**1.80**	**0.50**	2.80	1.70
		ORB-SLAM2 [20]	2.00	0.60	**2.30**	**1.60**
RGB-D	w/o	iMAP [33]	4.90	2.00	5.80	4.23
		NICE-SLAM [46]	4.26	6.19	6.87	5.77
		DI-Fusion [7]	4.40	2.00	5.80	4.07
		Vox-Fusion [43]	3.52	1.49	26.01	10.34
		ESLAM [8]	2.47	**1.11**	2.42	2.00
		Co-SLAM [39]	2.40	1.70	2.40	2.17
		Point-SLAM [27]	4.34	1.31	3.48	3.04
		Ours	**1.52**	1.58	**1.65**	**1.58**
	w/	BAD-SLAM [29]	1.70	1.10	1.70	1.50
		Kintinous [40]	3.70	2.90	3.00	3.20
		ORB-SLAM2 [20]	**1.60**	**0.40**	**1.00**	**1.00**

Gaussian-SLAM

香 港 大 學
THE UNIVERSITY OF HONG KONG

- Proposing a novel effective strategy for seeding new Gaussians for newly explored areas;
- Dividing the scene into many submaps, so that they can be independently optimized and do not need to be kept in memory;
- Deeply analysis the limitations of 3DGS (original version);



Gaussian-SLAM: Photo-realistic Dense SLAM with Gaussian Splatting

香 港 大 學
THE UNIVERSITY OF HONG KONG

- Novel strategies for **seeding** and optimizing 3DGS with proposed **online learning method**;
- Investigating **frame-to-model tracking** with Gaussian splatting via photometric error minimization
- An extension of Gaussian splatting to better encode geometry;

- Pinpoint the original offline 3DGS fails or prove ineffective:
  - ➢ Seeding strategy for online SLAM builds upon a sparse point cloud, which is uncertainty;
  - ➢ How many Gaussian should be considered for optimization (**too slow or Catastrophic forgetting**)?
  - ➢ The result of a splatting optimization highly **depend on the initialization** of Gaussians; Gaussians may grow suddenly in different directions **depending on the neighboring Gaussians;** the inherent symmetries of the 3D Gaussians allow parameter alterations without affecting the loss function, resulting in **non-unique solutions**;
  - ➢ While **good view coverage** in an offline setting constrains most Gaussians well, novel views in a sparse-view SLAM setting often contain artifacts resulting from previously under-constrained Gaussians.
  - ➢ Limited geometric accuracy;

Gaussian-SLAM: Photo-realistic Dense SLAM with Gaussian Splatting

- While the performance in terms of localization accuracy is mediocre in this paper, the mapping results are excellent. Moreover, it provides a thorough analysis of the issues present in 3DGS, offering valuable insights.

Photo-SLAM: Real-time Simultaneous Localization and Photorealistic Mapping for Monocular, Stereo, and RGB-D Cameras

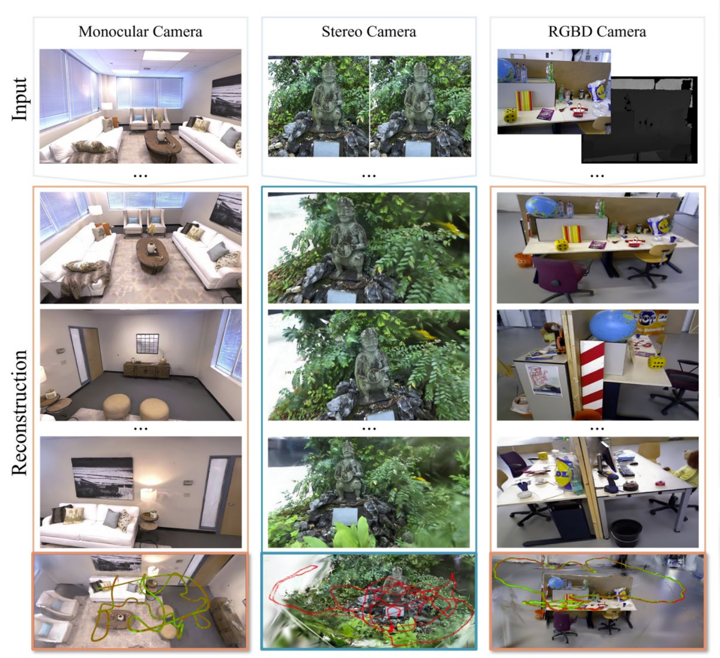
- A Hyper primitives map which is composed of point clouds storing ORB features, rotation, scaling, density, and spherical harmonic (SH) coefficients;
- The hyper primitives map allows the system to efficiently optimize tracking using a factor graph solver and learn the corresponding mapping by backpropagating the loss between the original images and rendering images. The images are rendered by 3D Gaussian splatting;
- A **Gaussian-Pyramid-based training method** to progressively learn multi-level features, enhancing photorealistic mapping performance;
- Simultaneously exploit **explicit geometric features for localization** and learn **implicit photometric features to represent the texture information** of the observed environment; The render speed is up to 1000 FPS;

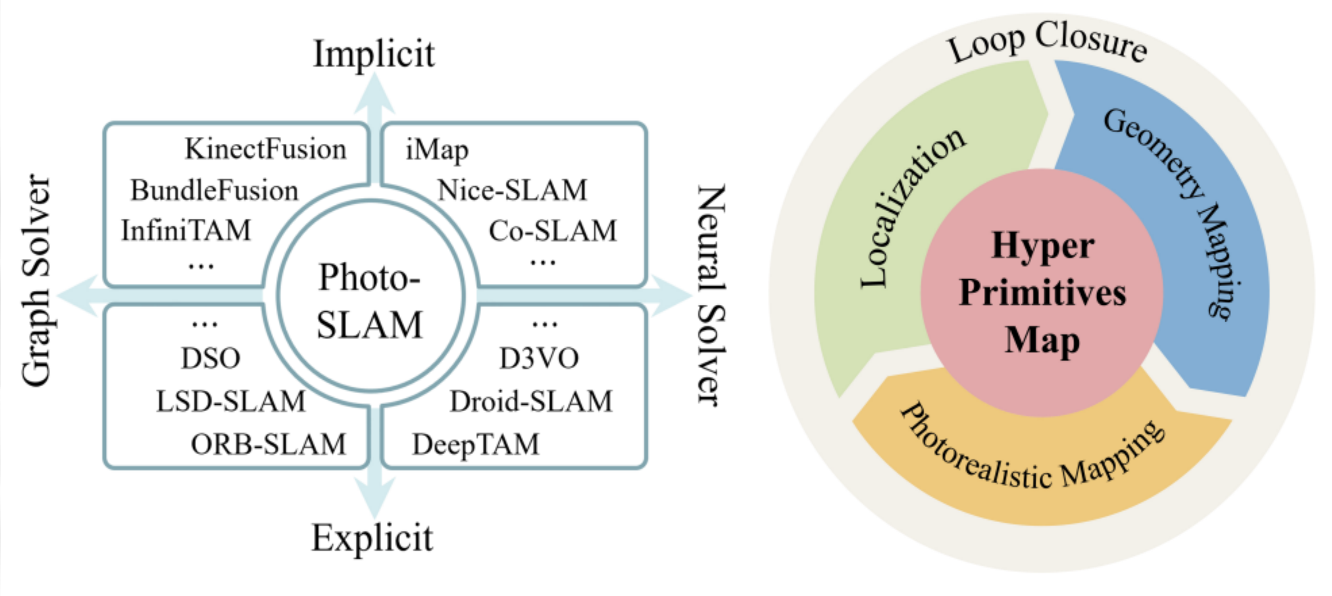
4.1. Implementation and Experiment Setup

We implemented Photo-SLAM fully in C++ and CUDA, making use of ORB-SLAM3 [2], 3D Gaussian splatting [18], and the LibTorch framework. The optimization

Photo-SLAM

- ORB-SLAM+3DGS (monocular, stereo, and RGB-D), Cpp-version, using libtorch;
- **Hyper primitives map** which is composed of point clouds storing ORB features, rotation, scaling, density, and spherical harmonic (SH) coefficients;
- Efficiently optimize tracking using a factor graph solver and learn the corresponding mapping by backpropagating the loss between the original images and rendering images;
- To achieve high-quality mapping without reliance on dense depth information, propose a **geometry-based densification strategy** and a **Gaussian-Pyramid-based (GP) learning** method;





(a) Taxonomy (b) Overview of Photo-SLAM

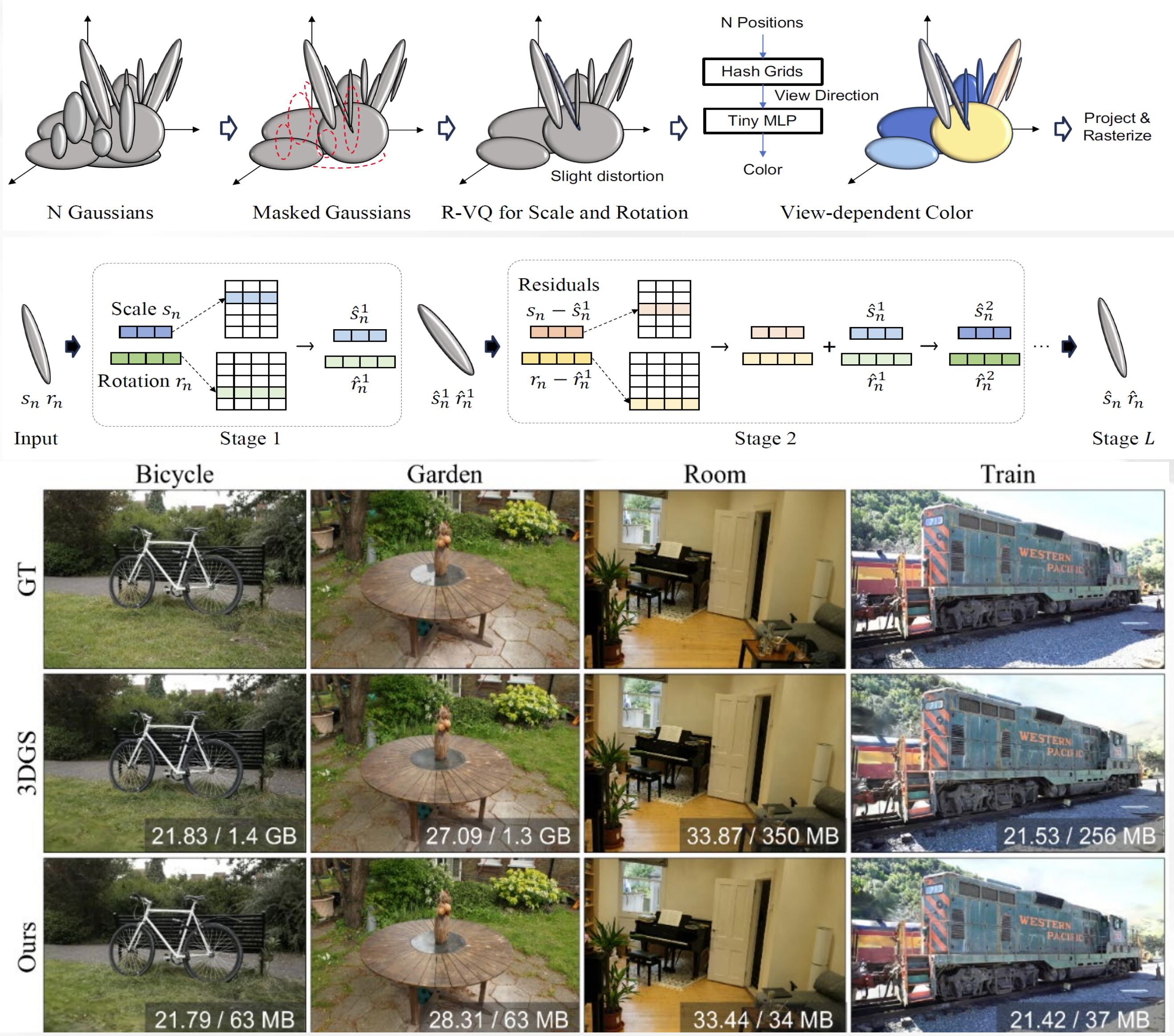
Photo-SLAM: Real-time Simultaneous Localization and Photorealistic Mapping for Monocular, Stereo, and RGB-D Cameras

On Replica Dataset		Localization (cm)		Mapping			Resources		
Cam	Method	RMSE ↓	STD ↓	PSNR ↑	SSIM ↑	LPIPS ↓	Tracking FPS ↑	Rendering FPS ↑	GPU Memory Usage ↓
Mono	ORB-SLAM3 [2]	3.942	3.115	-	-	-	**58.749**	-	0
	DROID-SLAM [33]	0.725	**0.308**	-	-	-	35.473	-	11 GB
	Nice-SLAM* [45]	99.9415	35.336	16.311	0.720	0.439	2.384	0.944	12 GB
	Orbeez-SLAM [4]	-	-	23.246	0.790	0.336	49.200	1.030	6 GB
	Go-SLAM [43]	71.054	24.593	21.172	0.703	0.421	25.366	0.821	22 GB
	Ours (Jetson)	1.235	0.756	29.284	0.883	0.139	18.315	95.057	**4 GB**
	Ours (Laptop)	**0.713**	0.524	33.049	**0.926**	0.086	19.974	353.504	**4 GB**
	Ours	1.091	0.892	**33.302**	**0.926**	**0.078**	41.646	**911.262**	6 GB
RGB-D	ORB-SLAM3 [2]	1.833	1.478	-	-	-	**52.209**	-	0
	DROID-SLAM [33]	0.634	0.248	-	-	-	36.452	-	11 GB
	BundleFusion [6]	1.606	0.969	23.839	0.822	0.197	8.630	-	5 GB
	Nice-SLAM [45]	2.350	1.590	26.158	0.832	0.232	2.331	0.611	12 GB
	Orbeez-SLAM [4]	0.888	0.562	32.516	0.916	0.112	41.333	1.401	6 GB
	ESLAM [16]	**0.568**	0.274	30.594	0.866	0.162	6.687	2.626	21 GB
	Co-SLAM [35]	1.158	0.602	30.246	0.864	0.175	14.575	3.745	**4 GB**
	Go-SLAM [43]	0.571	**0.218**	24.158	0.766	0.352	19.437	0.444	24 GB
	Ours (Jetson)	0.581	0.289	31.978	0.916	0.101	17.926	116.395	**4 GB**
	Ours (Laptop)	0.590	0.289	34.853	**0.944**	0.062	20.597	396.082	**4 GB**
	Ours	0.604	0.298	**34.958**	0.942	**0.059**	42.485	**1084.017**	5 GB

Compact 3DGS

To solve the problem of requiring a large amount of memory and storage:

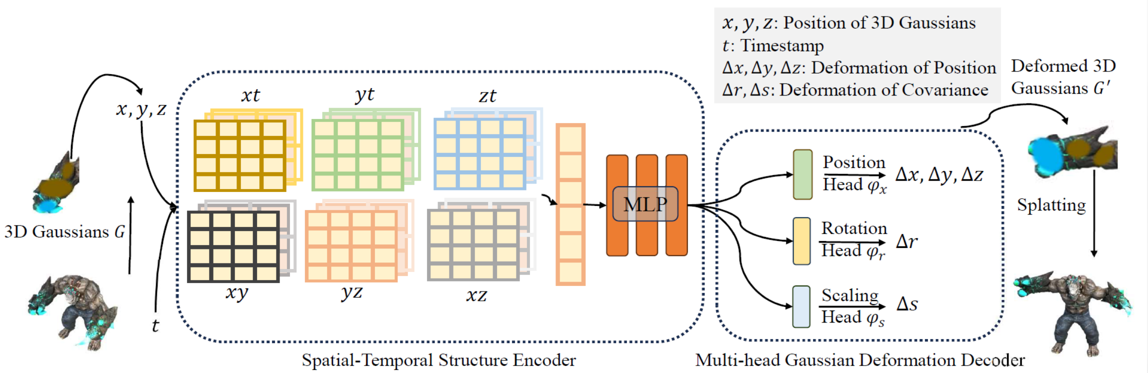
- Learnable mask strategy to identifies and removes non-essential Gaussians (Volume as well as transparency);
- Employing grid-based neural field (hash-based) to represent the view-dependent color rather than SH;
- Learn codebooks to represent geometric attributes of Gaussian;

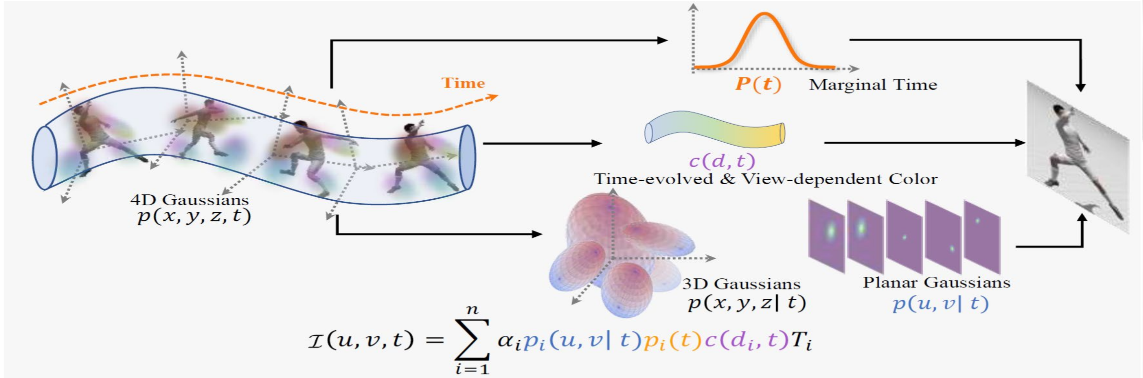


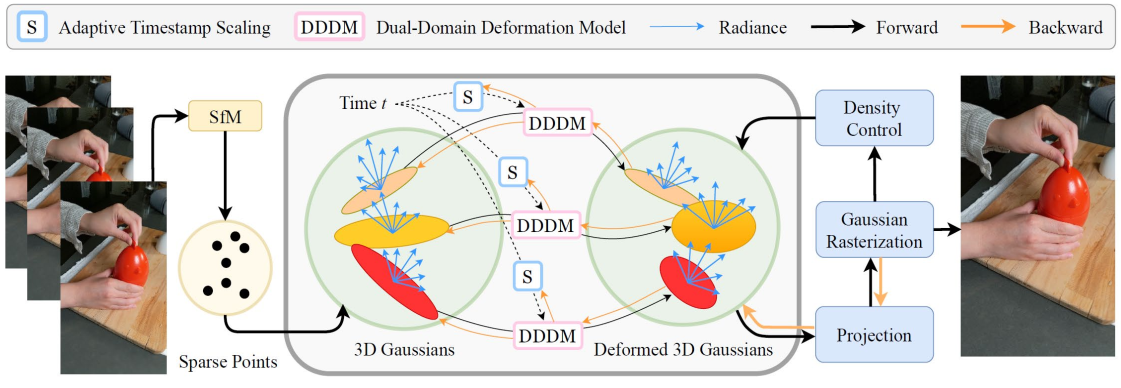
4D Gaussian Splatting

4D NeRF are expected to show consistent appearance, geometry, and motions from arbitrary viewpoints, while the 4DGS is also proposed to generate dynamic scenes;

- 3DGS+deformation MLP network (learn how to deform the static 3DGS at different time stamps);
- Considering the spacetime as an entirety, integrating the 3D Gaussian with temporal extension into 4D Gaussian; 4D Gaussian can be decomposed into a conditional 3D Gaussian and a marginal 1D Gaussian (temporal);
- Formulate the 4D scene as a set of deformable 3D Gaussian points;







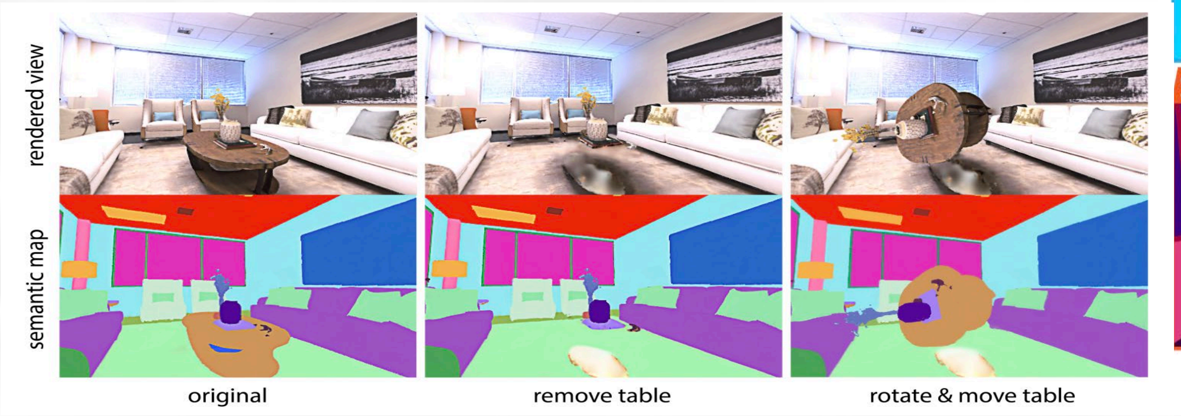
SGS-SLAM: Semantic gaussian splatting for neural dense slam

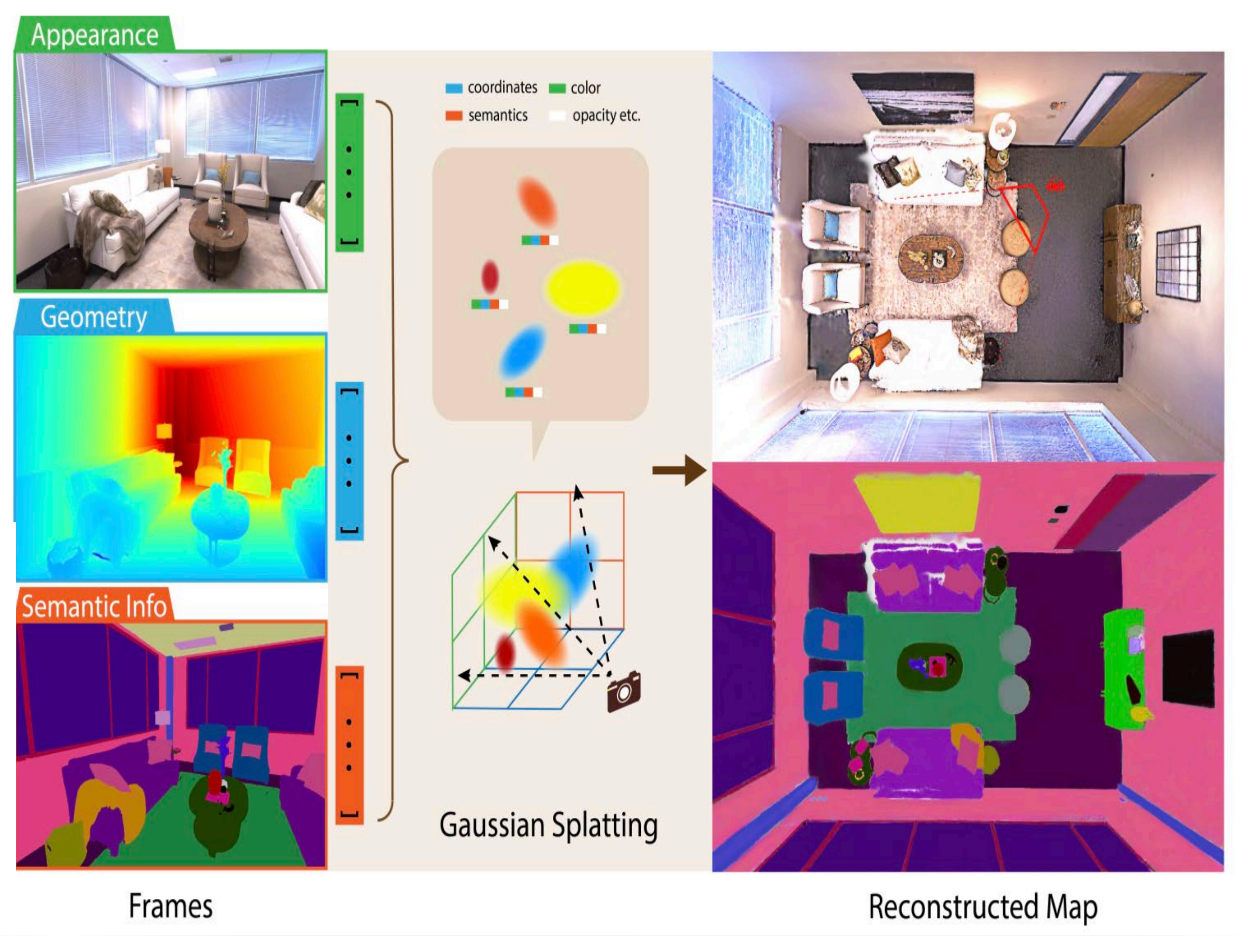
香 港 大 學
THE UNIVERSITY OF HONG KONG

- First semantic dense visual SLAM system grounded in 3D Gaussians;

$$S_{\text{pix}} = \sum_{i=1}^{n} s_i f^{2D}_{i,\text{pix}} \prod_{j=1}^{i-1} (1 - f^{2D}_{j,\text{pix}})$$

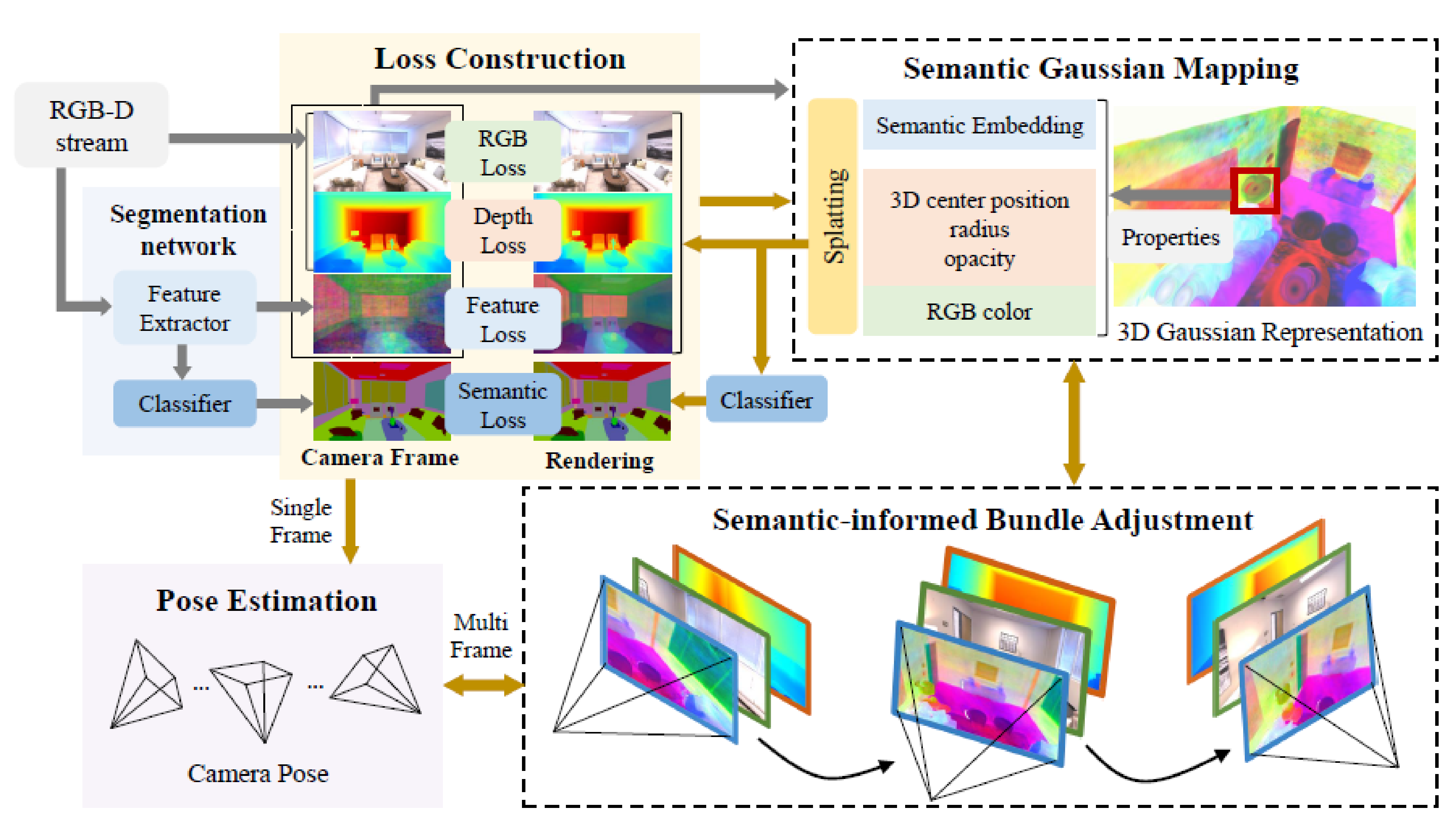
the semantic color associated with the Gaussian





SemGauss-SLAM: Dense Semantic Gaussian Splatting SLAM

- Nothing new;



High-Fidelity SLAM Using Gaussian Splatting with Rendering-Guided Densification and Regularized Optimization

香 港 大 學
THE UNIVERSITY OF HONG KONG

- Gaussian densification strategy based on the rendering loss to map unobserved areas and refine reobserved areas;
- Regularization, re-rendering loss;

$$C = \sum_{i \in N} c_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j)$$

$$D = \sum_{i \in N} z_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j)$$

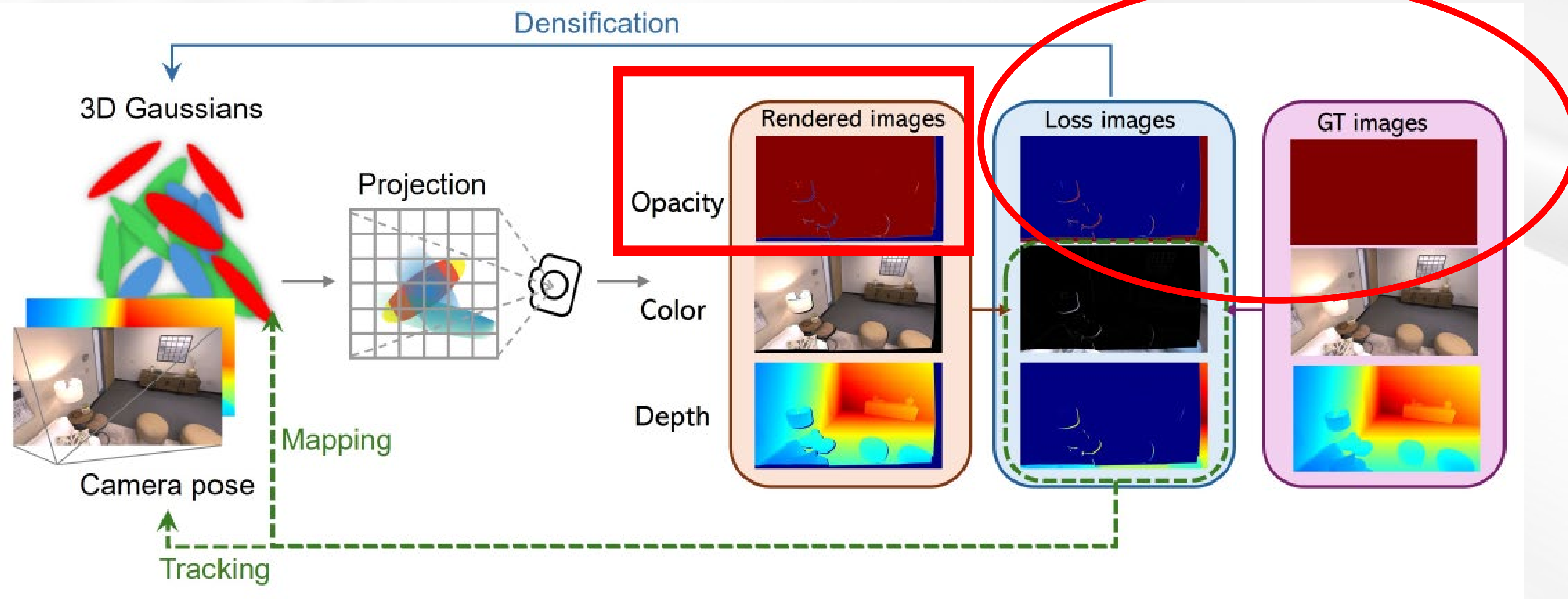
$$\alpha_i = o_i \cdot \exp\left[-\frac{1}{2}(x - \mu')(\Sigma')^{-1}(x - \mu')\right].$$

$$\mathcal{L}'_{\text{mapping}} = \mathcal{L}_{\text{mapping}} + \mathcal{L}_{\text{reg}}$$

$$\mathcal{L}_{\text{mapping}} = \lambda_{\text{color}} |\hat{C} - C| + \lambda_{\text{depth}} |\hat{D} - D| + \lambda_{\text{SSIM}} \text{SSIM}(\hat{C}, C)$$

$$\mathcal{L}_{\text{reg}} = \sum_{i \in G} \Omega_i^s |s_i^t - s_i^*| + \Omega_i^c |r_i^t - r_i^*| + \Omega_i^d |z_i^t - z_i^*|,$$

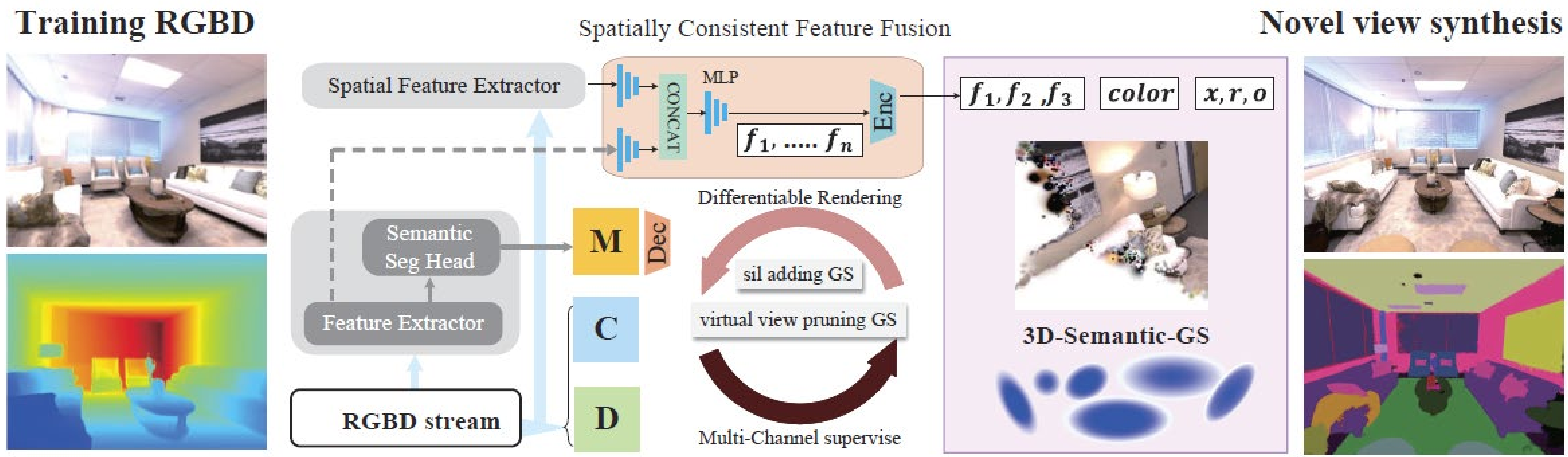
$$\boxed{O = \sum_{i \in N} \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j)}$$



NEDS-SLAM (Semantic 3DGS SLAM)

香 港 大 學
THE UNIVERSITY OF HONG KONG

- Encoder-decoder to compress the high-dimensional semantic features into a compact 3D Gaussian representation;
- Virtual Camera View (just like sliding window) Pruning method to eliminate outlier GS points;
- Combines the appearance features estimated by **DepthAnything** with the semantic features extracted from pretrained model;
- Deeply describe the SplaTAM updating process; Using similar scheme to pruning Gaussians as MonoGS;



MM3DGS SLAM: Multi-modal 3D Gaussian Splatting for SLAM Using Vision, Depth, and Inertial Measurements

香 港 大 學
THE UNIVERSITY OF HONG KONG

- VIO + 3DGS: depth and IMU pre-integration for pose optimization;
- Integrating inertial measurements and depth estimates from an unposed monocular RGB or RGB-D camera into SLAM framework using 3D Gaussians for scene representation;

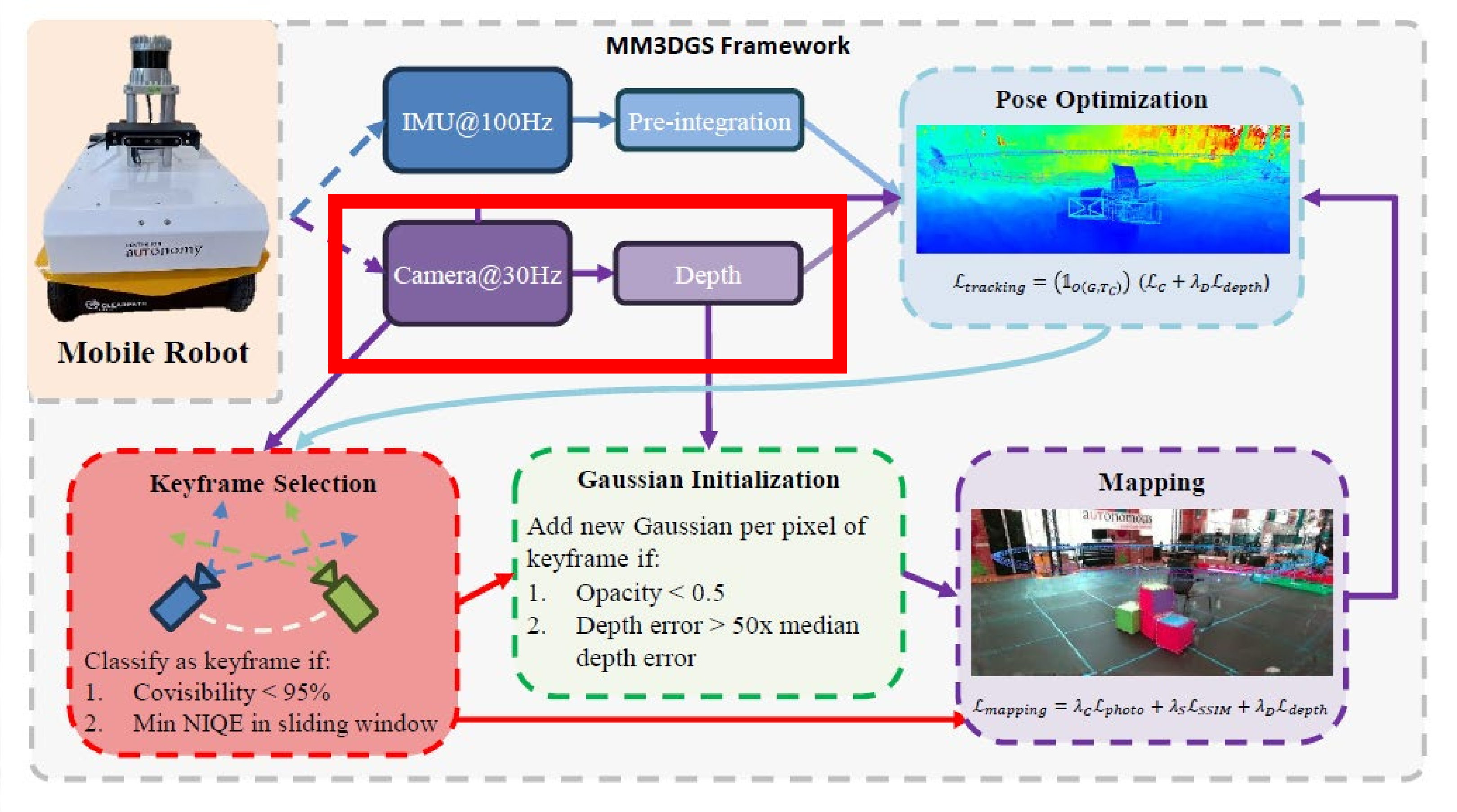


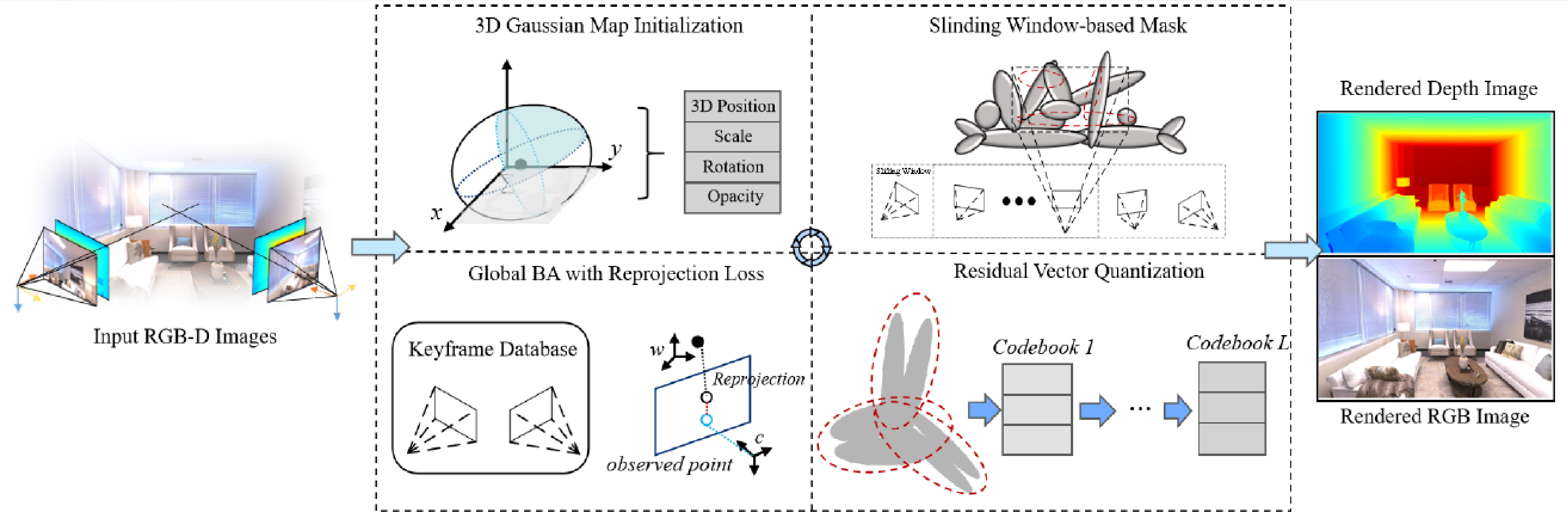
TABLE II: Monocular RGB configuration results on the TUM RGB-D dataset. ATE RMSE ↓ is in cm. Monocular RGB SLAM provides comparable performance as RGB-D baselines.

Method	fr1/desk	fr1/desk2	fr2/xyz
SplaTAM (RGB-D)	**3.35**	6.54	**1.24**
Ours (RGB)	3.51	**5.78**	2.04

Compact 3D Gaussian Splatting For Dense Visual SLAM

香 港 大 學
THE UNIVERSITY OF HONG KONG

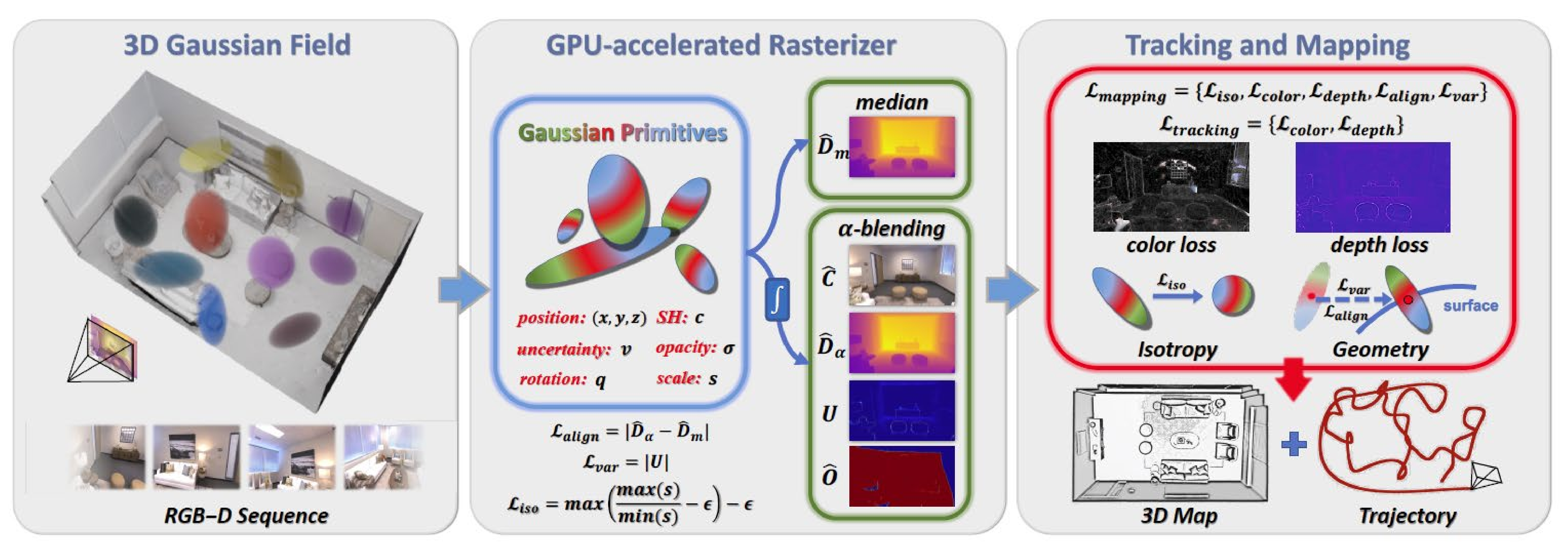
- To address he critical high memory demand and slow training speed issue;
- A sliding window-based masking strategy is first proposed to reduce the redundant ellipsoids;
- Novel geometry codebook to compress 3D Gaussian geometric attributes;



CG-SLAM: Efficient Dense RGB-D SLAM in a Consistent Uncertainty-aware 3D Gaussian Field

香 港 大 學
THE UNIVERSITY OF HONG KONG

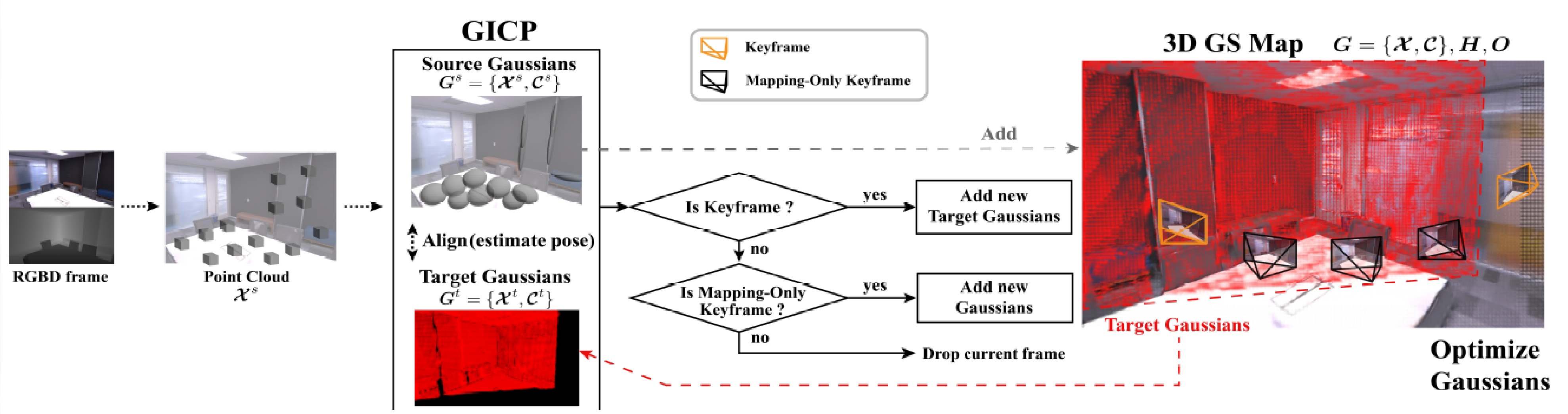
- A scale regularization;
- A novel depth uncertainty model is proposed to ensure the selection of valuable Gaussian primitives during optimization;



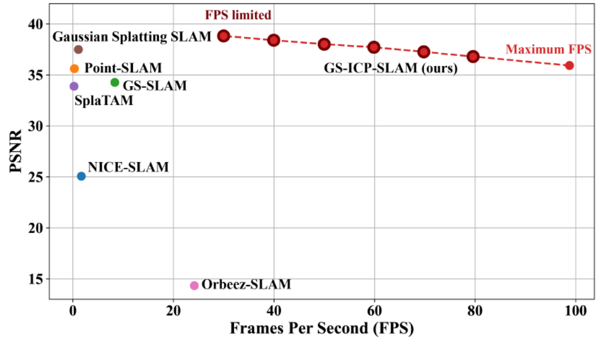
RGBD GS-ICP SLAM

香 港 大 學
THE UNIVERSITY OF HONG KONG

- GICP+3DGS with keyframe selection methods;
- Use G-ICP to align the current frame with the 3D GS map which contains covariance (solely need to compute the covariance for the current frame);
- When adding keyframes to the 3D GS map, utilize the covariance computed in GICP during tracking (no need for densifying or opacity reset);

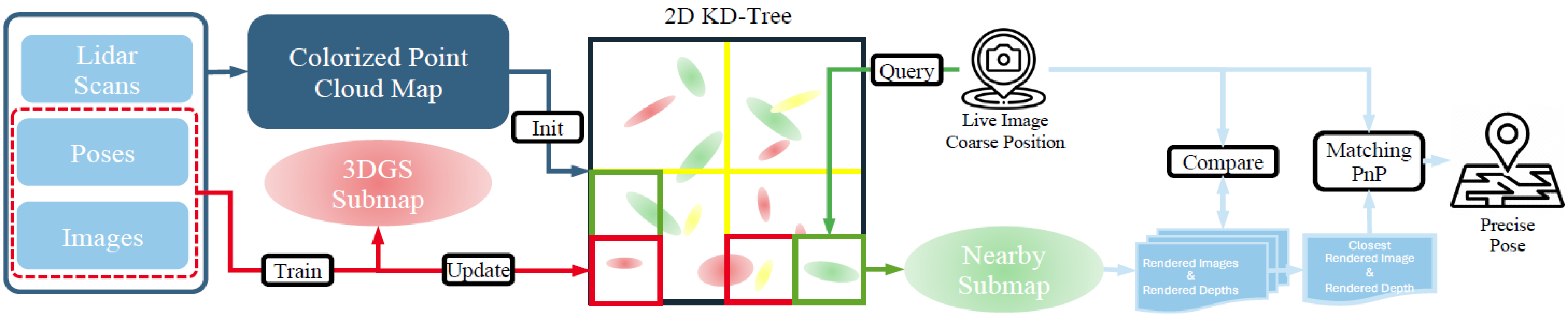


	Method	fr1/desk	fr2/xyz	fr3/office	Avg.
Decoupled	ORB-SLAM3 [4]	**1.7**	0.4	1.7	**1.3**
	Photo-SLAM [12]	2.6	**0.3**	**1.0**	**1.3**
Coupled	NICE-SLAM* [47]	2.8	2.1	7.2	4.0
	Point-SLAM* [32]	**2.7**	**1.3**	3.9	2.6
	GS-SLAM [44]	3.3	**1.3**	6.6	3.7
	SplaTAM* [14]	3.3	**1.3**	5.1	3.2
	Ours (limited to 30 FPS)	**2.7**	1.8	**2.7**	**2.4**



3DGS-ReLoc: 3D Gaussian Splatting for Map Representation and Visual ReLocalization

- By leveraging LiDAR data to initiate the training of the 3D Gaussian Splatting map, the system constructs maps that are both detailed and geometrically accurate;
- The combination of 2D voxel map and KD-tree to mitigate memory usage and facilitate rapid spatial queries;
- For visual localization tasks;



HGS-Mapping: Online Dense Mapping Using Hybrid Gaussian Representation in Urban Scenes

- Hybrid Gaussian Representation, which is comprised of Sphere Gaussian, 3D Gaussian, and 2D Gaussian Plane components;
- Implementing an adaptive update method for Gaussians, which dynamically densifies Gaussians based on the reconstruction loss and prunes the Gaussians of low importance;

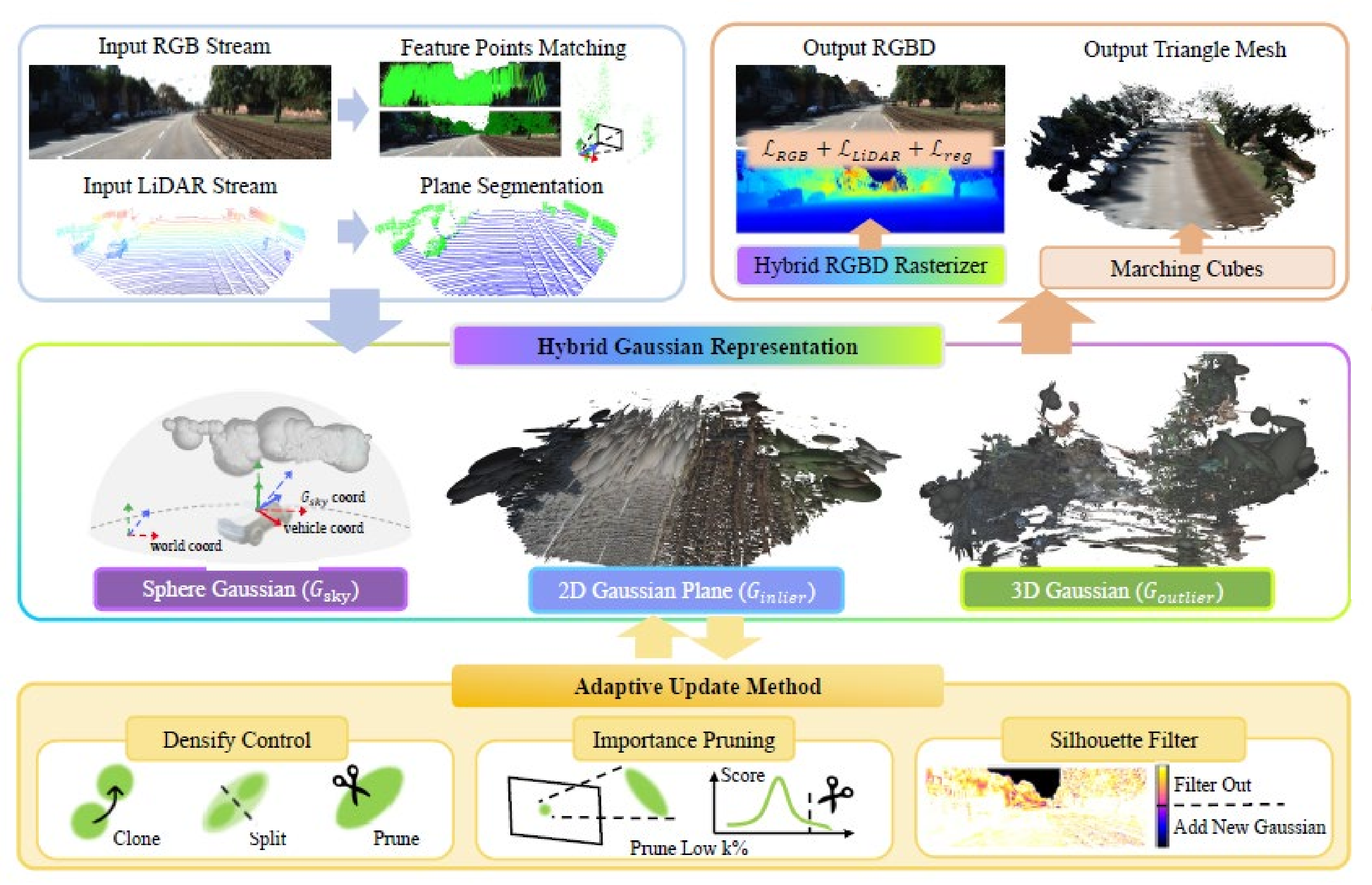


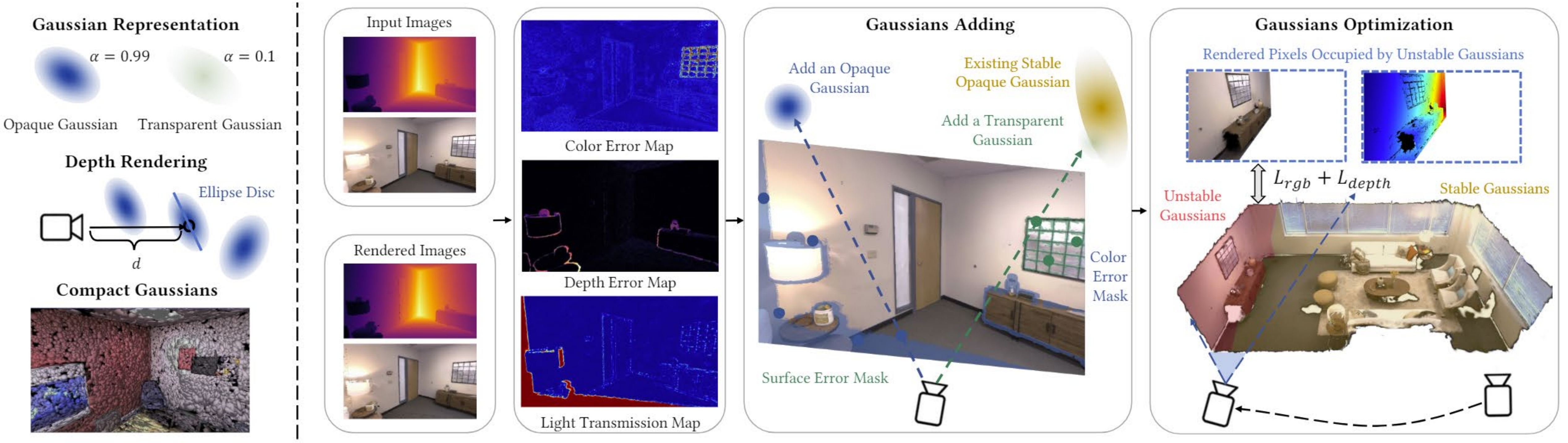
Table 1: Quantitative results of RGB rendering on four urban datasets [2,3,9,32].

	VKITTI2		KITTI		nuScenes		Waymo	
	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑
Mip NeRF 360	24.878	0.7502	21.395	0.6499	28.530	0.8751	24.826	0.8440
Instant-NGP	21.586	0.6305	22.530	0.7418	27.046	0.8352	26.154	0.8557
3DGS(LIDAR)	24.236	0.7928	19.227	0.6185	28.091	0.9133	17.664	0.8491
Splatam(masked)	20.003	0.7401	14.778	0.4783	18.998	0.7459	20.371	0.7704
Ours	29.114	0.9019	22.520	0.8007	30.862	0.9404	26.445	0.8832

RTG-SLAM: Real-time 3D Reconstruction at Scale Using Gaussian Splatting

香 港 大 學
THE UNIVERSITY OF HONG KONG

- Each Gaussian to be either opaque or nearly transparent, with the opaque ones fitting the surface and dominant colors, and transparent ones fitting residual colors; Letting a single opaque Gaussian well fit a local surface region without the need of multiple overlapping Gaussians, hence largely reducing the memory and computation cost;
- Categorizing Gaussians into stable and unstable ones; Only optimizing the unstable Gaussians and only render the pixels occupied by unstable Gaussians;
- Frame-to-model ICP for tracking;



Results from the paper

香 港 大 學
THE UNIVERSITY OF HONG KONG

- https://gapszju.github.io/RTG-SLAM/

Table 1: Comparison of time and memory performance on Replica (Off 0) and Azure Dataset (Home). Here ✕ means out of memory.

Method	Dataset	Tracking /Frame	Mapping /Iteration	Mapping /Frame	FPS	Model Size (MB)	Memory Cost (MB)
NICE-SLAM [2022b]	Replica	1.05s	60.9ms	1.03s	0.95	87	9890
	Azure	0.68s	116.5ms	1.58s	0.63	136	10057
Co-SLAM [2023]	Replica	0.11s	7.8ms	0.10s	9.26	**7**	7899
	Azure	0.11s	7.2ms	0.12s	8.65	**7**	17342
ESLAM [2023]	Replica	0.15s	16.7ms	0.10s	6.80	46	18777
	Azure	0.13s	15.4ms	0.11s	7.54	139	✕
Point-SLAM [2023]	Replica	1.05s	38.1ms	2.27s	0.44	15431	9890
	Azure	4.54s	68.4ms	4.00s	0.22	42536	9950
SplaTAM [2023]	Replica	1.16s	32.1ms	1.96s	0.51	310	9166
	Azure	2.00s	53.4ms	3.22s	0.31	520	✕
Ours	Replica	**0.02s**	**3.5ms**	**0.05s**	**17.24**	71	**2751**
	Azure	**0.03s**	**4.3ms**	**0.05s**	**17.90**	399	**8782**

Table 2: Comparison of tracking accuracy (unit: *cm*) on TUM-RGBD.

Method	fr1_desk	fr2_xyz	fr3_office	Avg.
NICE-SLAM[2022b]	4.30	31.73	3.87	13.28
Co-SLAM[2023]	2.92	1.75	3.55	2.74
ESLAM[2023]	2.49	1.11	2.74	2.11
Point-SLAM[2023]	2.56	1.20	3.37	2.38
SplaTAM[2023]	3.33	1.55	5.28	3.39
Ours	1.66	**0.38**	1.13	1.06
ElasticFusion[2015]	2.53	1.17	2.52	2.07
ORB-SLAM2[2017]	**1.60**	0.40	**1.00**	**1.00**
BAD-SLAM[2019]	1.70	1.10	1.70	1.50

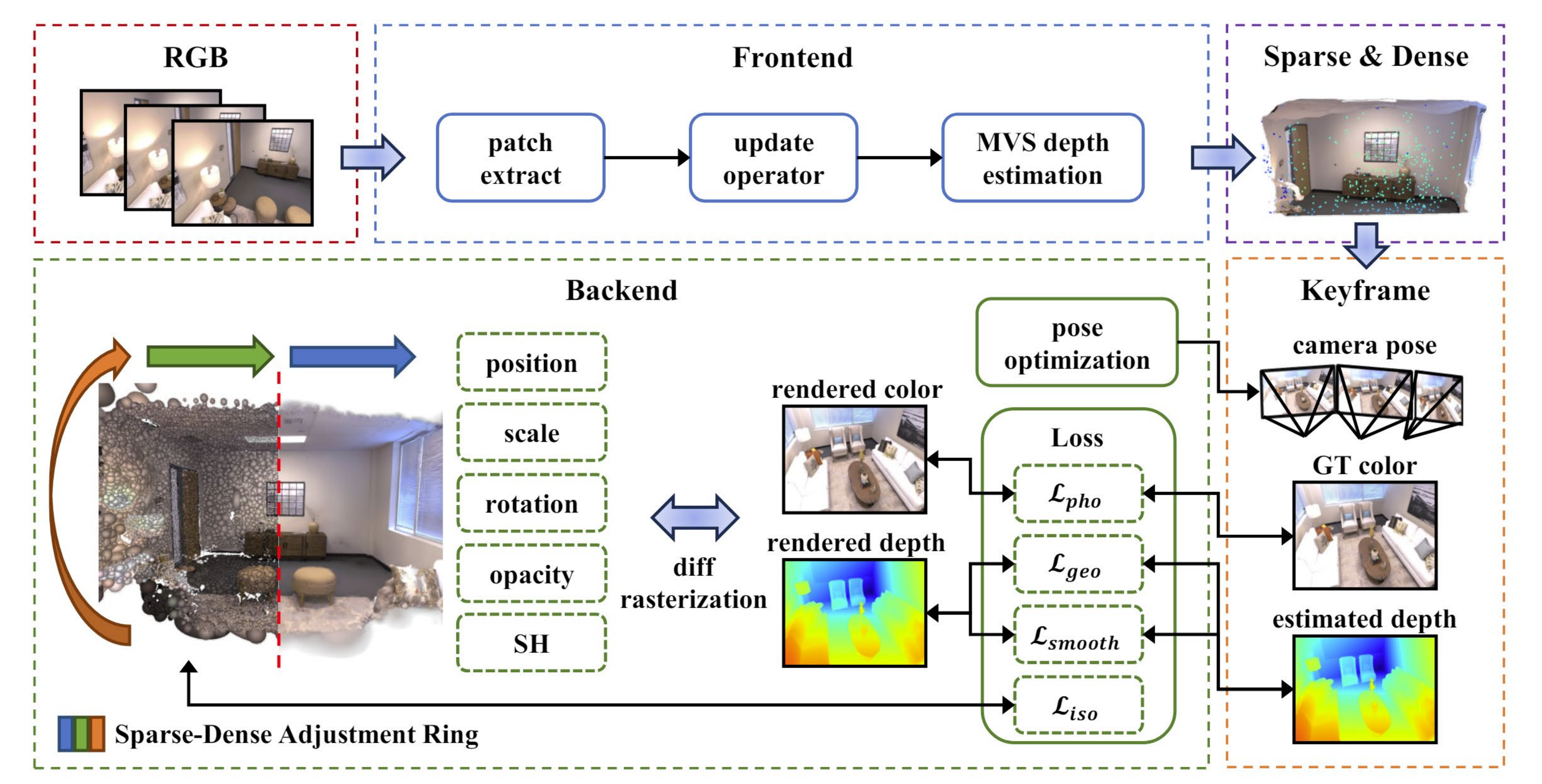
Table 3: Comparison of geometry accuracy on ScanNet++.

Method	Acc.↓	Acc. Ratio↑	Comp.↓	Comp. Ratio↑
NICE-SLAM[2022b]	4.45	74.49	2.04	86.63
Co-SLAM[2023]	5.26	78.86	1.06	96.25
ESLAM[2023]	4.43	74.51	1.05	97.42
Point-SLAM[2023]	**0.67**	**99.12**	**0.68**	**98.94**
SplaTAM[2023]	1.32	95.31	1.54	93.55
Ours	0.95	96.41	1.11	97.16

MGS-SLAM: Monocular Sparse Tracking and Gaussian Mapping with Depth Smooth Regularization

香 港 大 學
THE UNIVERSITY OF HONG KONG

- The sparse visual odometry tracks camera poses in RGB stream, while Gaussian Splatting handles map reconstruction;
- The fifth monocular Mono-GS: MVS depth estimation network + Sparse-Dense Adjustment Ring (scale consistency);



Monocular Gaussian SLAM with Language Extended Loop Closure

- Monocular RGB (DPVO)+language-extended loop closure module;
- Training the 3DGS with history keyframes within a sliding window;

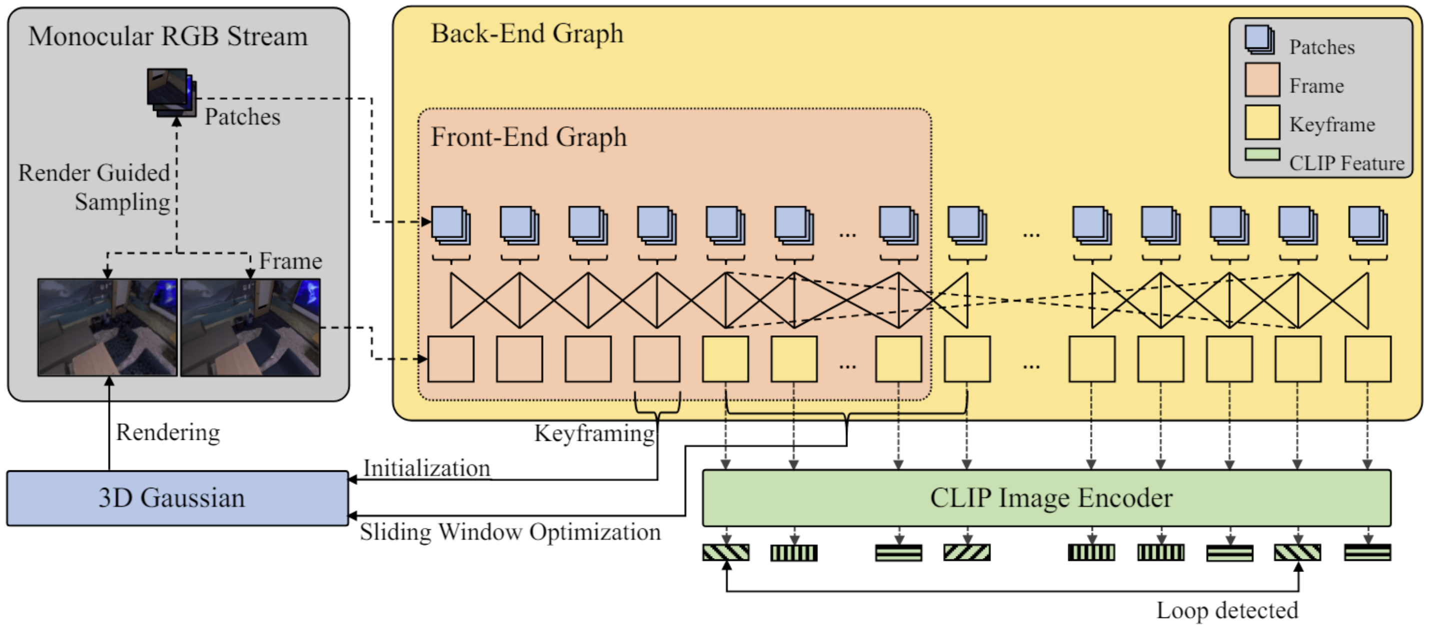


Fig. 1: System Overview. Our system consists of the following components: 3D Gaussian map, CLIP feature-based loop closure module, Front-End and Back-End Graph for optimization based on DPVO [37]. 3D Gaussian map is initialized by optimized patches and trained using keyframes within the sliding window, and the images rendered with it in turn guide the sampling of the patches. The loop closure module continually detects loops between the current keyframe and history keyframes. Global optimization is performed on Back-End Graph each time a new keyframe is added.

Supasses RGB-D

香 港 大 學
THE UNIVERSITY OF HONG KONG

Table 2: Rendering Metrics on Replica dataset. Results of [17, 30, 41, 45] are taken from [17]. Cell color indicates best, second best and third best.

Method	Metric	Office0	Office01	Office02	Office03	Office04	Room0	Room1	Room2	Avg
Vox-Fusion [41]	PSNR ↑	27.79	29.83	20.33	23.47	25.21	22.39	22.36	23.92	24.41
	SSIM ↑	0.86	0.88	0.79	0.80	0.85	0.68	0.75	0.80	0.80
	LPIPS ↓	0.24	0.18	0.24	0.21	0.20	0.30	0.27	0.23	0.24
NICE-SLAM [45]	PSNR ↑	29.07	30.34	19.66	22.23	24.94	22.12	22.47	24.52	24.42
	SSIM ↑	0.87	0.89	0.80	0.80	0.86	0.69	0.76	0.81	0.81
	LPIPS ↓	0.23	0.18	0.24	0.21	0.20	0.33	0.27	0.21	0.23
Point-SLAM [30]	PSNR ↑	38.26	39.16	33.99	33.48	33.49	32.40	34.08	35.50	35.17
	SSIM ↑	0.98	0.99	0.96	0.96	0.98	0.97	0.98	0.98	0.98
	LPIPS ↓	0.10	0.12	0.16	0.13	0.14	0.11	0.12	0.11	0.12
SplaTAM [17]	PSNR ↑	38.26	39.17	39.17	29.70	31.81	32.86	33.89	35.25	34.11
	SSIM ↑	0.98	0.98	0.97	0.95	0.95	0.98	0.97	0.98	0.97
	LPIPS ↓	0.09	0.09	0.10	0.12	0.15	0.07	0.10	0.08	0.10
Ours	PSNR ↑	38.57	38.87	32.21	31.65	31.81	30.35	31.65	33.15	33.59
	SSIM ↑	0.96	0.96	0.93	0.93	0.94	0.88	0.91	0.94	0.93
	LPIPS ↓	0.16	0.21	0.23	0.22	0.21	0.23	0.26	0.23	0.22

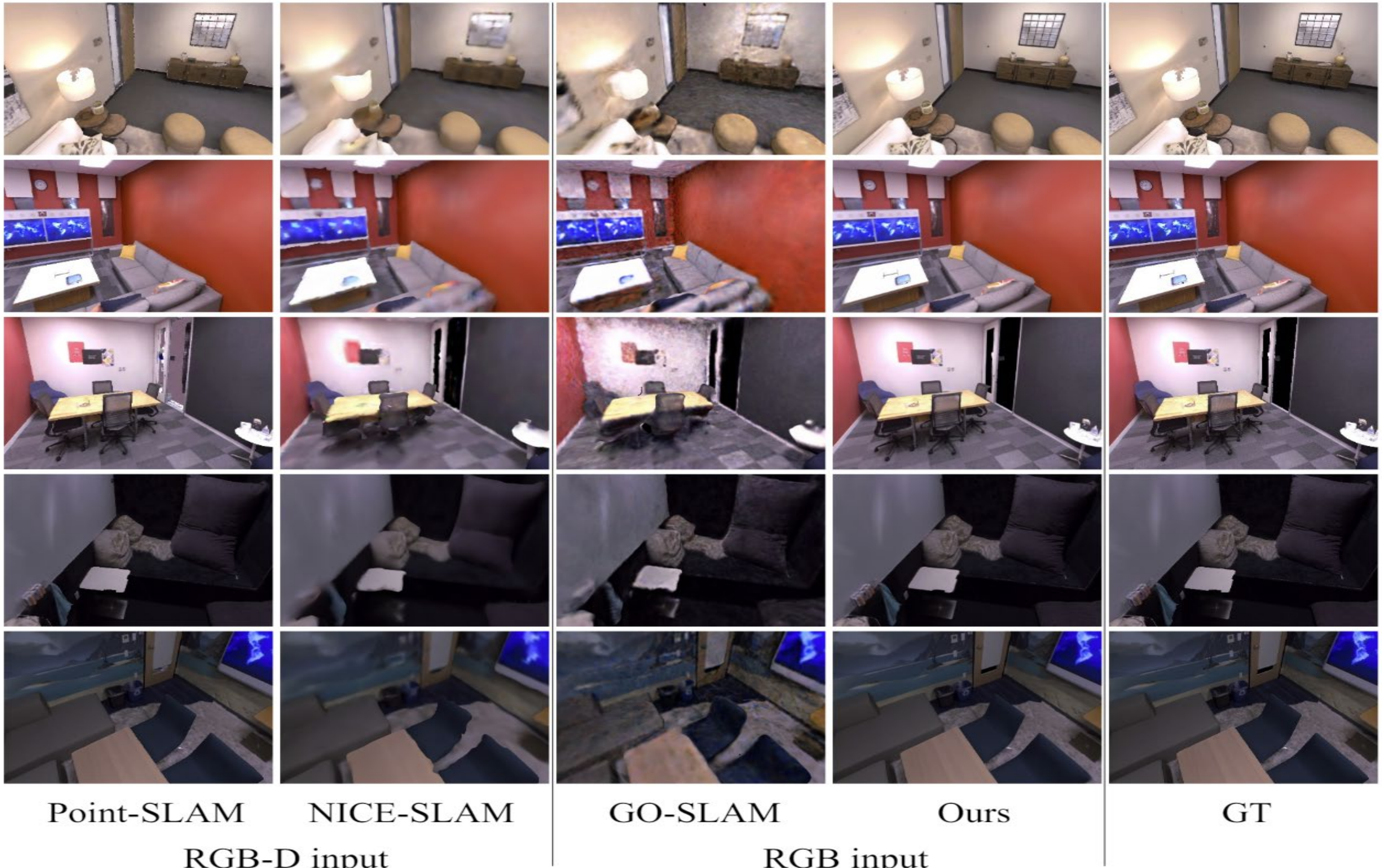
Table 4: ATE RMSE[m] on TUM RGB-D benchmark. Results of [17, 25, 30, 38, 41, 45] are taken from [17]. Results of [21] are partially blank since they are not provided in the paper. Cell color indicates best, second best and third best.

	Method	fr1/desk	fr1/desk2	fr1/room	fr2/xyz	fr3/off	Avg
RGB-D	ElasticFusion [38]	0.0253	0.0683	0.2149	0.0117	0.0252	0.0691
	ORBSLAM2 [25]	0.0160	0.0220	0.0470	0.0040	0.0100	0.0198
	Vox-Fusion [41]	0.0352	0.0600	0.1953	0.0149	0.2601	0.1131
	Nice-SLAM [45]	0.0426	0.0499	0.3449	0.3173	0.0387	0.1587
	Point-SLAM [30]	0.0434	0.0454	0.3092	0.0131	0.0348	0.0892
	SplaTAM [17]	0.0335	0.0654	0.1113	0.0124	0.0516	0.0548
Mono.	DROID-SLAM [36]	0.0177	0.0267	0.0433	0.0046	0.0289	0.0242
	GO-SLAM [43]	0.0150	0.0487	0.4096	0.0042	0.0210	0.0997
	GaussianSplatting-SLAM [21]	0.0415	-	-	0.0479	0.0439	-
	Ours	0.0282	0.0357	0.1641	0.0082	0.0322	0.0537

Table 5: ATE[m] on EuRoC dataset. Results of [4,24,25,36,43] are taken from [43]. Results of [4, 24, 25] are partially blank because they fail in these scenarios. Cell color indicates best, second best and third best.

	Method	MH01	MH02	MH03	MH04	MH05	V101	V102	V103	V201	V202	V203	Avg
Stereo	ORB-SLAM2 [25]	0.0350	0.0180	0.0280	0.1190	0.0600	0.0350	0.0200	0.0480	0.0370	0.0350	-	-
	ORB-SLAM3 [4]	0.0290	0.0190	0.0240	0.0850	0.0520	0.0350	0.0250	0.0610	0.0410	0.0280	0.5210	0.0840
	DROID-SLAM [36]	0.0150	0.0130	0.0350	0.0480	0.0400	0.0370	0.0110	0.0200	0.0180	0.0150	0.0170	0.0240
	GO-SLAM [43]	0.0160	0.0140	0.0230	0.0450	0.0450	0.0370	0.0110	0.0230	0.0160	0.0100	0.0220	0.0240
Mono.	ORB-SLAM [24]	0.0710	0.0670	0.0710	0.0820	0.0600	0.0150	0.0020	-	0.0210	0.0180	-	-
	ORB-SLAM3 [4]	0.0160	0.0270	0.0280	0.1380	0.0720	0.0330	0.0150	0.0330	0.0230	0.0290	-	-
	DROID-SLAM [36]	0.0130	0.0140	0.0220	0.0430	0.0430	0.0370	0.0120	0.0200	0.0170	0.0130	0.0140	0.0220
	GO-SLAM [43]	0.0172	0.0186	3.3548	5.2975	4.9379	0.7667	0.0123	1.3245	0.0249	0.0110	0.0179	1.4348
	Ours	0.0104	0.0131	0.0201	0.0458	0.0413	0.0341	0.0082	0.0153	0.0198	0.0091	0.0262	0.0221

Results



Results from the paper

THE UNIVERSITY OF HONG KONG 香 港 大 學

TABLE I

ATE [CM] RESULTS ON TUM DATASET

Input	Method	fr1/desk	fr2/xyz	fr3/office	Avg.
RGB-D	iMAP	4.90	2.00	5.80	4.23
	NICE-SLAM	4.26	6.19	6.87	5.77
	Vox-Fusion	3.52	1.49	26.01	10.34
	SplaTAM	**3.35**	**1.24**	**5.16**	**3.25**
Mono.	DSO	22.40	1.10	9.50	11.00
	DROID-VO	5.20	10.70	7.30	7.73
	DPVO	3.80	0.54	7.00	3.78
	MonoGS	4.15	4.79	4.39	4.44
	Ours	**2.33**	**0.44**	**3.00**	**1.92**

TABLE II

ATE [CM] RESULTS ON REPLICA DATASET

Input	Method	R0	R1	R2	O0	O1	O2	O3	O4	Avg.
RGB-D	iMAP	3.12	2.54	2.31	1.69	1.03	3.99	4.05	1.93	2.58
	NICE-SLAM	0.97	1.31	1.07	0.88	1.00	1.06	1.10	1.13	1.07
	Vox-Fusion	1.37	4.70	1.47	8.48	2.04	2.58	1.11	2.94	3.09
	SplaTAM	**0.31**	**0.40**	**0.29**	**0.47**	**0.27**	**0.29**	**0.32**	**0.55**	**0.36**
Mono.	DROID-VO	0.50	0.70	0.30	0.98	0.29	0.84	0.45	1.53	0.70
	NICER-SLAM	1.36	1.60	1.14	2.12	3.23	2.12	1.42	2.01	1.88
	DPVO	0.49	0.54	0.54	0.77	0.36	0.57	0.46	0.57	0.54
	MonoGS	9.94	✗	✗	✗	✗	✗	11.58	✗	10.76
	Ours	**0.36**	**0.35**	**0.32**	**0.35**	**0.28**	**0.26**	**0.32**	**0.34**	**0.32**

TABLE III

RENDERING PERFORMANCE ON REPLICA DATASET. BEST RESULTS ARE HIGHLIGHTED AS FIRST, SECOND, AND THIRD

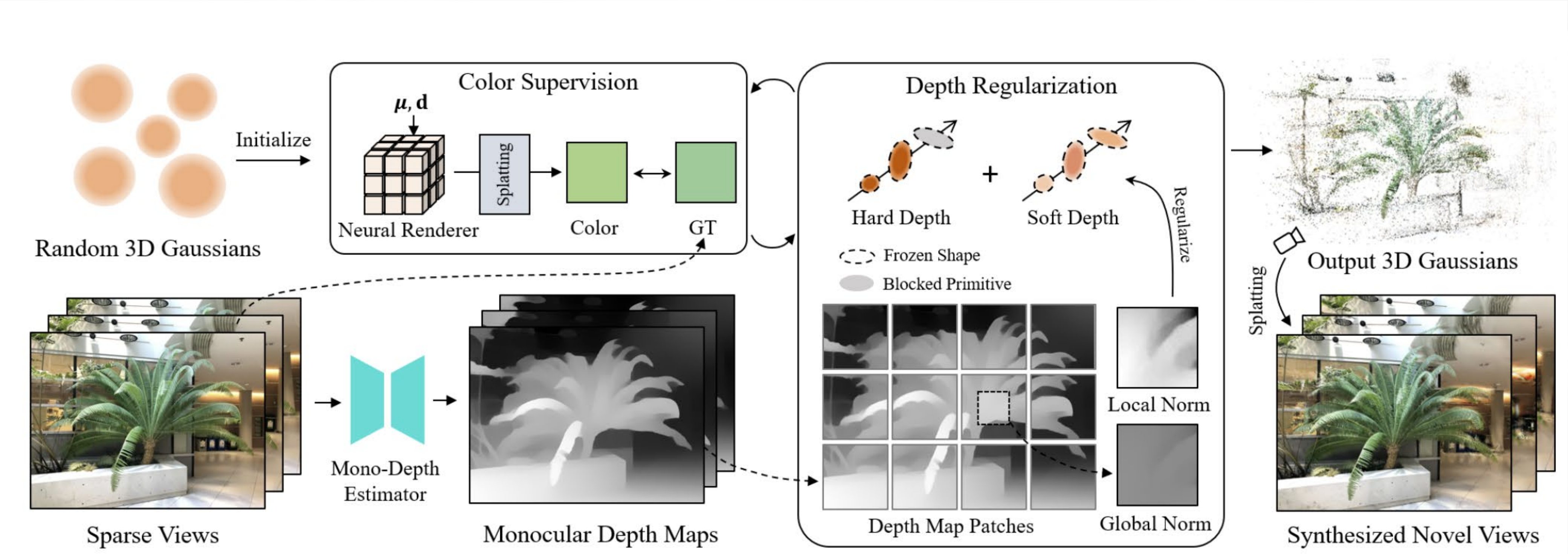
Method	Metric	R0	R1	R2	O0	O1	O2	O3	O4	Avg.
NICE-SLAM	PSNR[dB]↑	22.12	22.47	24.52	29.07	30.34	19.66	22.23	24.94	24.42
	SSIM↑	0.689	0.757	0.814	0.874	0.886	0.797	0.801	0.856	0.809
	LPIPS↓	0.330	0.271	0.208	0.229	0.181	0.235	0.209	0.198	0.233
Vox-Fusion	PSNR[dB]↑	22.39	22.36	23.92	27.79	29.83	20.33	23.47	25.21	24.41
	SSIM↑	0.683	0.751	0.798	0.857	0.876	0.794	0.803	0.847	0.801
	LPIPS↓	0.303	0.269	0.234	0.241	0.184	0.243	0.213	0.199	0.236
GO-SLAM	PSNR[dB]↑	23.25	20.70	21.08	21.44	22.59	22.33	22.19	22.76	22.04
	SSIM↑	0.712	0.739	0.708	0.761	0.726	0.740	0.752	0.722	0.733
	LPIPS↓	0.222	0.492	0.317	0.319	0.269	0.434	0.396	0.385	0.354
NICER-SLAM	PSNR[dB]↑	25.33	23.92	26.12	28.54	25.86	21.95	26.13	25.47	25.41
	SSIM↑	0.751	0.771	0.831	0.866	0.852	0.820	0.856	0.865	0.827
	LPIPS↓	0.250	0.215	0.176	0.172	0.178	0.195	0.162	0.177	0.191
Mono GS	PSNR[dB]↑	25.11	24.66	22.30	28.76	29.17	23.74	23.66	23.99	25.17
	SSIM↑	0.790	0.790	0.843	0.884	0.852	0.840	0.855	0.863	0.840
	LPIPS↓	0.260	0.360	0.351	0.293	0.274	0.290	0.216	0.340	0.298
Ours	PSNR[dB]↑	25.37	27.29	29.64	34.85	34.32	28.17	26.64	32.88	29.90
	SSIM↑	0.796	0.825	0.886	0.932	0.930	0.890	0.855	0.933	0.881
	LPIPS↓	0.153	0.072	0.071	0.069	0.098	0.112	0.086	0.079	0.093

Ji, Yiming, et al. "NEDS-SLAM: A Novel Neural Explicit Dense Semantic SLAM Framework using 3D Gaussian Splatting." *arXiv preprint arXiv:2403.11679* (2024).

Sun, Lisong C., et al. "MM3DGS SLAM: Multi-modal 3D Gaussian Splatting for SLAM Using Vision, Depth, and Inertial Measurements." arXiv preprint arXiv:2404.00923 (2024).

Dngaussian: Optimizing sparse-view 3d gaussian radiance fields with global-local depth normalization

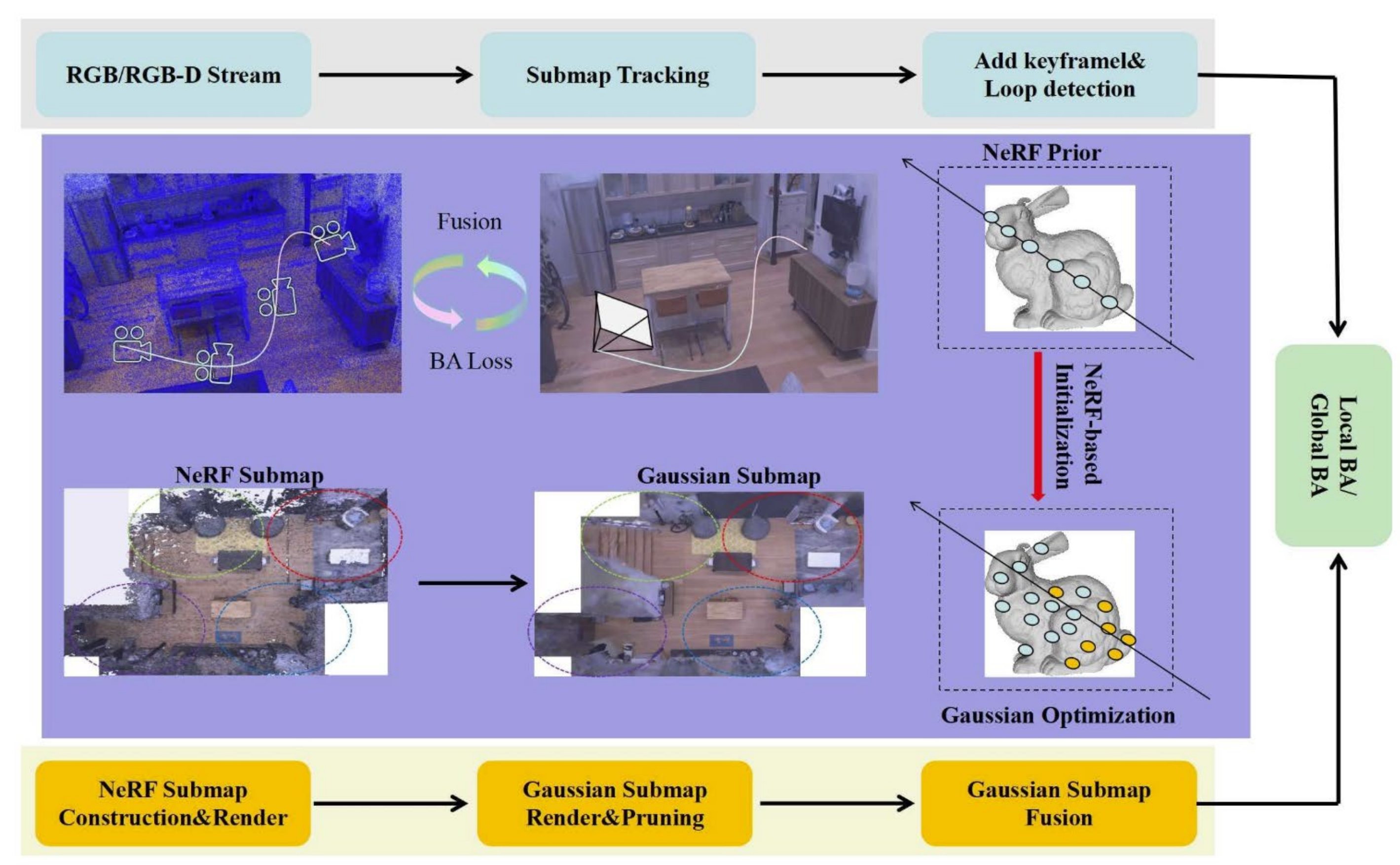
- 3DGS encounters degradation when input views decrease;
- Restoring accurate scene geometry under coarse monocular depth supervision while maintaining a fine-grained color appearance;
- Exploring distilling depth information from pre-trained monocular depth estimators to rectify the Gaussian fields of the ill-learned geometry, and pursue higher quality and efficiency for few-shot novel view synthesis;



NGM-SLAM: Gaussian Splatting SLAM with Radiance Field Submap

香 港 大 學
THE UNIVERSITY OF HONG KONG

- Utilizing neural radiance field + 3DGS, for large-scale scene and loop correction;
- Constructing submaps based on image stream using NeRF, the neural submaps are utilized to construct Gaussian;

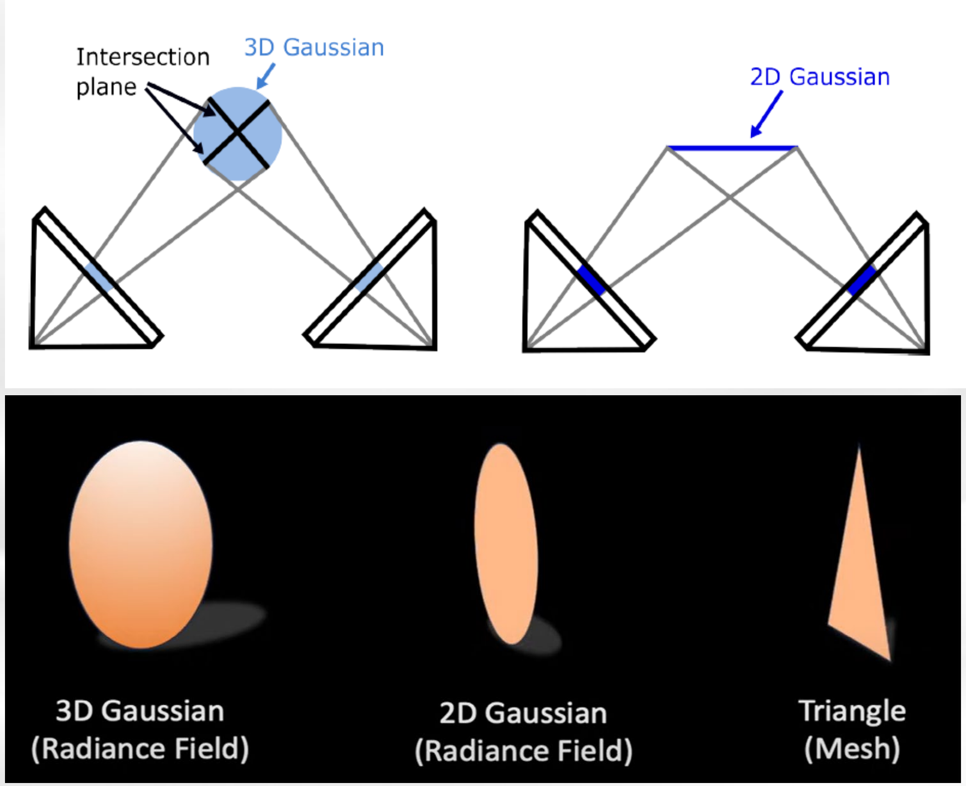


Metrics	PSNR(dB)↑	SSIM↑	LPIPS↓	ATE(cm)↓	Tracking FPS↑	System FPS↑	GPU Usage(G)↓
NICE-SLAM [9]	24.42	0.81	0.23	2.35	2.33	1.91	6.27
Co-SLAM [10]	30.24	0.86	0.18	1.16	14.58	**12.64**	**5.83**
Go-SLAM [10]	24.15	0.77	0.35	1.12	10.74	8.26	14.44
Point-SLAM [31]	33.49	0.97	0.14	0.73	1.10	0.42	7.31
SplaTAM [54]	31.81	0.96	0.16	0.55	1.07	0.42	18.87
MonoGS [41]	34.05	0.96	0.12	0.58	4.58	2.26	27.99
NGM-SLAM(Mono)	35.02	0.96	0.13	8.51	16.11	3.82	7.62
NGM-SLAM	**37.43**	**0.98**	**0.08**	**0.51**	**20.54**	5.71	5.98

Table 1: The average results of five measurements for eight scenes of a sequence of smaller rooms in the Replica[13] dataset are reported for PSNR (dB), SSIM, LPIPS, ATE (cm), Tracking FPS, System FPS, and GPU usage. The best results are bolded, and the second best results are indicated with an underline.

2DGS

- 3DGS evaluates a Gaussian's value at the intersection between a pixel ray and a 3D Gaussian, which leads to inconsistency depth when rendered from different viewpoints;
- 2DGS represents the 3D scene with 2D Gaussian primitives, 2D splatting process utilizing ray-splat intersection and rasterization, while incorporate depth distortion and normal consistency terms to further enhance the quality of the reconstructions;



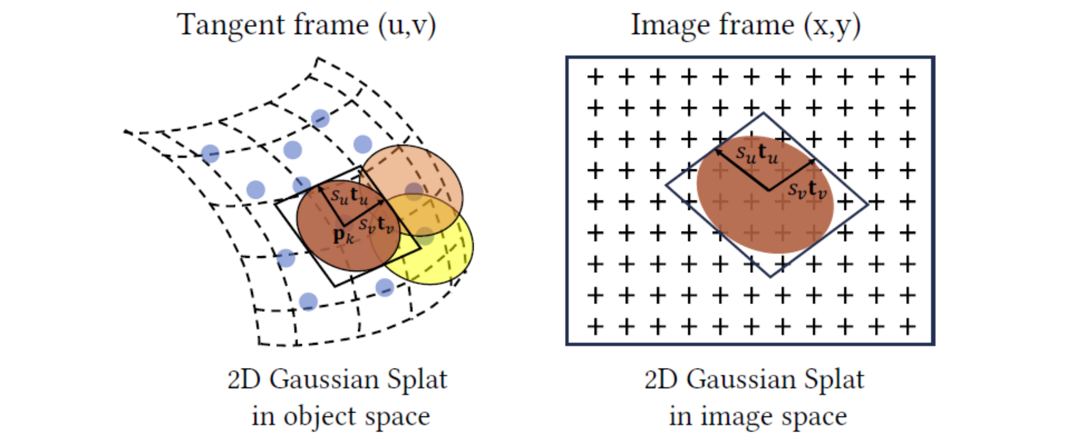


Fig. 3. Illustration of 2D Gaussian Splatting. 2D Gaussian Splats are elliptical disks characterized by a center point $\mathbf{p}_k$, tangential vectors $\mathbf{t}_u$ and $\mathbf{t}_v$, and two scaling factors (s_u and s_v) control the variance. Their elliptical projections are sampled through the ray-splat intersection (Section 4.2) and accumulated via alpha-blending in image space. 2DGS reconstructs surface attributes such as colors, depths, and normals through gradient descent.

2DGS vs 3DGS

accumulated via alpha-blending in image space. 2DGS reconstructs surface attributes such as colors, depths, and normals through gradient descent.

一个2D高斯在3D空间中的切平面为:

$$P(u,v) = \mathbf{p}_k + s_u \mathbf{t}_u u + s_v \mathbf{t}_v v = \mathbf{H}(u,v,1,1)^\top$$

scaling vector $\mathbf{S} = (s_u, s_v)$ $\begin{bmatrix} s_u & 0 & 0 \\ 0 & s_v & 0 \\ 0 & 0 & 0 \end{bmatrix}$

$$\text{where } \mathbf{H} = \begin{bmatrix} s_u \mathbf{t}_u & s_v \mathbf{t}_v & \mathbf{0} & \mathbf{p}_k \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{RS} & \mathbf{p}_k \\ \mathbf{0} & 1 \end{bmatrix}$$

中心点

3×3 rotation matrix $\mathbf{R} = [\mathbf{t}_u, \mathbf{t}_v, \mathbf{t}_w]$

$$\mathbf{t}_w = \mathbf{t}_u \times \mathbf{t}_v.$$

平面法向量 平面切向量

对于点 $\mathbf{u} = (u, v)$ 通过下式获得其高斯值

$$\mathcal{G}(\mathbf{u}) = \exp\left(-\frac{u^2+v^2}{2}\right)$$

⟹ 2D高斯,均值为0,方差为1

2D ~ 2D mapping

2D高斯 image plane

$$\mathbf{x} = (xz, yz, z, z)^\top = \mathbf{W}P(u,v) = \mathbf{W}\mathbf{H}(u,v,1,1)^\top$$

transformation matrix from world space to screen space

Ray-splat Intersection: $\mathbf{h}_u = (\mathbf{WH})^\top \mathbf{h}_x \quad \mathbf{h}_v = (\mathbf{WH})^\top \mathbf{h}_y$

$$\mathbf{h}_x = (-1, 0, 0, x) \quad \mathbf{h}_y = (0, -1, 0, y).$$

对于图像坐标系上的点 x,y 其对应的两个平面坐标代表的平面为

$$\mathbf{h}_u \cdot (u,v,1,1)^\top = \mathbf{h}_v \cdot (u,v,1,1)^\top = 0$$

⟹

$$u(\mathbf{x}) = \frac{\mathbf{h}_u^2 \mathbf{h}_v^4 - \mathbf{h}_u^4 \mathbf{h}_v^2}{\mathbf{h}_u^1 \mathbf{h}_v^2 - \mathbf{h}_u^2 \mathbf{h}_v^1} \qquad v(\mathbf{x}) = \frac{\mathbf{h}_u^4 \mathbf{h}_v^1 - \mathbf{h}_u^1 \mathbf{h}_v^4}{\mathbf{h}_u^1 \mathbf{h}_v^2 - \mathbf{h}_u^2 \mathbf{h}_v^1}$$

投影到2D平面

滤波器

$$\hat{\mathcal{G}}(\mathbf{x}) = \max\left\{\mathcal{G}(\mathbf{u}(\mathbf{x})), \mathcal{G}\left(\frac{\mathbf{x} - \mathbf{c}}{\sigma}\right)\right\}$$

$\frac{\sqrt{2}}{2}$

光栅化:

$$\mathbf{c}(\mathbf{x}) = \sum_{i=1} \mathbf{c}_i \, \alpha_i \, \hat{\mathcal{G}}_i(\mathbf{u}(\mathbf{x})) \prod_{j=1}^{i-1} (1 - \alpha_j \, \hat{\mathcal{G}}_j(\mathbf{u}(\mathbf{x})))$$

颜色参数 不透明度

$$\mathcal{L} = \mathcal{L}_c + \alpha \mathcal{L}_d + \beta \mathcal{L}_n$$

$$L_{\text{color}} = (1-\lambda) \cdot |\hat{I} - I|_1 + \lambda(1 - \text{SSIM}(\hat{I}, I))$$

$$\mathcal{L}_d = \sum_{i,j} \omega_i \omega_j |z_i - z_j|$$

$$\mathcal{L}_n = \sum_i \omega_i (1 - \mathbf{n}_i^\top \mathbf{N})$$

$$f^{3D}(p) = \text{sigmoid}(o) \exp\left(-\frac{1}{2}(p-\mu)^T \Sigma^{-1} (p-\mu)\right),$$

透明度 opacity

高斯椭球中心

将3D高斯投影到2D

$$\left\{ \begin{aligned} \mu^{2D} &= K\big((E\mu)/(E\mu)_z\big), \\ \Sigma^{2D} &= JE\Sigma E^T J^T, \end{aligned} \right.$$

内参 外参矩阵

为 μ^{2D} 的雅可比 $\dfrac{\partial \mu^{2D}}{\partial \mu}$.

协方差矩阵的计算

$$\Sigma = RSS^T R^T$$

尺度参数(x,y,z轴尺度)

高斯球的旋转

$$C_{\text{pix}} = \sum_{i \in V} c_i f_{i,\text{pix}}^{2D} \prod_{j=1}^{i-1} (1 - f_{j,\text{pix}}^{2D})$$

2D高斯(对应 μ^{2D} 与 Σ^{2D})

可理解为final透明度

表征颜色(RGB)

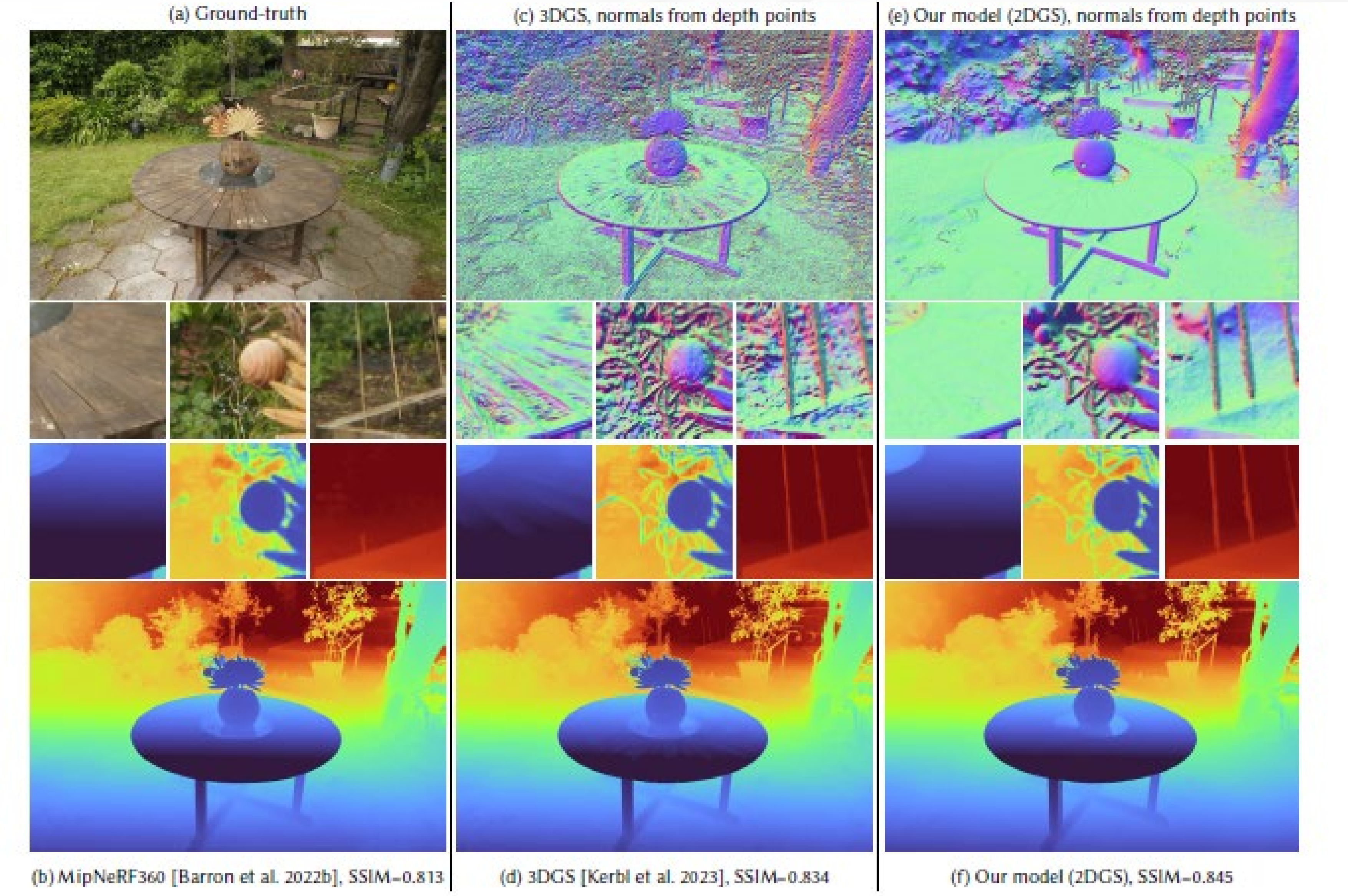
spherical harmonics parameters

某个pixel的颜色

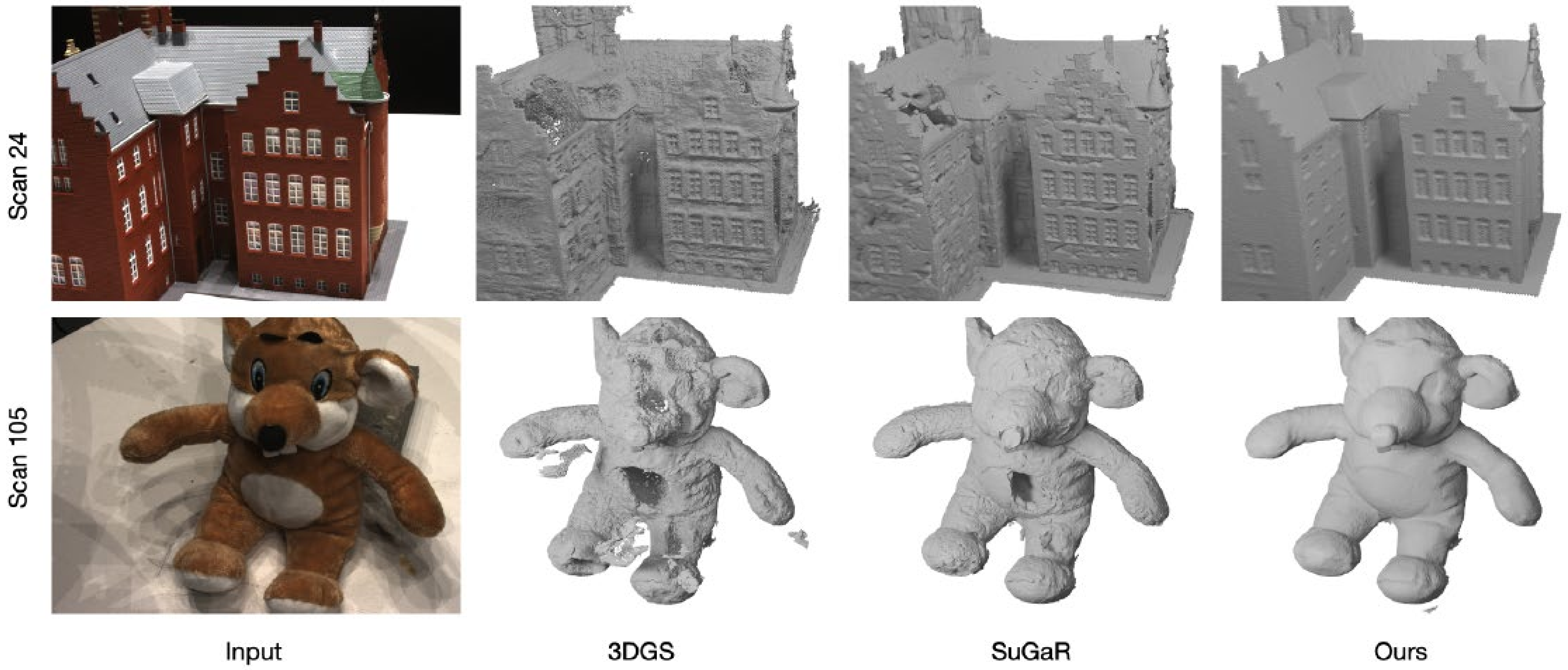
会影响到这个pixel的所有3D高斯集合

$$L_{\text{color}} = (1-\lambda) \cdot |\hat{I} - I|_1 + \lambda(1 - \text{SSIM}(\hat{I}, I))$$

2DGS vs 3DGS

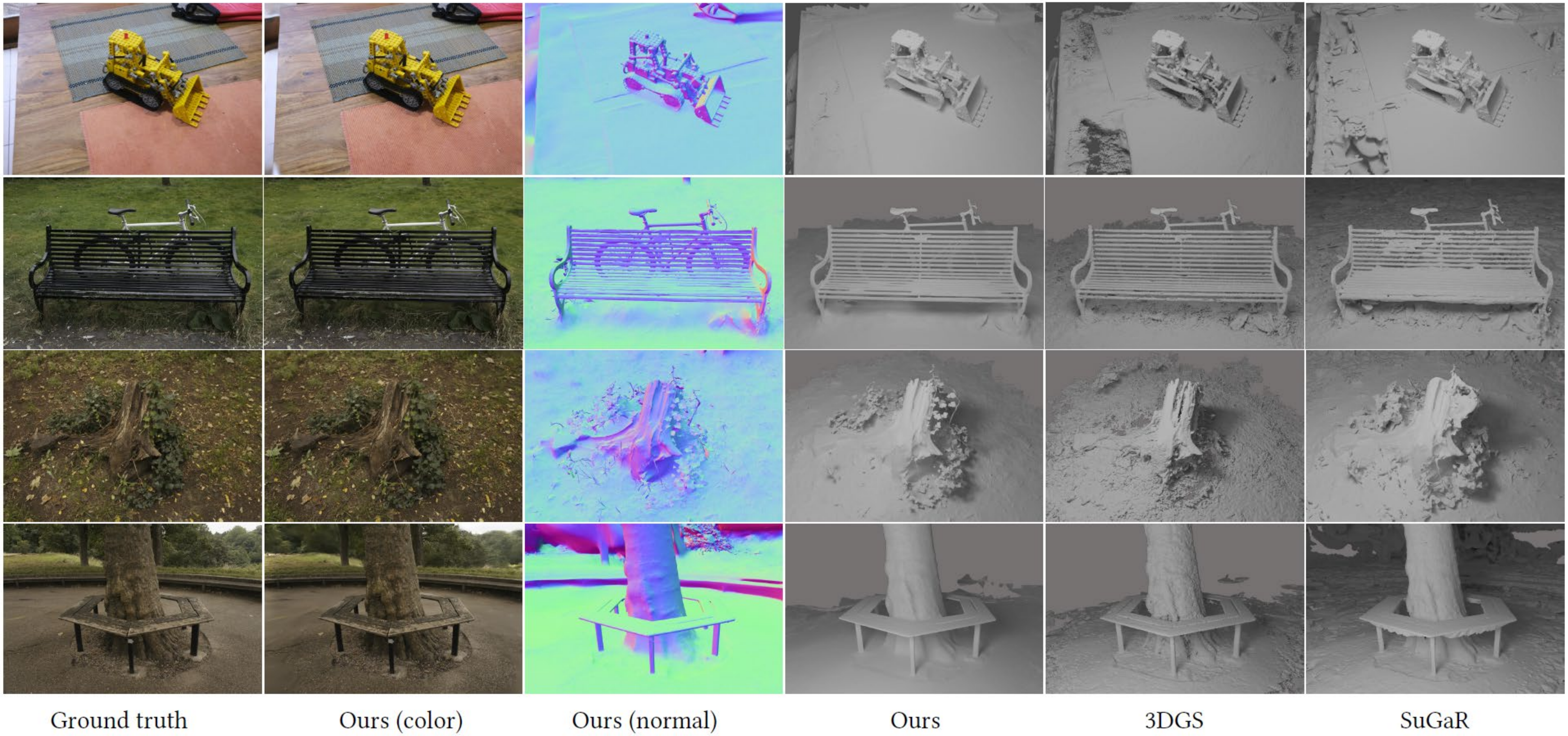

(a) Ground-truth
(c) 3DGS, normals from depth points
(e) Our model (2DGS), normals from depth points
(b) MipNeRF360 [Barron et al. 2022b], SSIM=0.813
(d) 3DGS [Kerbl et al. 2023], SSIM=0.834
(f) Our model (2DGS), SSIM=0.845

2DGS vs 3DGS



2DGS vs 3DGS

香 港 大 學
THE UNIVERSITY OF HONG KONG



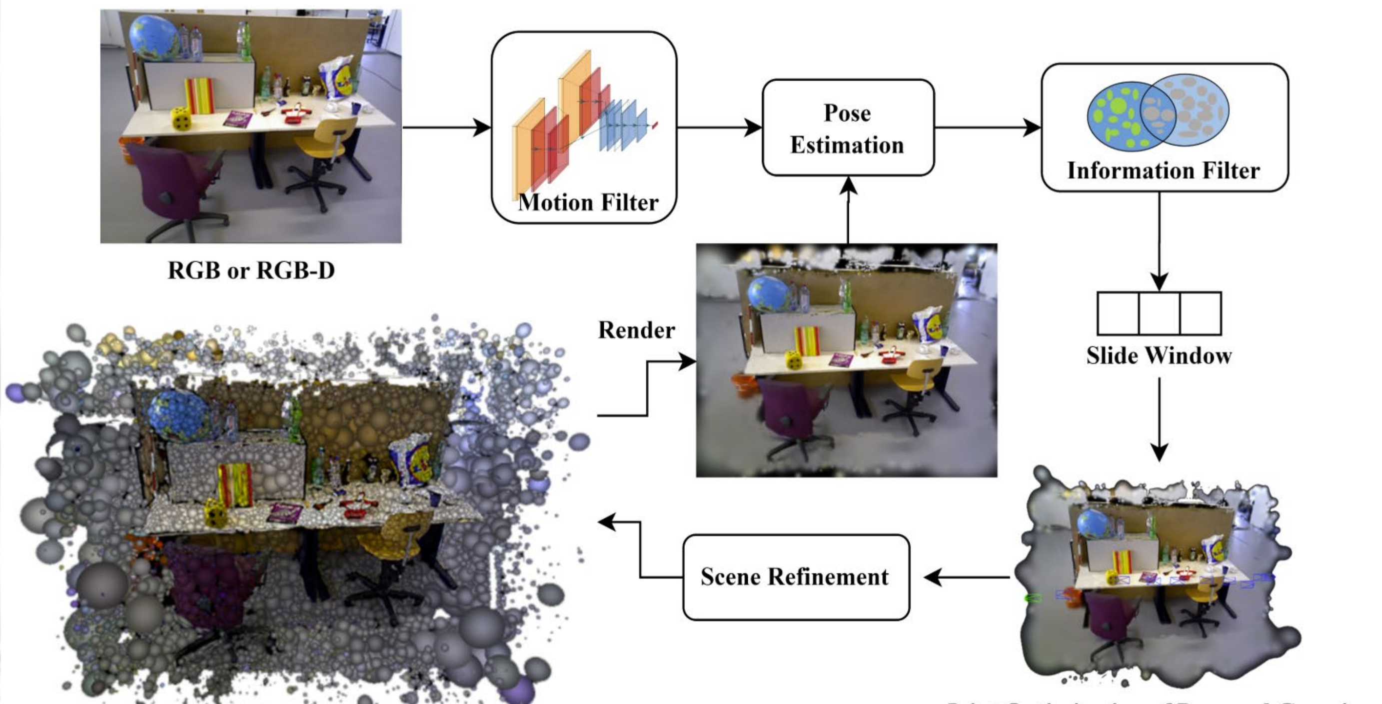
Fast-LIVO2


Application-3: Gaussian Splatting
FAST-LIVO2 + 3DGS
Colmap+ 3DGS
21s (point mapping) 15min:30s (training)
9h (point mapping) 10min:59s (training)
FAST-LIVO2 + 3DGS combination is even better!
PSNR 29.98
PSNR 29.12

MotionGS : Compact Gaussian Splatting SLAM by Motion Filter

香 港 大 學
THE UNIVERSITY OF HONG KONG

- A fusion of **deep visual feature**, dual keyframe selection and 3DGS;
- Pose tracking is achieved by feature extraction and direct pose optimization on each frame;
- Motion filter performs feature extraction on each frame and only retains frames that exceed the threshold (Like DROID-SLAM);
- The coarse-to-fine pose estimation and **compact** Gaussian scene representation are implemented by **dual keyfeature selection** and novel **loss functions**;

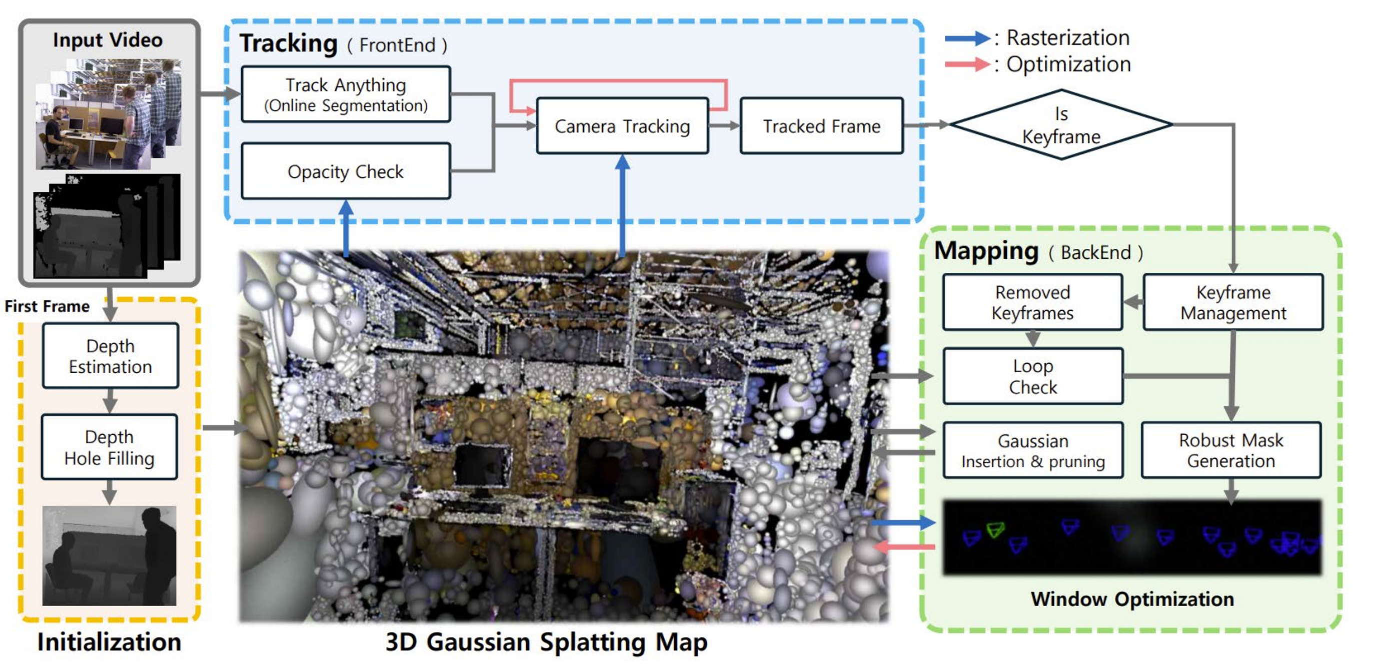


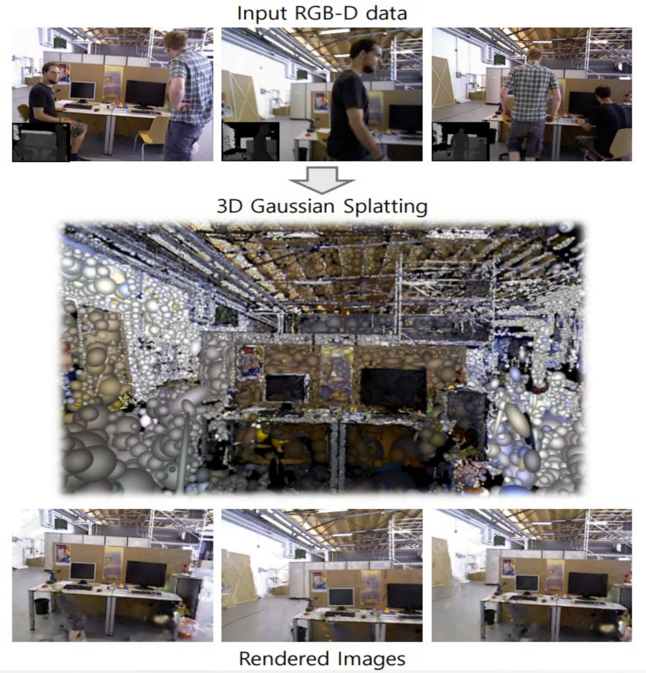
Class	Method	fr1	fr2	fr3	Arg
Mono	ORB-SLAM2	2.00	0.60	2.30	1.60
	DROID-SLAM	1.80	0.50	2.80	1.70
	MonoGS	4.15	4.79	4.39	4.44
	Ours	**3.53**	**3.93**	**2.43**	**3.30**
RGB-D(NeRF)	iMAP	4.90	2.00	5.80	4.23
	NICE-SLAM	4.26	6.19	6.87	5.77
	ESLAM	2.47	1.11	2.42	2.00
	Vox-Fusion	3.52	1.49	26.01	10.34
	Co-SLAM	2.40	1.70	2.40	2.17
	Point-SLAM	4.34	1.31	3.48	3.04
RGB-D(3DGS)	MonoGS	1.52	1.58	1.65	1.58
	SplaTAM	3.35	1.24	5.16	3.25
	GS-SLAM	3.30	1.32	6.60	3.70
	Ours	**1.47**	**1.38**	**1.41**	**1.46**

Table 1: Comparison of tracking results ATE (cm) on TUM. The red, blue, and green in the above table represent the first, second, and third, respectively.

DGS-SLAM: Gaussian Splatting SLAM in Dynamic Environment

- The first dynamic SLAM framework built on 3DGS;
- Developing based on the Gaussian Splatting SLAM (aka. The **MonoGS**) with a robust **filtering** process to handle dynamic objects;
- Introducing a **mask** generation method that enforces photometric consistency across keyframes, reducing noise from inaccurate segmentation and artifacts such as shadows;
- Proposing loop-aware window selection mechanism, which utilizes unique keyframe IDs of 3D Gaussians for loop detection;





Performance of removing dynamic objects

- Efficiently removing the dynamic object (through the online segmentation with robust mask);
- For the segmentation mask, the authors leverage Track Anything, an open-vocabulary video segmentation module that operates online;
- The loop-aware keyframe insertion is just for consistency of the global Gaussian map;

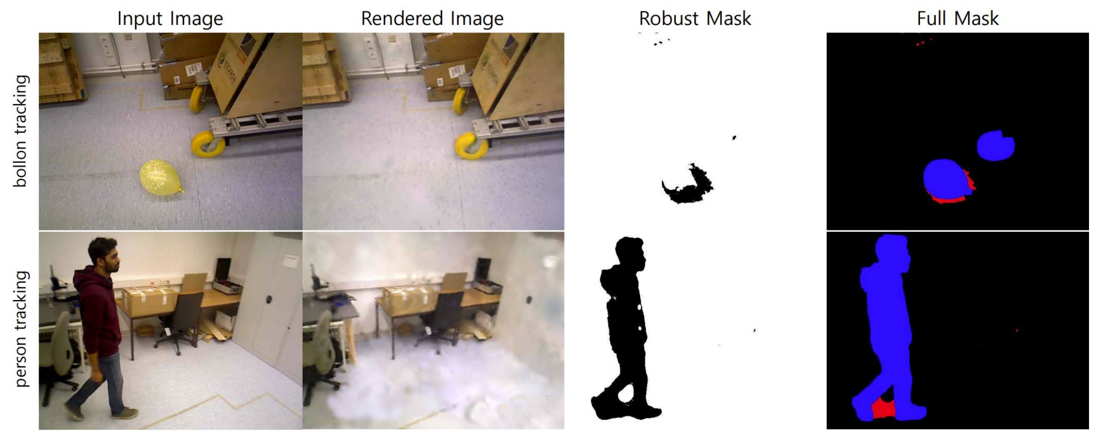
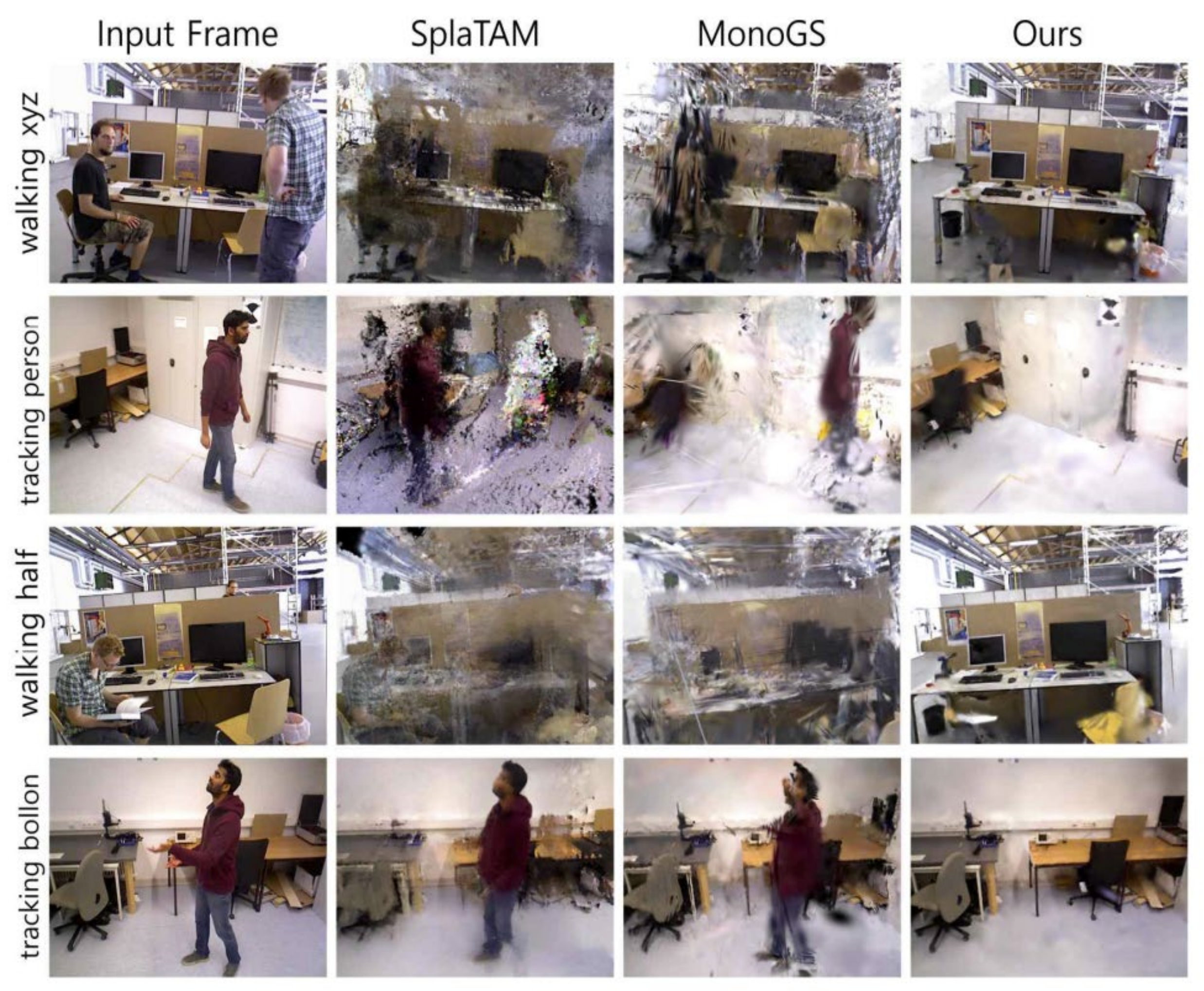


Fig. 4. Visualization of robust mask generation. From right to left: the input image, rendered image, robust mask, and full mask. In the full mask, blue represents the semantic segmentation mask, and red indicates the robust mask.



Tracking Performance in Dynamic Environment

香 港 大 學
THE UNIVERSITY OF HONG KONG

- The tracking pose is optimized by minimizing the difference between each input frame and the rendered result;
- The baseline (MonoGS) and SplaTAM often fail in dynamic scene;
- Bonn dataset is more complex and captured in larger scenes with various dynamic movements;
- The time analysis requiring excluding the time spent on semantic segmentation;

CAMERA TRACKING RESULTS ON DYNAMIC AND STATIC SCENES IN THE *TUM RGB-D* DATASET. THE UNITS FOR ATE AND S.D ARE IN CM.

Methods	Dense	Dynamic								Static				Avg.	
		f3/wk_xyz		f3/wk_hf		f3/wk_st		f3/st_hf		f1/xyz		f1/rpy			
Traditional SLAM methods		*ATE*	*S.D.*	*ATE*	*S.D.*	*ATE*	*S.D.*	*ATE*	*S.D.*	*ATE*	*S.D.*	*ATE*	*S.D.*	*ATE*	*S.D.*
ORB-SLAM3 [44]		28.1	12.2	30.5	9.0	2.0	1.1	**2.6**	1.6	**1.1**	0.6	2.2	1.3	11.1	4.3
DVO-SLAM [?]	✓	59.7	-	52.9	-	21.2	-	6.2	-	**1.1**	-	**2.0**	-	22.9	-
DynaSLAM [15]		**1.7**	-	**2.6**	-	**0.7**	-	2.8	-	-	-	-	-	**2.0**	-
ReFusion [5]	✓	9.9	-	10.4	-	1.7	-	11.0	-	-	-	-	-	8.3	-
Radiance-Field SLAM methods		*ATE*	*S.D.*	*ATE*	*S.D.*	*ATE*	*S.D.*	*ATE*	*S.D.*	*ATE*	*S.D.*	*ATE*	*S.D.*	*ATE*	*S.D.*
NICE-SLAM [23]	✓	113.8	42.9	X	X	88.2	27.8	45.0	14.4	4.6	3.8	3.4	2.5	51	18.3
Vox-Fusion [24]	✓	146.6	32.1	X	X	109.9	25.5	89.1	28.5	1.8	0.9	4.3	3.0	70.4	18
Co-SLAM [26]	✓	51.8	25.3	105.1	42.0	49.5	10.8	4.7	2.2	2.3	1.2	3.9	2.8	36.3	14.1
ESLAM [25]	✓	45.7	28.5	60.8	27.9	93.6	20.7	3.6	1.6	1.1	0.6	**2.2**	**1.2**	34.5	13.5
SplaTAM [9]	✓	134.4	32.1	746.1	250.5	97.8	26.9	14.1	6.8	**1.0**	0.5	2.6	1.3	166.0	52.9
MonoGS [10]	✓	73.4	20.1	65.6	24.8	5.5	3.0	2.7	**1.5**	**1.0**	**0.4**	2.5	1.3	37.7	25.1
GS-ICP SLAM [11]	✓	70.5	45.1	73.9	34.1	98.2	24.1	9.9	3.7	1.4	0.7	3.2	2.8	42.9	18.4
RoDyn-SLAM [45]	✓	8.3	5.5	5.6	**2.8**	1.7	0.9	4.4	2.2	1.5	0.8	2.8	1.5	4.1	2.3
DGS-SLAM **(ours)**	✓	**4.1**	**2.2**	**5.5**	**2.8**	**0.6**	**0.2**	4.1	1.6	1.2	0.6	2.4	1.3	**3.0**	**1.5**

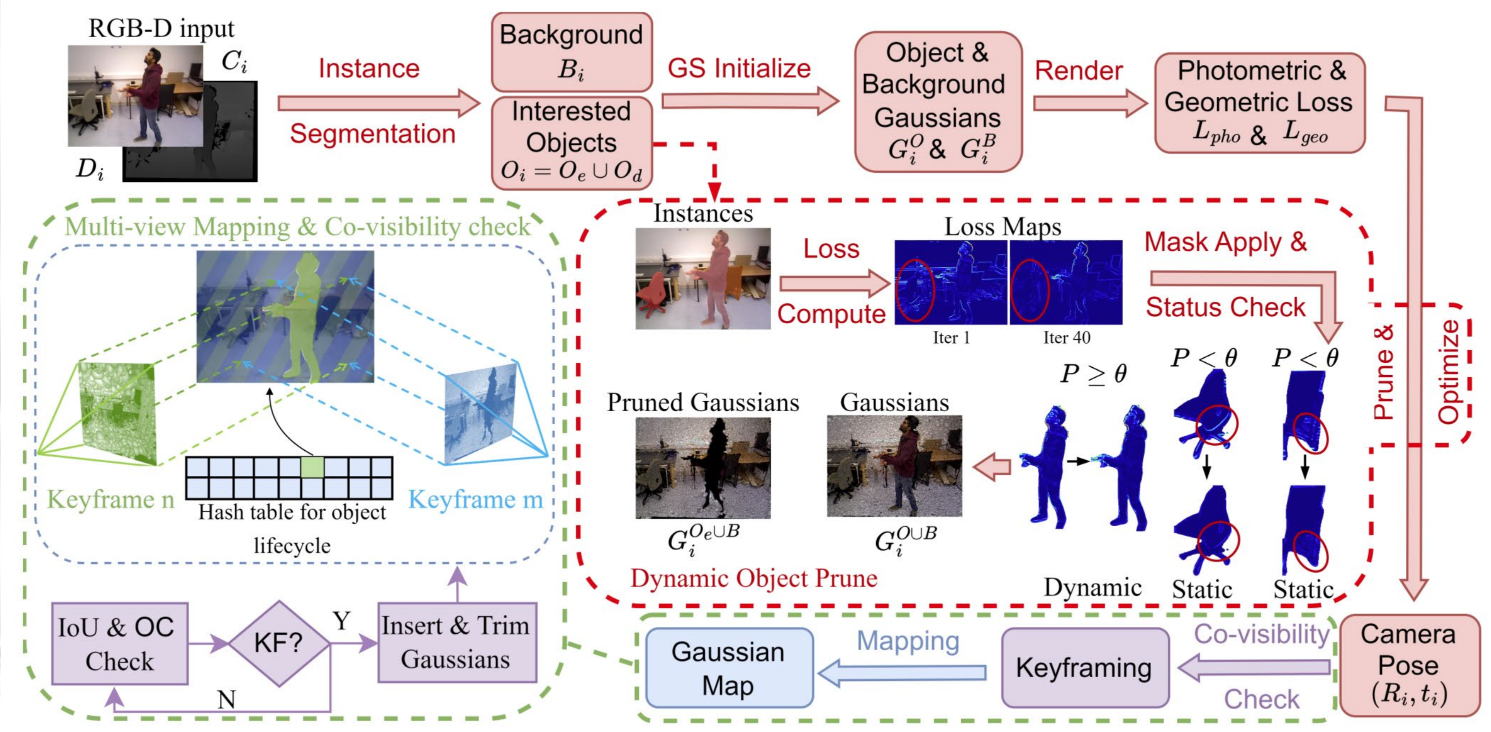
CAMERA TRACKING RESULTS ON DYNAMIC SCENES IN THE *BONN* *RGB-D* DATASET. THE UNITS FOR ATE AND S.D ARE IN CENTIMETERS (CM).

Methods	Dense	balloon		balloon2		ps_track		ps_track2		ball_track		mv_box2		Avg.	
Traditional SLAM methods		*ATE*	*S.D.*	*ATE*	*S.D.*	*ATE*	*S.D.*	*ATE*	*S.D.*	*ATE*	*S.D.*	*ATE*	*S.D.*	*ATE*	*S.D.*
ORB-SLAM3 [44]		5.8	2.8	17.7	8.6	70.7	32.6	77.9	43.8	**3.1**	1.6	**3.5**	1.5	29.8	15.2
Droid-VO [46]	✓	5.4	-	4.6	-	21.34	-	46.0	-	8.9	-	5.9	-	15.4	-
DynaSLAM [15]		**3.0**	-	**2.9**	-	**6.1**	-	**7.8**	-	4.9	-	3.9	-	**4.8**	-
ReFusion [5]	✓	17.5	-	25.4	-	28.9	-	46.3	-	30.2	-	17.9	-	27.7	-
Radiance-Field SLAM methods		*ATE*	*S.D.*	*ATE*	*S.D.*	*ATE*	*S.D.*	*ATE*	*S.D.*	*ATE*	*S.D.*	*ATE*	*S.D.*	*ATE*	*S.D.*
NICE-SLAM [23]	✓	X	X	66.8	20.0	54.9	27.5	45.3	17.5	21.2	13.1	31.9	13.6	44.1	18.4
Vox-Fusion [24]	✓	65.7	30.9	82.1	52.0	128.6	52.5	162.2	46.2	43.9	16.5	47.5	19.5	88.4	36.3
Co-SLAM [26]	✓	28.8	9.6	20.6	8.1	61.0	22.2	59.1	24.0	38.3	17.4	70.0	25.5	46.3	17.8
ESLAM [25]	✓	22.6	12.2	36.2	19.9	48.0	18.7	51.4	23.2	12.4	6.6	17.7	7.5	31.4	14.7
SplaTAM [9]	✓	35.7	14.1	36.4	17.4	124.8	36.5	163.0	51.3	12.8	16.8	17.9	9.3	65.1	24.2
MonoGS [10]	✓	33.2	16.4	26.5	14.2	63.2	29.0	47.2	15.4	4.3	2.2	22.9	12.4	32.9	14.2
GS-ICP SLAM [11]	✓	43.8	16.0	42.1	19.1	92.8	42.3	44.7	20.3	27.9	17.4	24.8	11.5	31.3	14.2
RoDyn-SLAM [45]	✓	7.9	2.7	11.5	6.1	14.5	4.6	13.8	**3.5**	13.3	4.7	12.6	4.7	12.3	4.4
DGS-SLAM **(Ours)**	✓	**2.9**	**0.8**	**6.0**	**2.8**	**9.8**	**4.1**	**11.1**	3.9	**5.6**	**2.8**	**8.8**	**3.8**	**7.3**	**3.0**

Gassidy: Gaussian Splatting SLAM in Dynamic Environments

香 港 大 學
THE UNIVERSITY OF HONG KONG

- Designing photometric geometric loss function, also based on **MonoGS with the YOLO** segmentation;
- To distinguish and filter environmental disturbances, the authors iteratively **analyze rendering loss flows** to detect features characterized by changes in loss values between dynamic objects and static components;
- One loss design three contributions (just a good writer 😂);



$$L^{O(j)}_{pho} = \frac{1}{a_j}\sum_{p\in O(j)} \left(|\hat{I}_p - I_p| \circ S_{O(j)}\right),$$

$$L^{B}_{pho} = \frac{1}{b}\sum_{p\in B} \left(|\hat{I}_p - I_p| \circ \neg \bigcup_{O(j)\in O} S_{O(j)}\right),$$

Utilizing both errors for tracking and mapping at the beginning, based on the loss difference, filtering out the dynamic object.
(For misleading reader)

$$L = \lambda_a L_{pho} + (1-\lambda_a) L_{geo},$$

photometric geometric loss function

The loss for the background and static objects decreases consistently over iterations as they become well-aligned with the scene geometry. In contrast, dynamic objects exhibit higher and more fluctuating loss values across iterations due to their motion;
Simply speaker: YOLO only segment objects and the author used complex ways for further filtering;

Mapping and Tracking Performance

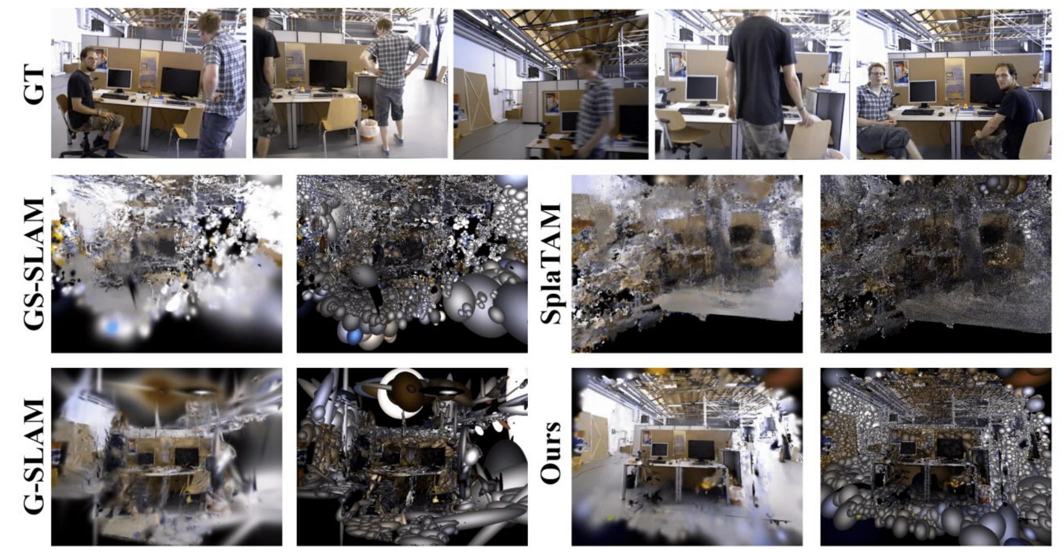
香 港 大 學
THE UNIVERSITY OF HONG KONG

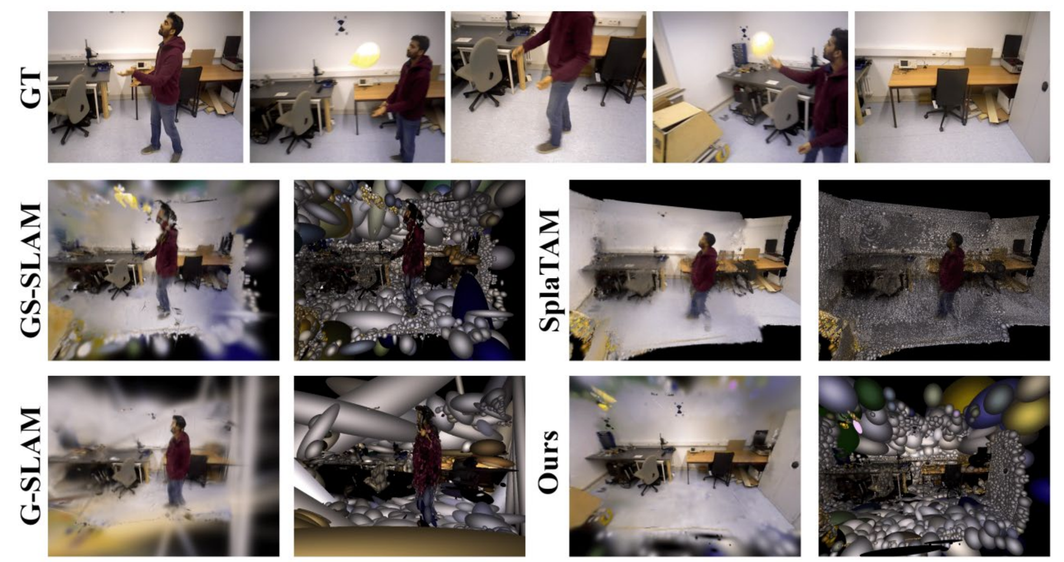
TABLE I: Camera tracking results on dynamic scenes from the **TUM RGB-D** dataset. The best results within each domain are highlighted in **bold**, and the best results among all domains are marked with underline. D/S indicates whether the method is dense or sparse reconstruction, "ATE" column shows the RMSE of the ATE, and the "Std." column presents the standard deviation of ATE. X means tracking failure, and – indicates not mentioned in the original report.

Methods	Type	*f3/wk_xyz*		*f3/wk_hf*		*f3/wk_st*		*f3/st_hf*		Avg.	
Keypoint-based SLAM methods	*D/S*	ATE	Std.	ATE	Std.	ATE	Std.	ATE	Std.	ATE	Std.
ORB-SLAM3 [19]	S	28.1	12.2	30.5	9.0	2.0	1.1	**2.6**	1.6	15.8	6.0
DynaSLAM [3]	S	**1.7**	–	**2.6**	–	**0.7**	–	2.8	–	**2.0**	–
NeRF-based SLAM methods	*D/S*	ATE	Std.	ATE	Std.	ATE	Std.	ATE	Std.	ATE	Std.
NICE-SLAM [8]	D	113.8	42.9	X	X	137.3	21.7	93.3	35.3	114.8	33.3
RoDyn-SLAM [9]	D	**8.3**	**5.5**	**5.6**	**2.8**	**1.7**	**0.9**	**4.4**	**2.2**	**4.1**	**2.3**
3DGS-based SLAM methods	*D/S*	ATE	Std.	ATE	Std.	ATE	Std.	ATE	Std.	ATE	Std.
GS-SLAM [15]	D	37.2	9.9	60.0	20.7	8.4	4.1	7.4	5.4	28.5	10.0
SplaTAM [16]	D	149.2	37.4	157.8	54.4	85.3	16.1	14.0	6.8	125.6	109.6
Gaussian-SLAM [20]	D	133.7	54.8	80.7	31.6	19.1	5.2	5.4	2.2	59.7	23.5
GassiDy (Ours)	D	**3.5**	**1.6**	**3.7**	**1.9**	**0.6**	**0.3**	**2.4**	**1.4**	**2.6**	**1.3**

TABLE II: Camera tracking results on dynamic scenes from the **BONN Dynamic RGB-D** dataset. The notation is identical to Table I.

Methods	Type	*balloon*		*balloon2*		*ps_track*		*ps_track2*		*mv_box2*		Avg.	
Keypoint-based SLAM methods	*D/S*	ATE	Std.	ATE	Std.	ATE	Std.	ATE	Std.	ATE	Std.	ATE	Std.
ORB-SLAM3 [19]	S	5.8	2.8	17.7	8.6	70.7	32.6	77.9	43.8	**3.1**	1.6	35.0	17.9
DynaSLAM [3]	S	**3.0**	–	**2.9**	–	**6.1**	–	**7.8**	–	3.9	–	**4.74**	–
NeRF-based SLAM methods	*D/S*	ATE	Std.	ATE	Std.	ATE	Std.	ATE	Std.	ATE	Std.	ATE	Std.
NICE-SLAM [8]	D	X	X	66.8	20.0	54.9	27.5	45.3	17.5	31.9	13.6	49.7	19.7
RoDyn-SLAM [9]	D	**7.9**	**2.7**	**11.5**	**6.1**	**14.5**	**4.6**	**13.8**	**3.5**	**12.6**	**4.7**	**12.1**	**4.32**
3DGS-based SLAM methods	*D/S*	ATE	Std.	ATE	Std.	ATE	Std.	ATE	Std.	ATE	Std.	ATE	Std.
GS-SLAM [15]	D	39.5	19.3	35.6	19.5	93.3	36.3	51.2	19.1	6.1	4.5	45.1	19.7
SplaTAM [16]	D	35.8	14.3	38.7	15.0	138.4	48.1	126.3	36.7	22.0	12.3	54.6	33.7
Gaussian-SLAM [20]	D	65.2	25.5	34.8	22.1	109.2	58.9	118.7	57.2	31.7	20.9	71.9	36.9
GassiDy (Ours)	D	**2.6**	**0.8**	**7.6**	**3.4**	**10.3**	**4.4**	**13.0**	**4.8**	**5.4**	**1.9**	**7.8**	**3.1**

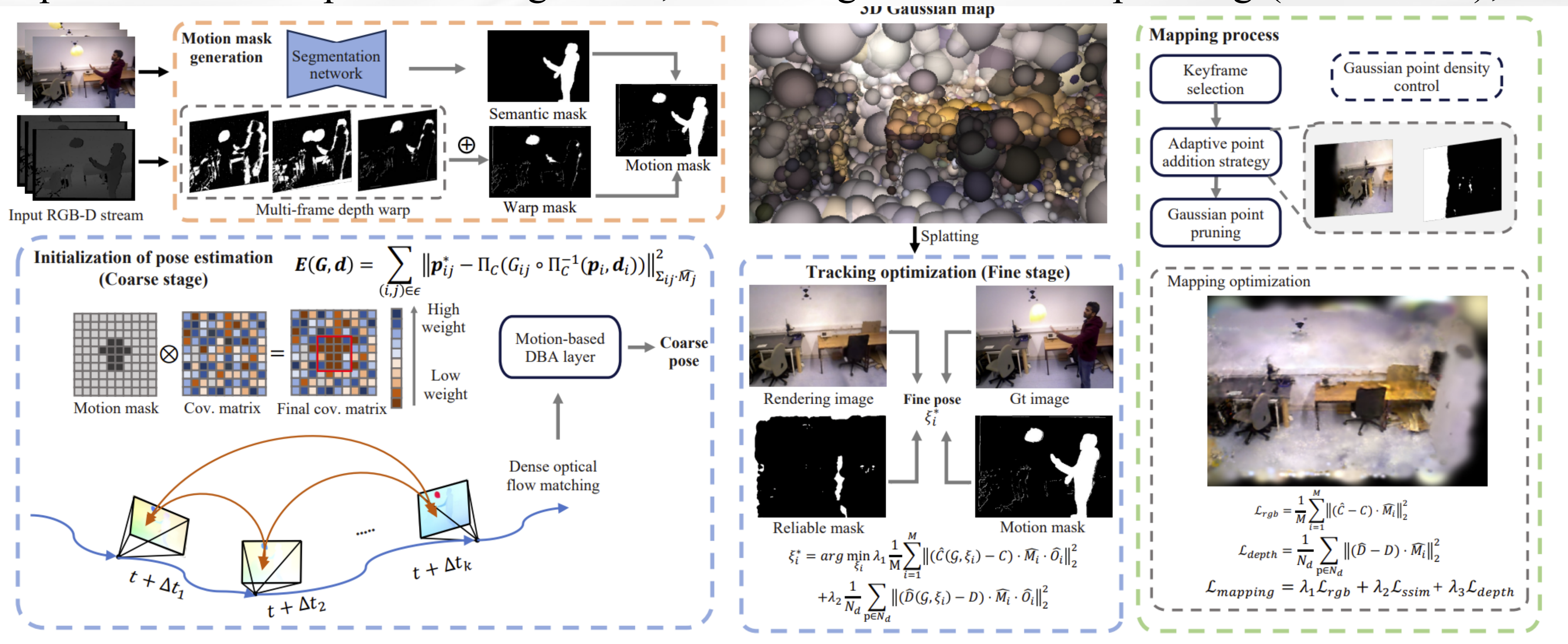




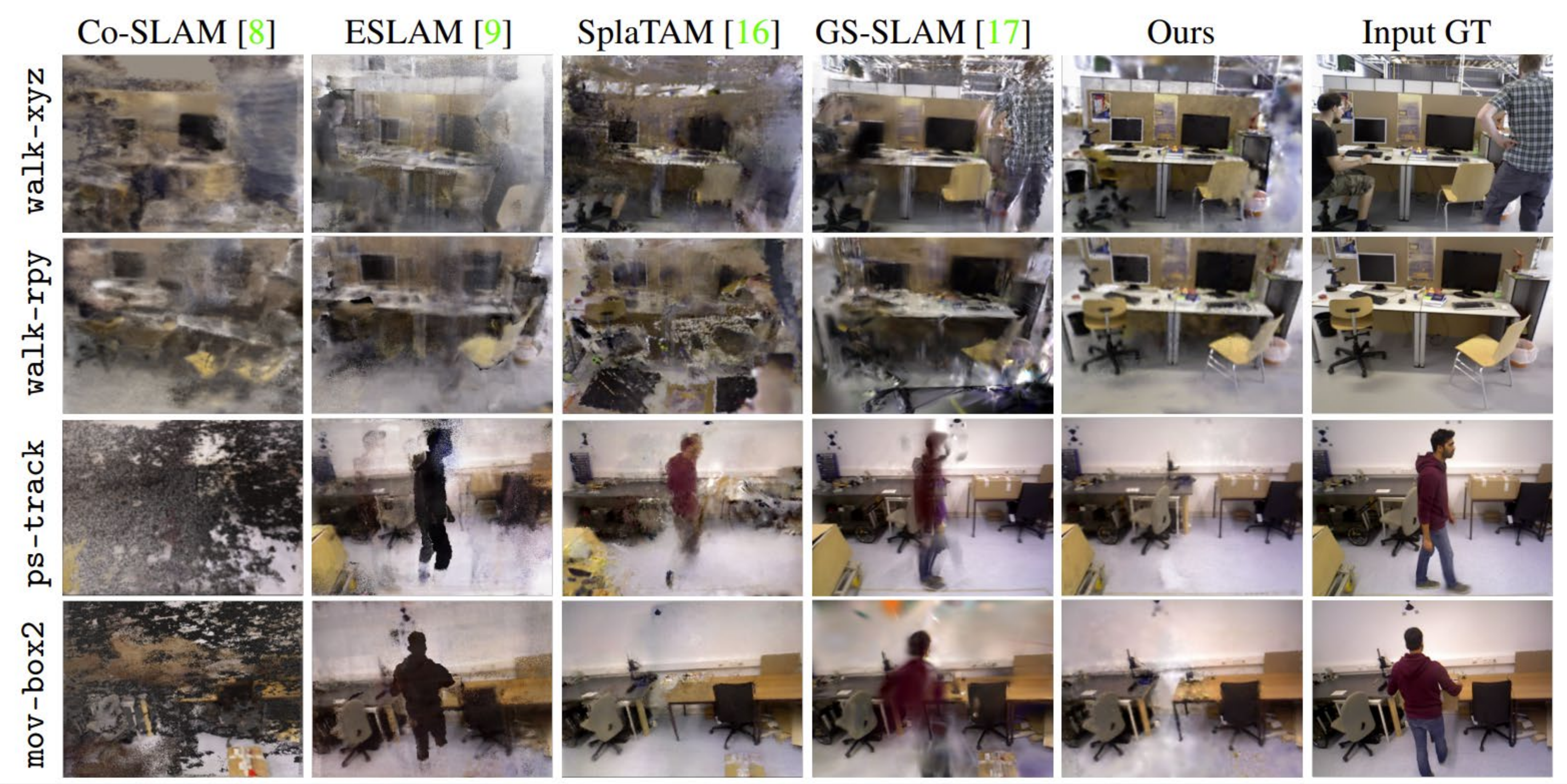
DG-SLAM: Robust Dynamic Gaussian Splatting SLAM with Hybrid Pose Optimization

香 港 大 學
THE UNIVERSITY OF HONG KONG

- Dynamic 3DGS-SLAM with motion mask generation (modified based on a CVPR23 work, considering the warp depth mask);
- Hybrid camera tracking algorithm: Droid-SLAM to provide initial pose estimation and devise coarse-to-fine optimization for pose tracking;
- Adaptive Gaussian point management, including addition and pruning (unconcern);



The Mapping Performance



The Tracking Performance

香 港 大 學
THE UNIVERSITY OF HONG KONG

- The Improvement of the Tracking is not obvious when comparison the DGS-SLAM and Gassidy, consider the proposed hybrid pose tracking scheme;

Method	f3/w_r	f3/w_x	f3/w_s	f3/s_x	f2/d_p	f3/l_o	Avg.
ORB-SLAM3 [3]	68.7	28.1	2.0	**1.0**	**1.5**	**1.0**	17.1
ReFusion [5]	-	9.9	1.7	4.0	-	-	5.2
Co-fusion [42]	-	69.6	55.1	2.7	-	-	42.5
MID-fusion [43]	-	6.8	2.3	6.2	-	-	5.1
EM-fusion [44]	-	6.6	1.4	3.7	-	-	3.9
iMAP*[6]	139.5	111.5	137.3	23.6	119.0	5.8	89.5
NICE-SLAM[7]	X	113.8	88.2	7.9	X	6.9	54.2
Vox-Fusion[10]	X	146.6	109.9	3.8	X	26.1	71.6
Co-SLAM[8]	52.1	51.8	49.5	6.0	7.6	2.4	28.3
ESLAM[9]	90.4	45.7	93.6	7.6	X	2.5	48.0
Rodyn-SLAM[33]	7.8	8.3	1.7	5.1	5.6	2.8	5.3
SplaTAM[16]	100.4	218.3	115.2	1.7	5.4	5.1	74.4
GS-SLAM[17]	33.5	37.7	8.4	2.7	8.6	1.8	15.5
DROID-VO[19]	10.0	1.7	0.7	1.1	3.7	2.3	3.3
DG-SLAM(Ours)	**4.3**	**1.6**	**0.6**	**1.0**	3.2	2.3	**2.2**

Table 2: **Camera tracking results on several dynamic scene sequences in the *TUM* dataset**. "∗" denotes the version reproduced by NICE-SLAM. "X" and "-" denote the tracking failures and absence of mention, respectively. The metric is Absolute Trajectory Error (ATE) and the unit is [cm].

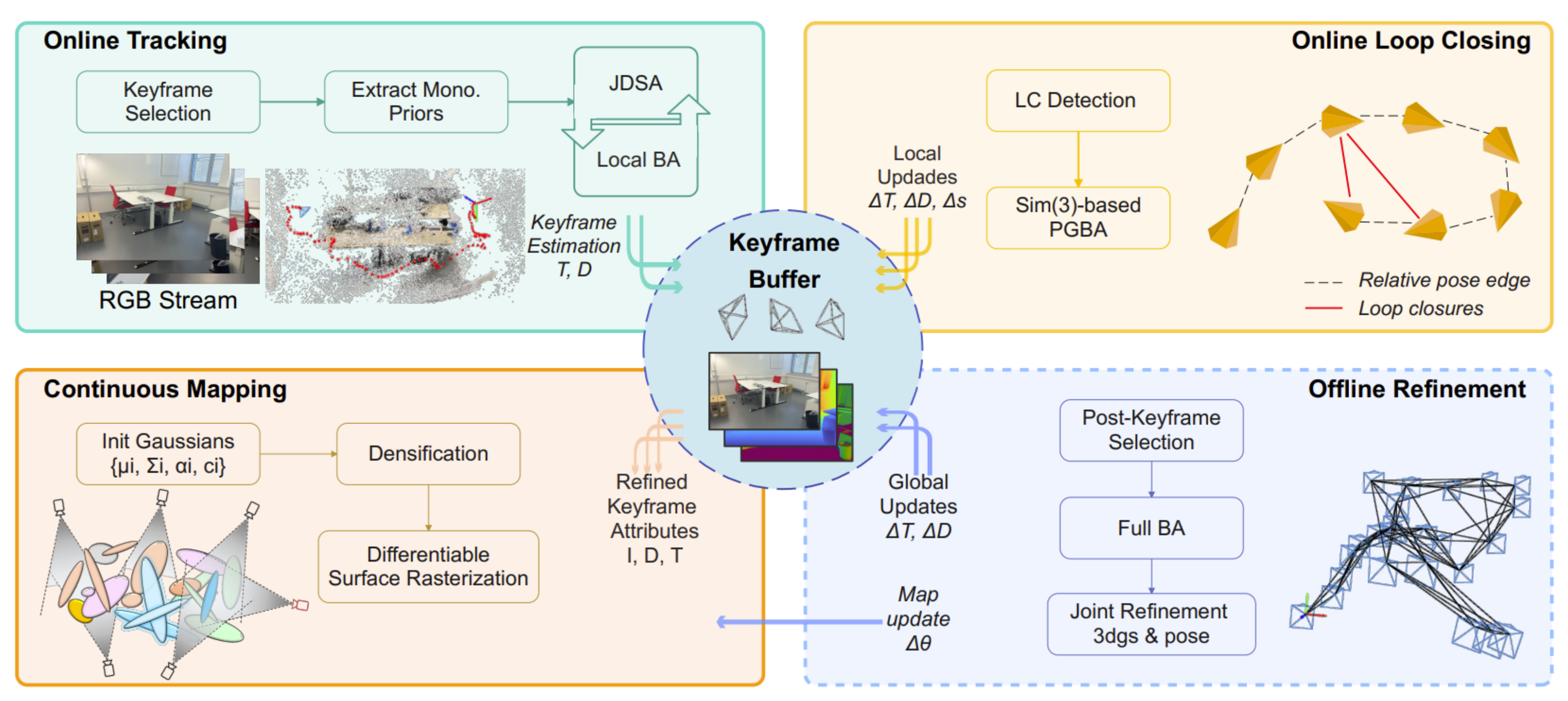
Method	ball	ball2	ps_tk	ps_tk2	ball_tk	mv_box2	Avg.
ORB-SLAM3 [3]	5.8	17.7	70.7	77.9	**3.1**	**3.5**	29.8
ReFusion [5]	17.5	25.4	28.9	46.3	30.2	17.9	27.7
iMAP*[6]	14.9	67.0	28.3	52.8	24.8	28.3	36.1
NICE-SLAM[7]	X	66.8	54.9	45.3	21.2	31.9	44.1
Vox-Fusion[10]	65.7	82.1	128.6	162.2	43.9	47.5	88.4
Co-SLAM[8]	28.8	20.6	61.0	59.1	38.3	70.0	46.3
ESLAM[9]	22.6	36.2	48.0	51.4	12.4	17.7	31.4
Rodyn-SLAM[33]	7.9	11.5	14.5	13.8	13.3	12.6	12.3
SplaTAM[16]	35.5	36.1	149.7	91.2	12.5	19.0	57.4
GS-SLAM[17]	37.5	26.8	46.8	50.4	31.9	4.8	33.1
DROID-VO[19]	5.4	4.6	21.4	46.0	8.9	5.9	15.4
DG-SLAM(Ours)	**3.7**	**4.1**	**4.5**	**6.9**	10.0	**3.5**	**5.5**

Table 3: **Camera tracking results on several dynamic scene sequences in the *BONN* dataset.** "∗" denotes the version reproduced by NICE-SLAM. "X" denotes the tracking failures. The metric is ATE and the unit is [cm].

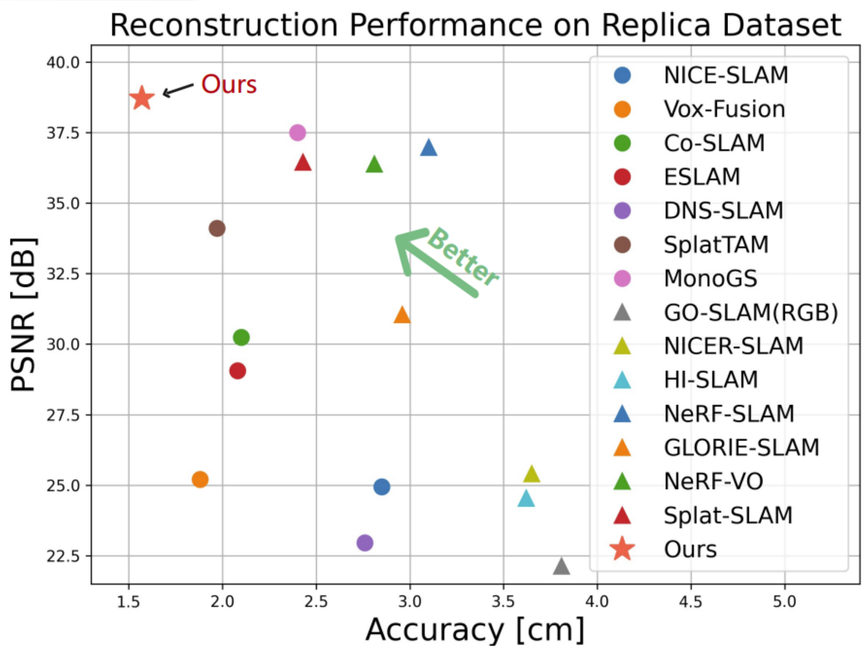
HI-SLAM2: Geometry-Aware Gaussian SLAM for Fast Monocular Scene Reconstruction

香 港 大 學
THE UNIVERSITY OF HONG KONG

- Monocular **RGB learning-based** dense SLAM to generate depth, and then using it for **3DGS** as map representation;
- Enhancing geometry estimation by combining monocular geometry priors with learning-based dense SLAM, while leveraging 3DGS as compact map representation for efficient and accurate scene modeling;
- Adapting loop closure to ensure the global consistency;
- Grid-based scale alignment strategy to maintain the scale consistency of the estimated depth;

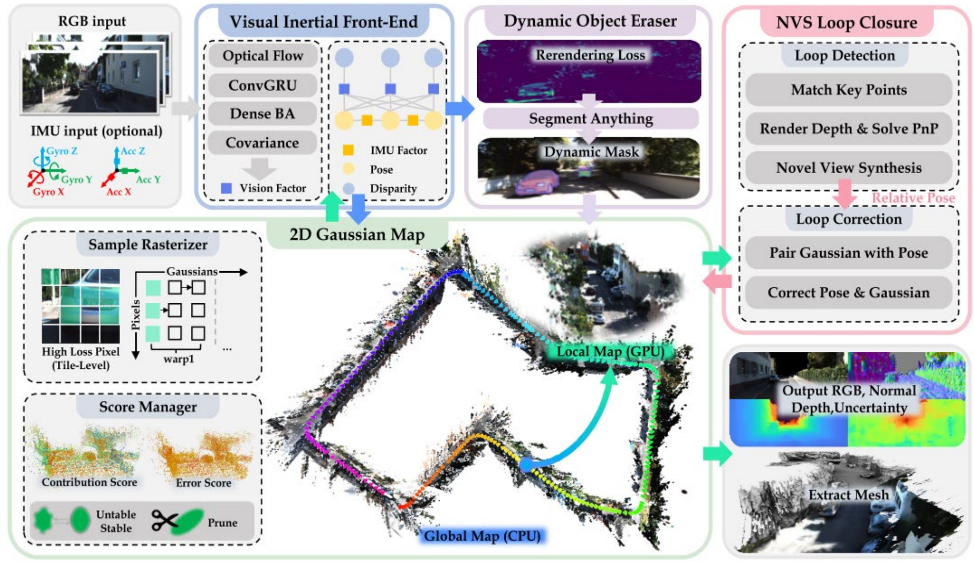


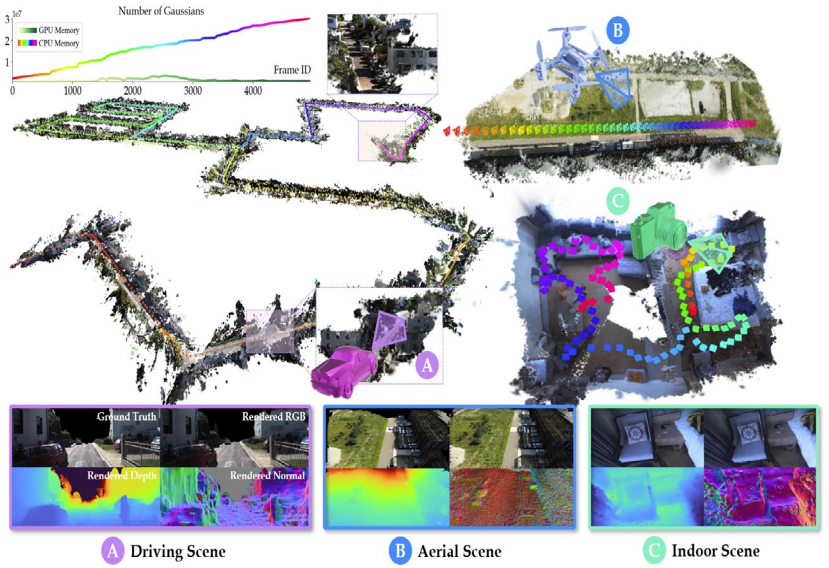
RGB-only methods▲, and RGB-D methods●



VINGS-Mono: Visual-Inertial Gaussian Splatting Monocular SLAM in Large Scenes

- Monocular (inertial) **2D Gaussian Splatting** SLAM framework designed for large scenes, supporting **kilometer-scale large scenes and mobile app**;
- To address storage and optimization efficiency, a **score manager** (contribution and error) is developed to manage (prune) the 2D Gaussian Map by integrating local and global map representations;
- A **sample rasterizer** to significantly accelerate the backpropagation algorithm of Gaussian Splatting;
- A **single-to-multi pose refinement module** (GS-based pose refinement) back-propagates rendering errors from a single frame to optimize the poses of all frames within the frustum's field of view (different keyframe), improving overall pose consistency;
- **Loop Closure module** leverages the Novel View Synthesis (NVS) capabilities of Gaussian Splatting for loop closure detection and correction of the Gaussian map (**simultaneously adjusting millions of Gaussian attributes (actually just position and rotation)** upon detecting a loop);
- **Dynamic Eraser** to address the inevitable presence of dynamic objects;





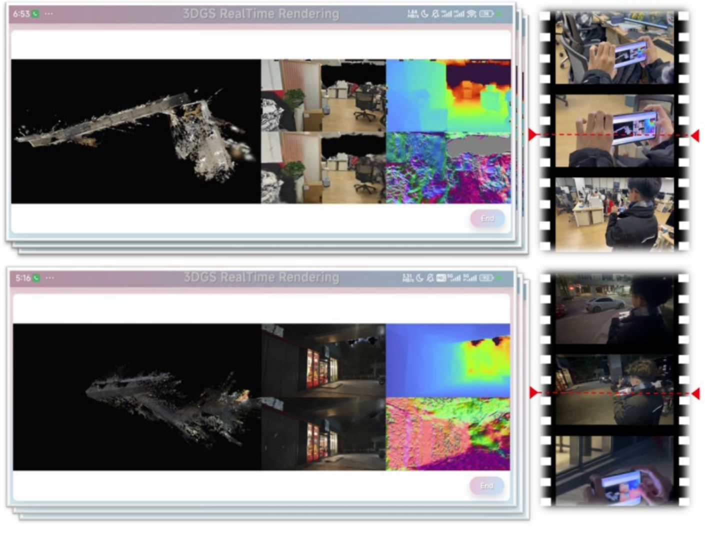


Fig. 13: Mobile App of VINGS-Mono.

VINGS-Mono

香 港 大 學
THE UNIVERSITY OF HONG KONG

- The visual front-end is build based on DBA-Fusion;
- While the mapping is modified from the 2DGS, such as score manager, sample rasterizer, single-to-multi pose refinement;
- Leveraging the novel view synthesis capabilities of GS from new viewpoints to determine if a loop has been detected (based on the “Lightglue”), the loop detection problem is transformed to **whether the newly captured image can serve as a novel viewpoint of the Gaussian Map**;
- Heuristics-guided segmentation method to distinguish masks of dynamic objects, building based on “Fast segment anything” and redesigning the re-rendering loss;

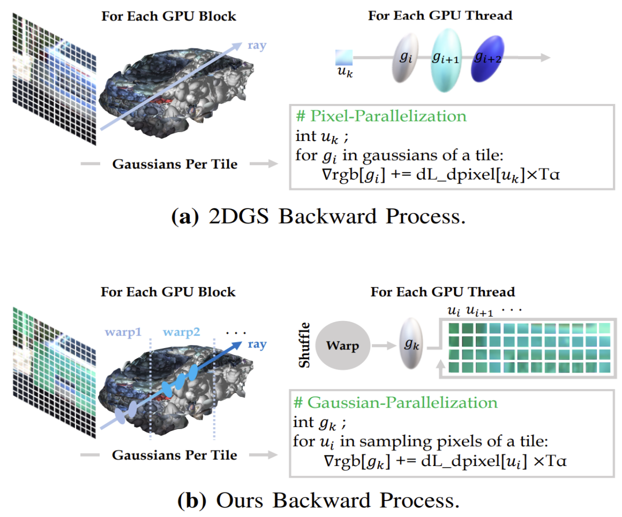


Fig. 3: Sample Rasterizer. In our backpropagation process, each thread is responsible for one Gaussian, and the number of iterations depends on the number of sampled pixels.

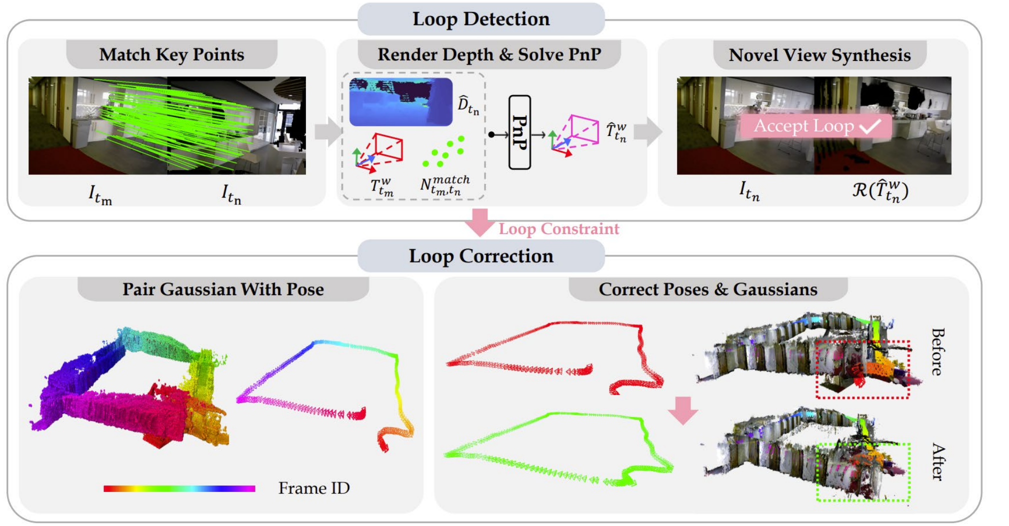


Fig. 4: Pipeline of NVS Loop Closure. We perform feature matching, filtering, and novel view synthesis on keyframes that meet the distance threshold requirements to achieve loop detection. Once a loop is detected, we implement loop correction of the pose and Gaussian map through pairwise Gaussian with pose alignment and graph optimization.

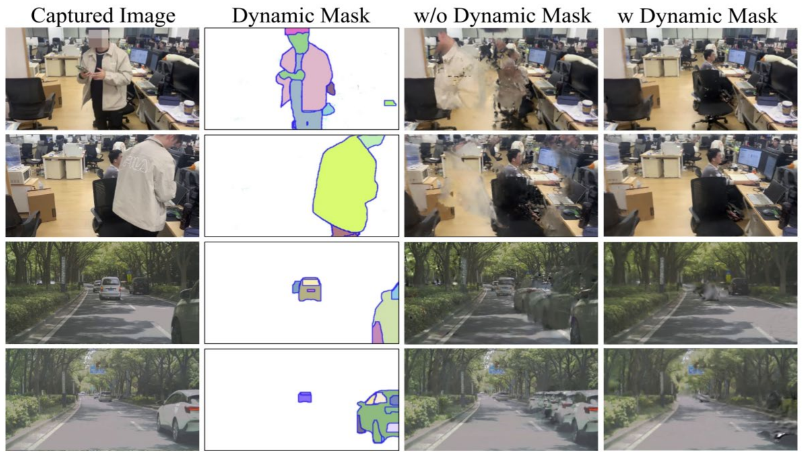


Fig. 5: Effect of Dynamic Object Eraser. Our dynamic eraser can filter out moving people indoors and fast-moving vehicles outdoors, preventing the Gaussian map from being affected by dynamic floaters.

Performance of VINGS-Mono

香 港 大 學
THE UNIVERSITY OF HONG KONG

TABLE I: Monocular Localization results (ATE [cm]) on the indoor datasets ScanNet and BundleFusion. Red, orange, and yellow represent the best, second-best, and third-best performance, respectively. For all evaluation scenarios, the same dataset with ground truth values was used as a reference to compute the average metrics.

ATE (cm) ↓	ScanNet						BundleFusion				
	0054	0059	0106	0169	0233	0465	apt0	apt2	copyroom	office0	office2
ORB-SLAM3	243.26	90.67	178.13	60.15	25.01	181.86	89.38	148.04	19.70	31.41	73.91
DROID-SLAM	161.22	67.26	11.20	17.39	69.85	116.42	87.37	265.64	27.59	116.33	49.32
NeRF-SLAM	147.20	26.95	18.75	13.53	37.23	73.32	85.50	241.72	59.20	59.08	83.57
MonoGS	70.189	97.24	150.89	191.98	62.45	113.19	122.59	142.54	53.41	62.67	127.02
PhotoSLAM	332.03	205.01	359.85	151.61	195.71	294.20	247.19	320.91	54.03	271.87	298.98
Ours	44.08	15.96	16.13	16.84	60.71	92.83	44.22	136.69	39.10	44.44	39.10

TABLE III: Visual inertial localization results (t_{rel} in % and r_{rel} in $^\circ/100m$) on KITTI and KITTI360.

t_{rel}↓ r_{rel}↓	KITTI Sync										KITTI360 Unsync									
	02		06		07		08		09		00		02		05		06		10	
	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}
VINS-Mono	2.08	1.68	4.27	0.32	2.08	0.63	3.22	0.33	4.72	0.65	1.89	0.17	1.01	0.20	1.19	0.22	1.35	0.18	3.61	0.22
ORB-SLAM3	3.51	1.42	4.01	0.94	4.41	0.95	3.36	0.87	4.30	0.89	2.39	0.12	1.31	0.22	1.41	0.23	1.69	0.18	5.34	0.21
Selective-VIO	2.41	0.78	1.90	0.52	1.72	1.01	2.23	0.91	2.83	0.80	-	-	-	-	-	-	-	-	-	-
iSLAM	2.08	0.53	2.40	0.32	2.22	0.47	2.78	0.43	2.51	0.41	7.75	0.36	38.46	0.56	9.36	1.01	32.18	1.46	4.74	0.36
Ours	2.64	0.44	2.01	0.40	1.01	0.80	1.90	0.23	2.84	0.38	0.76	0.10	0.58	0.17	1.16	0.23	0.73	0.16	4.23	0.42

TABLE II: Monocular Localization results (ATE [m]) on the outdoor datasets Waymo and Hierarchical3DGS. "-" indicates that the system failed to track in this scenario, "*" indicates only the first 50 frames were tested due to tracking failure.

RMSE [m] ↓	Waymo			Hierarchical3DGS	
	Scene01	Scene03	Scene14	SmallCity	Campus
ORB-SLAM3	1.21	2.49	2.48	-	-
DROID-SLAM	2.38	2.94	3.98	5.83	1.87
NeRF-SLAM	2.05	5.87	6.43	4.58	1.44
GO-SLAM	3.15	3.07	5.13	5.79	3.50
MonoGS	2.73	10.73	6.59	6.05*	20.81*
PhotoSLAM	3.15	6.41	7.30	47.72*	34.04*
Ours	0.91	2.67	2.27	2.82	1.03

TABLE IV: Localization results on several dynamic scene sequences in the BONN dataset [67].

ATE [cm] ↓	ball	ps_tk	ps_tk2	mv_box2	Avg.
ReFusion	17.5	28.9	46.3	17.9	27.65
RodynSLAM	7.9	14.5	13.8	12.6	12.2
Ours (wo Eraser)	11.75	37.48	48.31	23.44	30.25
Ours (w Eraser)	4.08	4.63	5.05	3.58	4.34

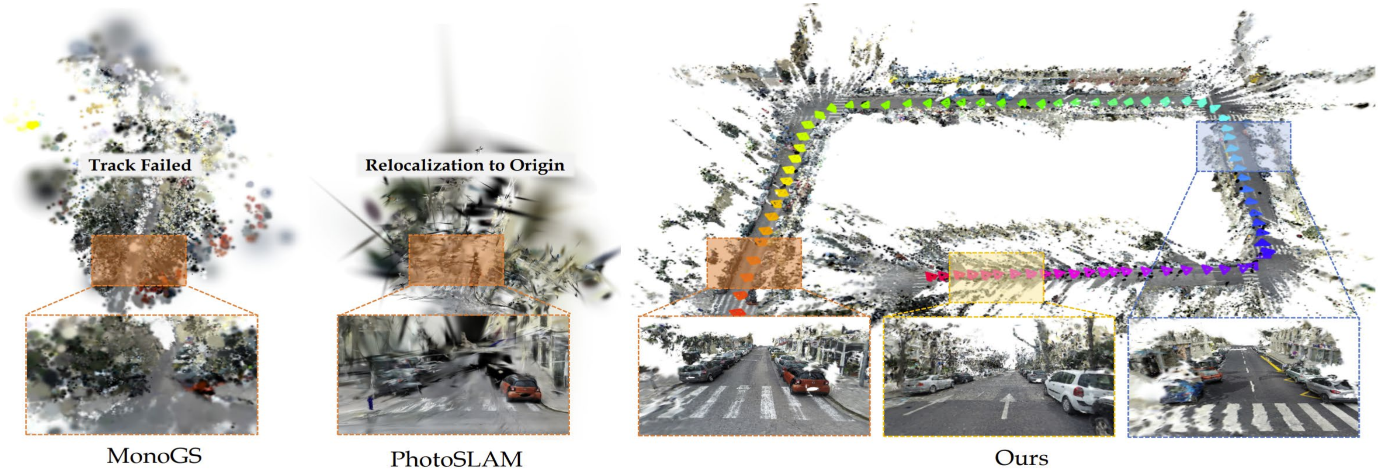


Fig. 6: VO Performance on SmallCity of Hierarchical [60]. MonoGS fails in tracking due to being obscured by large floaters, and Photoslam cannot match feature points to relocate to the starting point due to the lack of complex textures in and ego fast motion. In contrast, our method robustly and stably achieves localization and constructs high-quality Gaussian maps.

Performance of VINGS-Mono

香 港 大 學
THE UNIVERSITY OF HONG KONG

TABLE VI: Quantitative analysis results on the outdoor datasets KITTI, KITTI360, Waymo, Hierarchical, and MegaNeRF. "-" indicates that the system failed to track and render images in the whole scenario.

		KITTI			KITTI360			Waymo			Hierarchical		MegaNeRF	
		02	07	08	05	06	10	01	03	14	SmallCity	Campus	Building	Rubble
GO-SLAM	SSIM↑	0.39	0.46	0.51	0.44	0.43	0.38	0.78	0.70	0.63	0.33	0.33	0.53	0.63
	LPIPS↓	0.49	0.45	0.43	0.47	0.45	0.20	0.20	0.30	0.34	0.57	0.54	0.40	0.32
	PSNR↑	15.01	12.81	14.62	14.27	14.24	21.07	21.07	21.22	19.54	14.30	13.41	20.71	20.81
MonoGS	SSIM↑	0.34	0.43	0.52	0.53	0.55	0.20	0.83	0.74	0.82	-	0.52	0.23	0.24
	LPIPS↓	0.85	0.78	0.75	0.68	0.61	0.85	0.40	0.63	0.56	-	0.72	0.96	0.94
	PSNR↑	10.63	12.59	15.01	16.08	15.63	10.20	22.63	19.29	23.00	-	14.49	11.06	11.50
PhotoSLAM	SSIM↑	0.44	0.52	0.48	0.51	0.56	0.51	0.74	0.69	0.76	0.39	0.57	0.31	0.27
	LPIPS↓	0.66	0.56	0.65	0.55	0.49	0.65	0.39	0.47	0.42	0.71	0.56	0.76	0.67
	PSNR↑	15.25	15.03	14.25	15.57	15.81	14.78	15.08	15.35	15.99	11.57	11.40	15.47	14.09
Ours	SSIM↑	0.68	0.73	0.79	0.80	0.80	0.82	0.85	0.86	0.85	0.81	0.78	0.82	0.82
	LPIPS↓	0.26	0.29	0.27	0.17	0.17	0.16	0.18	0.16	0.19	0.22	0.21	0.15	0.15
	PSNR↑	19.96	20.15	20.93	24.52	22.82	24.47	23.48	24.72	23.76	22.07	21.46	25.45	25.21

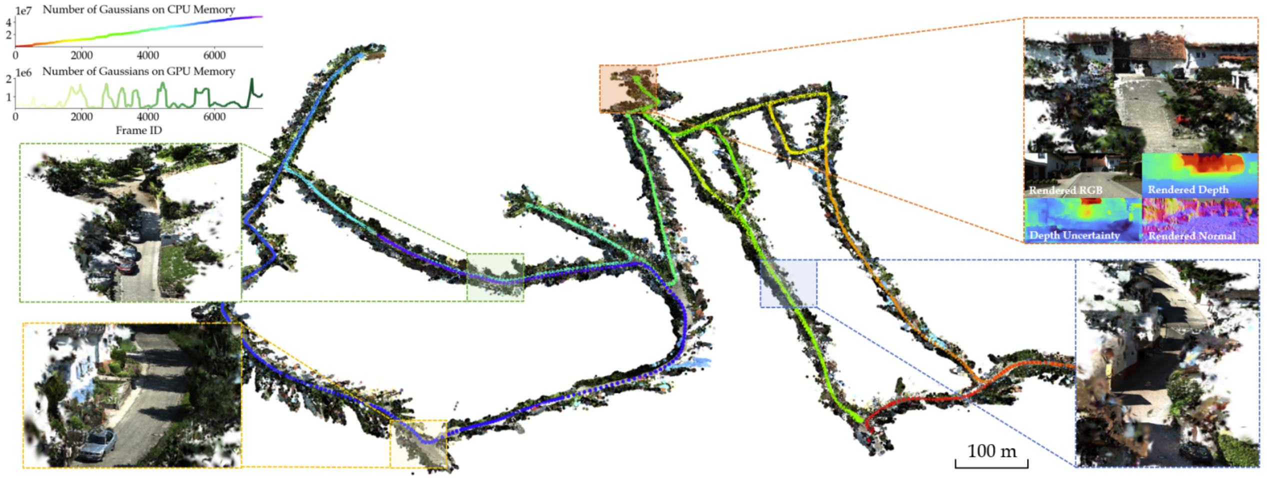


Fig. 8: Visualization of KITTI360's gaussian map. The trajectory length of scene 2013_05_28_drive_0006 is 8.05 km, and the entire Gaussian map contains 51.73 million ellipsoids. We recorded the number of Gaussians throughout the training process and zoomed in on different parts of the map for clearer visualization.

TABLE V: Quantitative results on the indoor datasets Scannet and BundleFusion. We mark the best two results with first and second. All quantitative metrics are computed as averages based on renderings at the same keyframes.

		ScanNet						BundleFusion				
		0054	0059	0106	0169	0233	0465	apt0	apt2	copyroom	office0	office2
GO-SLAM	SSIM↑	0.59	0.32	0.47	0.42	0.48	0.09	0.52	0.34	0.61	0.23	0.51
	LPIPS↓	0.53	0.60	0.59	0.57	0.55	0.75	0.54	0.59	0.49	0.72	0.55
	PSNR↑	19.70	13.15	14.58	14.49	17.22	8.65	17.24	12.24	18.40	12.60	17.31
MonoGS	SSIM↑	0.83	0.74	0.76	0.78	0.74	0.69	0.74	0.39	0.78	0.68	0.67
	LPIPS↓	0.61	0.59	0.60	0.61	0.67	0.74	0.62	0.82	0.57	0.68	0.67
	PSNR↑	21.37	18.55	17.58	19.15	19.73	17.19	18.80	11.50	17.83	16.76	18.98
PhotoSLAM	SSIM↑	0.83	0.772	0.78	0.79	0.78	0.74	0.66	0.59	0.73	0.43	0.33
	LPIPS↓	0.35	0.41	0.37	0.39	0.37	0.45	0.56	0.60	0.36	0.63	0.68
	PSNR↑	20.54	17.17	16.09	17.46	23.95	19.88	11.46	11.68	16.96	9.21	8.55
Ours	SSIM↑	0.84	0.775	0.83	0.80	0.77	0.69	0.75	0.63	0.74	0.65	0.68
	LPIPS↓	0.20	0.24	0.18	0.22	0.22	0.25	0.28	0.41	0.33	0.39	0.23
	PSNR↑	26.31	20.51	23.10	22.27	23.67	21.27	20.45	18.61	18.47	19.85	22.23

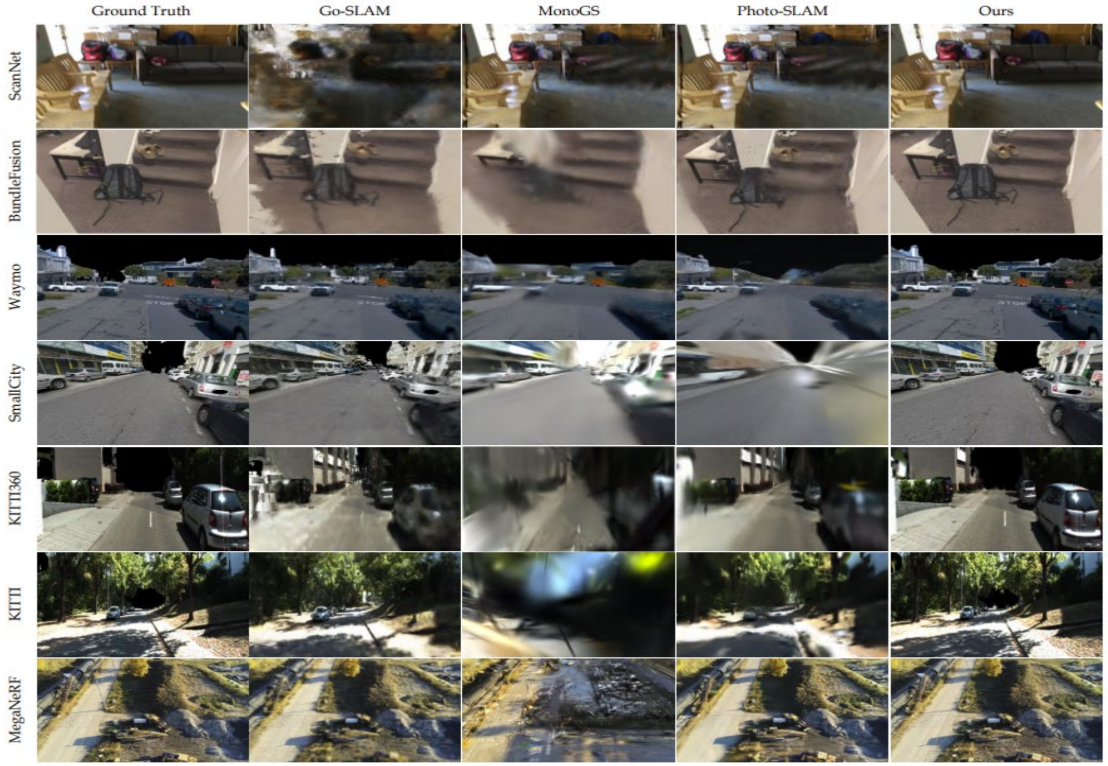
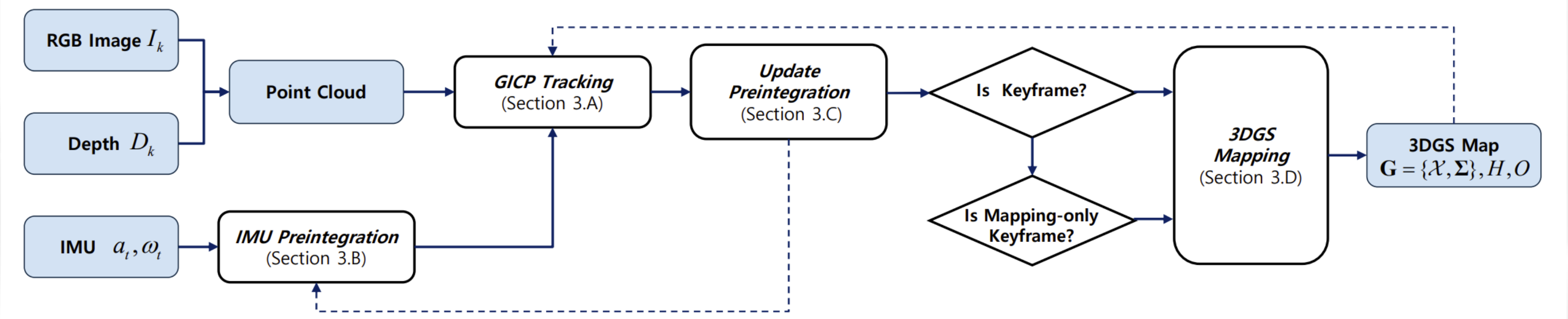


Fig. 7: Qualitative Rendering Results. We compared our method on two indoor [54], [55] and five outdoor scenes [56]–[60], with three advanced monocular SLAM algorithms, including the NeRF-based GO-SLAM [7] and two GS-based methods, MonoGS [5] and PhotoSLAM [6]. VINGS-Mono significantly outperforms existing methods in rendering quality.

VIGS SLAM: IMU-based Large-Scale 3D Gaussian Splatting SLAM

香 港 大 學
THE UNIVERSITY OF HONG KONG

- RGB-D and IMU sensors for **large-scale indoor** environments, build based on GS-ICP SLAM;
- ICP-based tracking framework that combines IMU pre-integration to provide a good initial guess for accurate pose estimation;



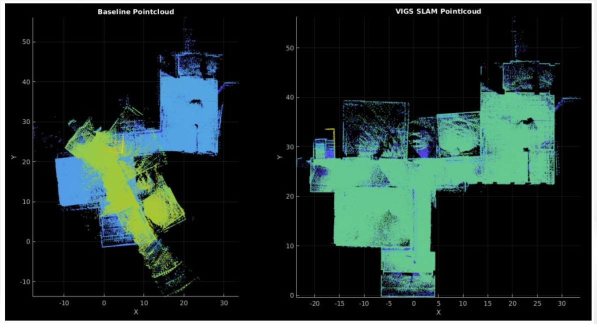


Fig. 3. Comparison of map reconstruction on uHumansV1. GS-ICP SLAM (left), VIGS SLAM (right)

TABLE I

TRACKING AND RENDERING PERFORMANCE COMPARED TO EXISTING METHODS ON UHUMANSV1 DATASET

uHumansV1 [22]		*Humans12*				*Humans24*				*Humans60*			
		ATE ↓ (cm)	PSNR ↑	SSIM ↑	LPIPS↓	ATE ↓ (cm)	PSNR ↑	SSIM ↑	LPIPS ↓	ATE ↓ (cm)	PSNR ↑	SSIM ↑	LPIPS ↓
Traditional VIO	ORB-SLAM3 [12]	191.76	-	-	-	15.82	-	-	-	17.15	-	-	-
	VINS-Mono [16]	18.85	-	-	-	25.60	-	-	-	22.29	-	-	-
3DGS	MonoGS [6]	603.50	20.99	0.714	0.503	1138.15	21.35	0.740	0.499	970.68	17.49	0.637	0.612
	PhotoSLAM [8]	650.97	14.27	0.596	0.598	1573.21	14.23	0.609	0.592	1315.79	14.78	0.596	0.587
	GS-ICP SLAM [10]	677.87	22.89	0.788	0.345	776.79	23.49	0.794	0.334	973.49	20.85	0.741	0.422
	Ours (VIGS SLAM)	**35.35**	**26.22**	**0.849**	**0.205**	**25.03**	**25.89**	**0.853**	**0.207**	**46.86**	**23.91**	**0.820**	**0.252**

The algorithm with the highest performance is indicated in **bold**, and the second-best is underlined. Traditional VIO methods are excepted in best algorithm.

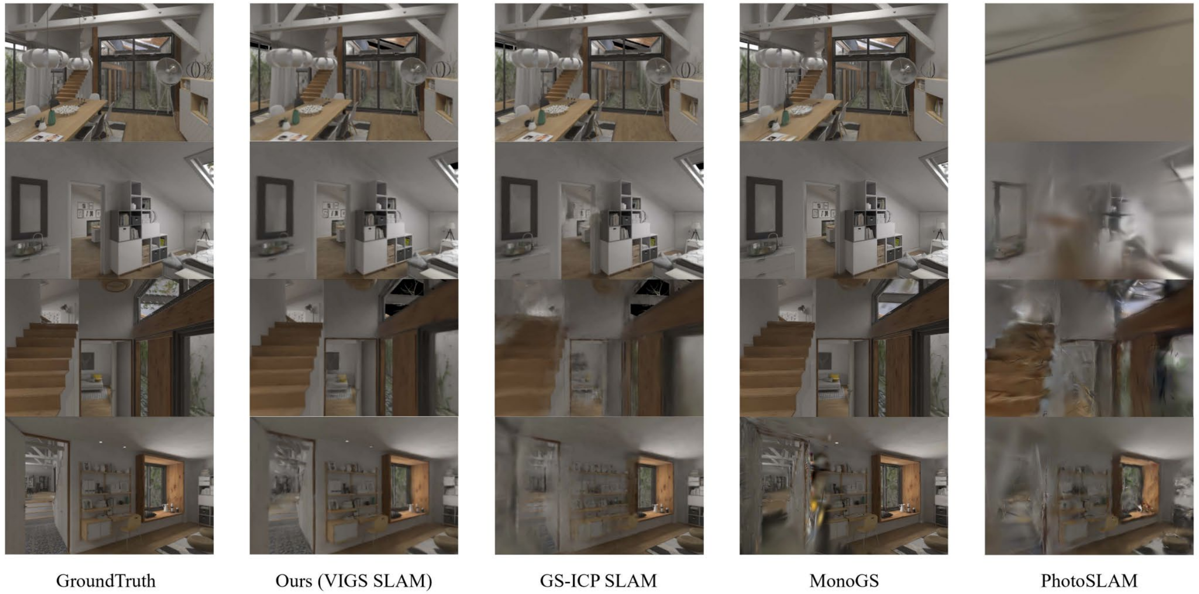


Fig. 5. Qualitative rendering results on the apartment scene of uHumansV2 dataset, compared with VIGS SLAM(ours), GS-ICP SLAM, MonoGS, and PhotoSLAM.

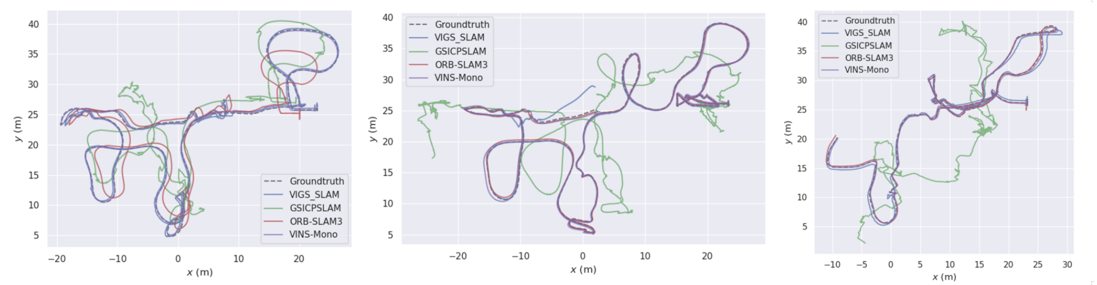


Fig. 4. Trajectory in Humans12(left), Humans24(middle), Humans60(right) of uHumansV1, compared with ORB-SLAM3, VINS-Mono, GS-ICP SLAM, and VIGS SLAM(ours).

RGBDS-SLAM: A RGB-D Semantic Dense SLAM Based on 3D Multi Level Pyramid Gaussian Splatting

香 港 大 學
THE UNIVERSITY OF HONG KONG

- A RGB-D semantic dense SLAM system based on 3D **multi-level pyramid gaussian splatting**, which uses multi-level image pyramid to extract rich detail information at different resolution levels and perform gaussian splatting training;
- Built on Photo-SLAM (also with multi-level pyramid) with a tightly coupled **multi-features reconstruction optimization**, which reasonably couples RGB, depth, and semantic features through various constraints;

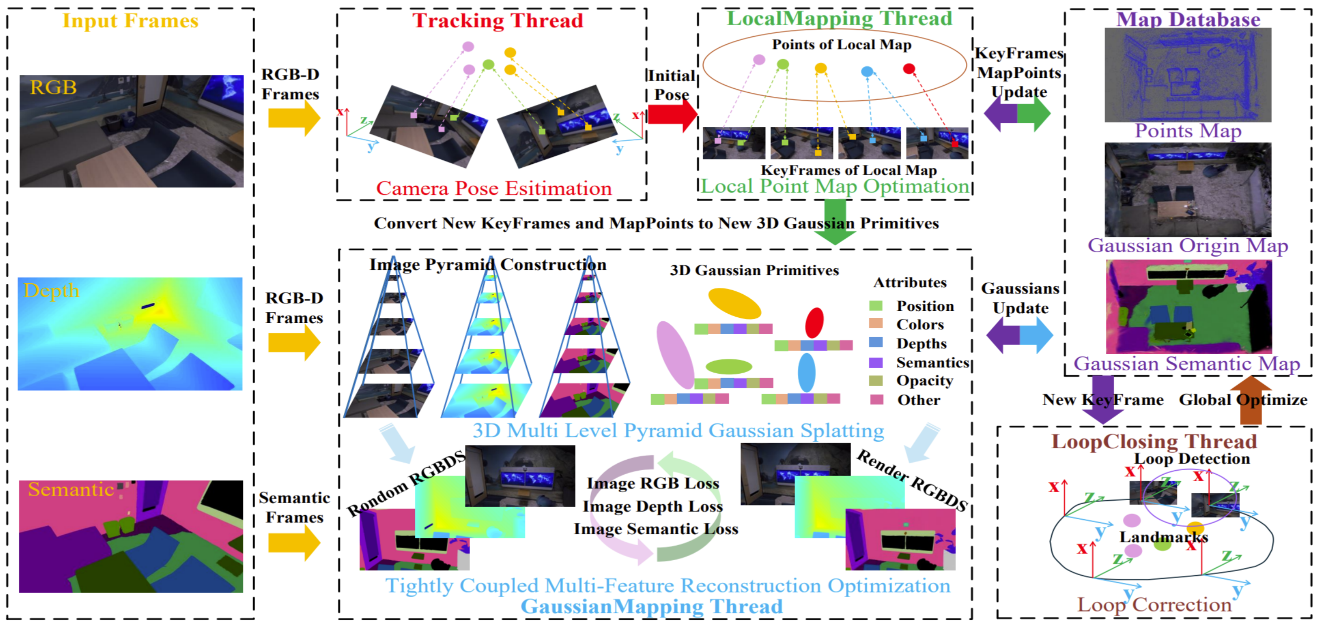


Fig. 1. Overview of the proposed RGBDS-SLAM. Our method is an enhancement of ORB-SLAM3 [6], taking RGB, depth, and semantic frames as input and outputting a map database with the point map, gaussian origin map, and gaussian semantic map. It consists of four threads: Tracking, LocalMapping, GaussianMapping, and LoopClosing.

For RGB images, we consider L1 and SSIM loss:

$$L_r(i) = (1-\lambda_r)\left|I_r^{rd}(i) - I_r^{gt}(i)\right| + \lambda_r SSIM(I_r^{rd}(i), I_r^{gt}(i)) \tag{10}$$

For depth images, we only consider L1 loss:

$$L_d(i) = \left|I_d^{rd}(i) - I_d^{gt}(i)\right| \tag{11}$$

For semantic images, we similarly consider L1 and SSIM loss:

$$L_s(i) = (1-\lambda_s)\left|I_s^{rd}(i) - I_s^{gt}(i)\right| + \lambda_s SSIM(I_s^{rd}(i), I_s^{gt}(i)) \tag{12}$$

Finally, we tightly couple multiple features into a reconstruction optimization framework to perform joint optimization:

$$L_{reconstruction}(i) = L_r(i) + L_d(i) + L_s(i) \tag{13}$$

Performance of RGBDS-SLAM

- Built on Photo-SLAM (just add the semantic on photo-SLAM);

TABLE I
QUANTITATIVE COMPARISON OF RGB RECONSTRUCTION QUALITY BETWEEN OUR METHOD AND BASELINES ON 8 SEQUENCES OF REPLICA DATASET.

Method		Metric	office0	office1	office2	office3	office4	room0	room1	room2	avg
NeRF-based SLAM	NICE-SLAM [8]	PSNR↑	29.07	30.34	19.66	22.23	24.94	22.12	22.47	24.52	24.42
		SSIM↑	0.874	0.886	0.797	0.801	0.856	0.689	0.757	0.814	0.809
		LPIPS↓	0.229	0.181	0.235	0.209	0.198	0.330	0.271	0.208	0.233
	Vox-Fusion [9]	PSNR↑	27.79	29.83	20.33	23.47	25.21	22.39	22.36	23.92	24.41
		SSIM↑	0.857	0.876	0.794	0.803	0.847	0.683	0.751	0.798	0.801
		LPIPS↓	0.241	0.184	0.243	0.213	0.199	0.303	0.269	0.234	0.236
	Co-SLAM [10]	PSNR↑	34.14	34.87	28.43	28.76	30.91	27.27	28.45	29.06	30.24
		SSIM↑	0.961	0.969	0.938	0.941	0.955	0.910	0.909	0.932	0.939
		LPIPS↓	0.209	0.196	0.258	0.229	0.236	0.324	0.294	0.266	0.252
	ESLAM [10]	PSNR↑	33.71	30.20	28.09	28.77	29.71	25.32	27.77	29.08	29.08
		SSIM↑	0.960	0.923	0.943	0.948	0.945	0.875	0.902	0.932	0.929
		LPIPS↓	0.184	0.228	0.241	0.196	0.204	0.313	0.298	0.248	0.239
3D GS-based SLAM	SplaTAM [17]	PSNR↑	38.26	39.17	31.97	29.70	31.81	32.86	33.89	35.25	34.11
		SSIM↑	0.98	0.98	0.97	0.95	0.95	0.98	0.97	0.98	0.970
		LPIPS↓	0.09	0.09	0.10	0.12	0.15	0.07	0.10	0.08	0.100
	Photo-SLAM [21]	PSNR↑	38.48	39.09	33.03	33.79	36.02	30.72	33.51	35.03	34.96
		SSIM↑	0.964	0.961	0.938	0.938	0.952	0.899	0.934	0.951	0.942
		LPIPS↓	0.050	0.047	0.077	0.066	0.054	0.075	0.057	0.043	0.059
	NEDS-SLAM [22]	PSNR↑	/	/	/	/	/	/	/	/	34.76
		SSIM↑	/	/	/	/	/	/	/	/	0.962
		LPIPS↓	/	/	/	/	/	/	/	/	0.088
	SGS-SLAM [24]	PSNR↑	38.54	39.20	32.90	32.05	32.75	32.50	34.25	35.10	34.66
		SSIM↑	**0.984**	**0.982**	**0.965**	**0.966**	0.949	**0.976**	**0.978**	**0.982**	**0.973**
		LPIPS↓	0.086	0.087	0.101	0.115	0.148	0.070	0.094	0.070	0.096
	RGBDS-SLAM(Ours)	PSNR↑	**42.46**	**42.57**	**35.80**	**36.53**	**39.47**	**35.77**	**38.59**	**39.58**	**38.85**
		SSIM↑	0.981	0.976	0.959	0.958	**0.969**	0.955	0.968	0.973	0.967
		LPIPS↓	**0.023**	**0.029**	**0.052**	**0.046**	**0.034**	**0.037**	**0.029**	**0.027**	**0.035**

/ indicates that the paper does not provide relevant data, **bold data** indicates optimal data, and underlined data indicates suboptimal data.

TABLE II
QUANTITATIVE COMPARISON OF AVERAGE RESULTS ON DEPTH, ATE, AND FPS METRICS BETWEEN OUR METHOD AND BASELINES ON 8 SEQUENCES OF REPLICA DATASET.

Method		Depth(cm)↓	ATE Mean (cm)↓	ATE RMSE (cm)↓	Tracking FPS↑	Mapping FPS↑
NeRF-based SLAM	NICE-SLAM [8]	1.903	1.795	2.503	13.70	0.20
	Vox-Fusion [9]	2.913	1.027	1.473	2.11	2.17
	Co-SLAM [10]	1.513	0.935	1.059	17.24	10.20
	ESLAM [11]	0.945	0.545	0.678	18.11	3.62
	SNI-SLAM [15]	0.766	0.397	0.456	16.03	2.48
3D GS-based SLAM	SplaTAM [17]	0.490	/	0.360	5.26	3.03
	Photo-SLAM [21]	/	/	0.604	**42.49**	/
	NEDS-SLAM [22]	0.470	/	0.354	/	/
	SGS-SLAM [24]	0.356	**0.327**	**0.412**	5.27	3.52
	RGBDS-SLAM(Ours)	**0.342**	0.499	0.589	29.55	**32.22**

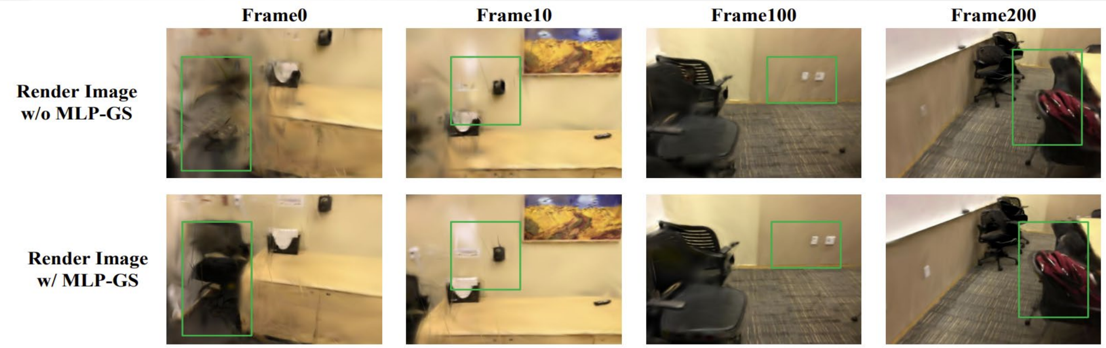


Fig. 6. Ablation study of the multi-level pyramid gaussian splatting in our proposed method on ScanNet dataset. The first row shows the multi-frame RGB image rendering results using the standard GS process instead of our proposed MLP-GS. The second row shows the corresponding multi-frame RGB image rendering results using MLP-GS. The areas with significant differences in the images are highlighted with green boxes.

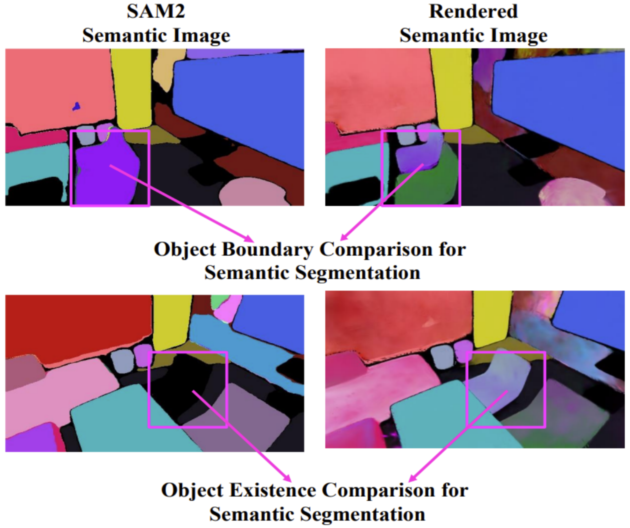


Fig. 7. Comparison between the SAM2 segmentation results and the rendered results after our method performs semantic reconstruction. The first row displays a comparison of object boundaries in the semantic segmentation, while the second row shows a comparison of object existence in the semantic segmentation.

OpenGS-SLAM: Open-Set Dense Semantic SLAM with 3D Gaussian Splatting for Object-Level Scene Understanding

香 港 大 學
THE UNIVERSITY OF HONG KONG

- 3DGS for dense semantic (object-level) SLAM in open-set environments, incorporates 2D semantic label to 3D explicit semantic label to each Gaussian;
- To solve the **non-differentiable nature of the semantic label attribute**, Gaussian voting splatting is proposed for fast 2D label map rendering and scene updating;
- Confidence-based 2D label consensus method is designed for **consistent labeling across multiple views**;
- Segmentation Counter Pruning strategy;

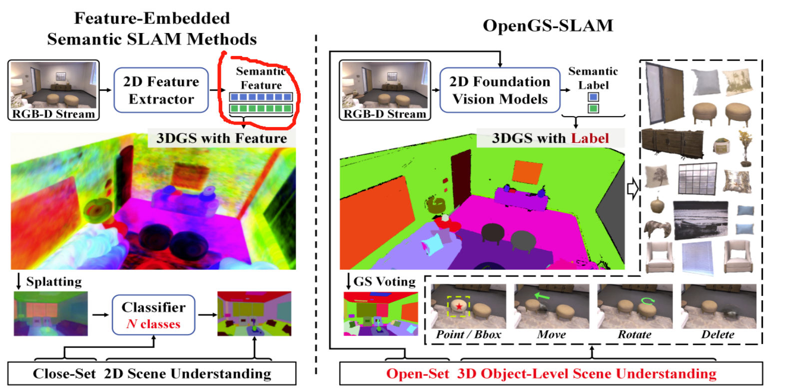


Fig. 1: Compared to the feature-embedded methods [12], [13], our approach integrates semantic labels into the 3D Gaussian scene representation, ensuring that Gaussians belonging to the same object are consistently labeled. This enables more effective 3D object-level scene understanding and interaction. By leveraging 2D foundational vision models, our approach facilitates open-set dense semantic SLAM. The images on the left are from [12].

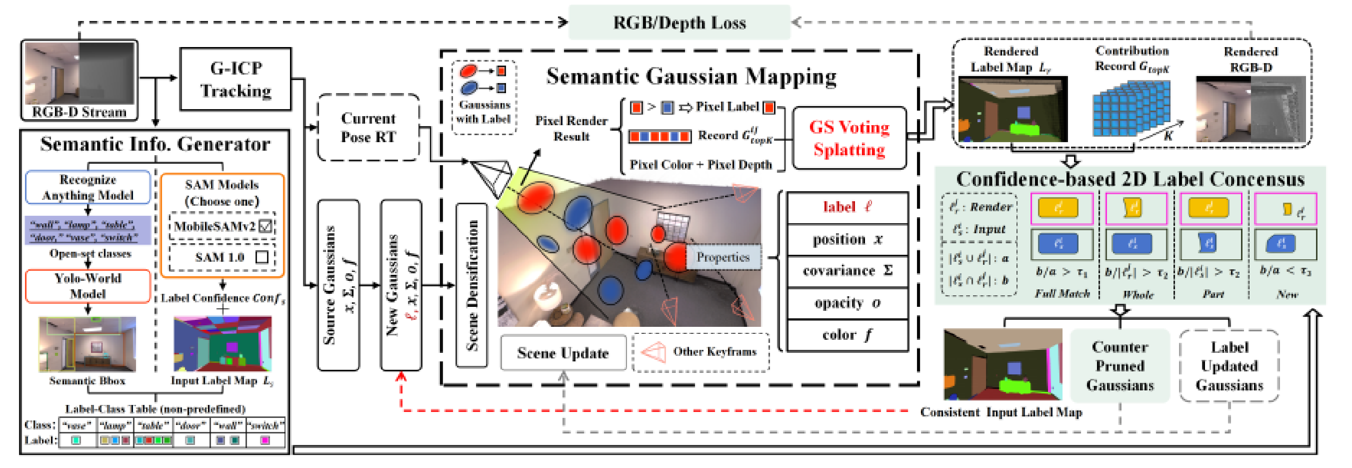
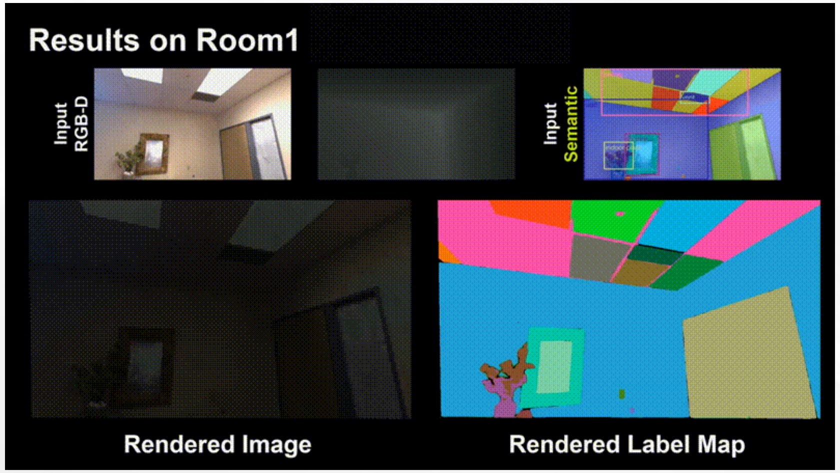


Fig. 2: **An overview of OpenGS-SLAM.** Our method takes an RGB-D stream as input. RGB images are first processed by the Semantic Information Generator and G-ICP to extract semantic information and estimate the current pose. Using this pose, we perform precise and efficient semantic rendering via Gaussian Voting Splatting. We then unify the input label map with the current map through Confidence-based 2D Label Consensus, ensuring semantic consistency. During this process, partial Gaussian data is updated, and counter Gaussians are pruned.



OpenGS-SLAM

- Not require any predefined semantic categories;

TABLE I: Quantitative comparison of semantic segmentation accuracy(mIoU↑) against Radiance-Based Semantic SLAM methods on the Replica [35] datasets. All models use the ground-truth semantic labels from the replica dataset.

Methods	R0	R1	R2	Of0	Of1	Of2	Of3	Of4
NIDS-SLAM [8]	82.45	84.08	76.99	85.94	-	-	-	-
DNS-SLAM [9]	88.32	84.90	81.20	84.66	-	-	-	-
SNI-SLAM [7]	88.42	87.43	86.16	87.63	78.63	86.49	74.01	80.22
SGS-SLAM [17]	92.95	92.91	92.10	92.90	-	-	-	-
NEDS-SLAM [13]	90.73	91.20	-	90.42	-	-	-	-
Ours(Prior)	**93.24**	**94.11**	**92.79**	**93.22**	**92.16**	**93.25**	**93.14**	**93.77**

TABLE II: Comparison of zero-shot novel-view Semantic Segmentation with 3DGS-based **open-set** scene understanding methods. (average performance on Replica [35])

Method	mIoU(%) ↑	Acc(%) ↑	Render FPS ↑	Learnable Parameters(MB) ↓
Featrue 3DGS [26]	48.89	57.51	11.03	894.91
GS-Grouping [28]	59.15	69.94	16.14	717.12
Ours(MobileSAMv2)	57.48	67.14	164.78	314.11
Ours(SAM1.0)	**61.91**	**73.11**	**165.47**	**301.71**

TABLE III: Camera Tracking and Reconstruction Results on Replica [35]. (average performance on 8 scenes)

Category	Methods	ATE↓	PSNR↑	SSIM↑	LPIPS↓
Visual SLAM	SplaTAM [10]	0.35	33.91	0.969	0.097
	GS-SLAM [37]	0.50	34.27	0.975	0.082
	LoopSplat [38]	0.26	36.63	**0.985**	0.112
	GICP-SLAM [16]	**0.16**	**38.86**	0.976	**0.041**
Semantic SLAM	SNI-SLAM [7]	0.46	29.43	0.935	0.235
	SGS-SLAM [17]	0.41	34.66	0.973	0.096
	NEDS-SLAM [13]	0.35	34.76	0.962	0.088
	SemGauss-SLAM [12]	0.33	35.03	**0.982**	0.062
	Ours (SAM1.0)	**0.16**	**39.49**	0.978	**0.034**

TABLE IV: Camera Tracking Results on Tum [36]. ATE.↓

Category	Methods	fr1-desk	fr2-xyz	fr3-office	Avg.
Visual SLAM	SplaTAM [10]	3.35	1.24	5.16	3.25
	GS-SLAM [37]	3.65	-	-	-
	LoopSplat [38]	**2.08**	1.58	3.22	2.29
	GICP-SLAM [16]	2.41	1.77	2.67	2.28
Semantic SLAM	Ours (SAM1.0)	2.40	**1.57**	**2.48**	**2.15**

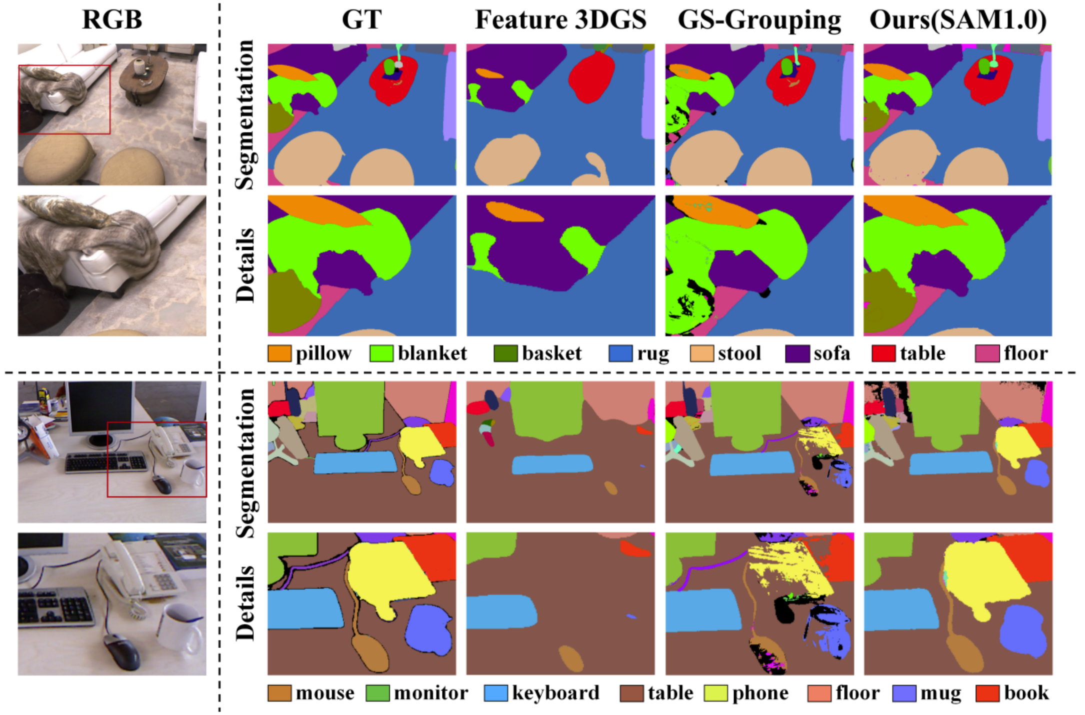


Fig. 4: Qualitative comparison of novel-view **open-set** semantic segmentation. For TUM, novel views refer to viewpoints that are not included in the training data, and the ground truth is obtained from manual annotations.

FGS-SLAM: Fourier-based Gaussian Splatting for Real-time SLAM with Sparse and Dense Map Fusion

香 港 大 學
THE UNIVERSITY OF HONG KONG

- **Uncertainty in gaussian position and initialization** parameters introduces challenges for 3DGS. Thus, introducing **adaptive densification method** based on **Fourier frequency domain analysis** to establish gaussian priors for rapid convergence.
- Map-sharing mechanism, the sparse map is for efficient GICP pose tracking, and dense map for high-fidelity visual representations;
- First SLAM system **leveraging frequency domain analysis for gaussian initialization** with 36 FPS;

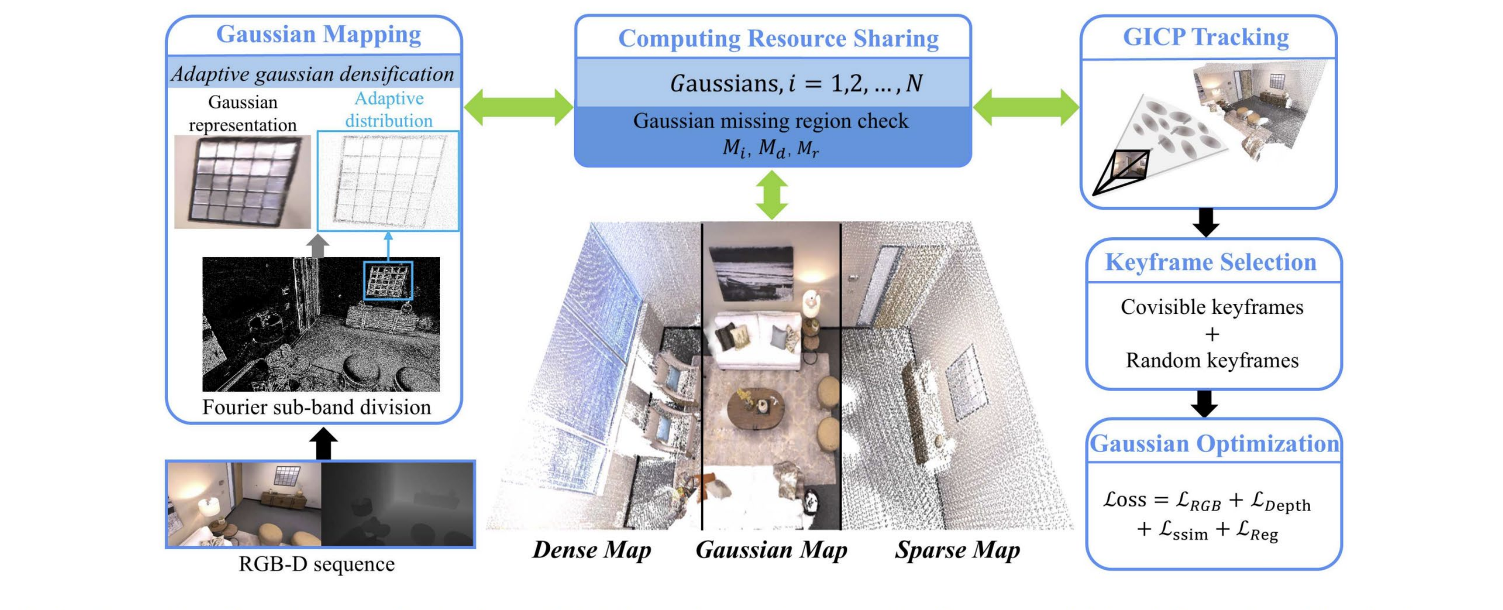
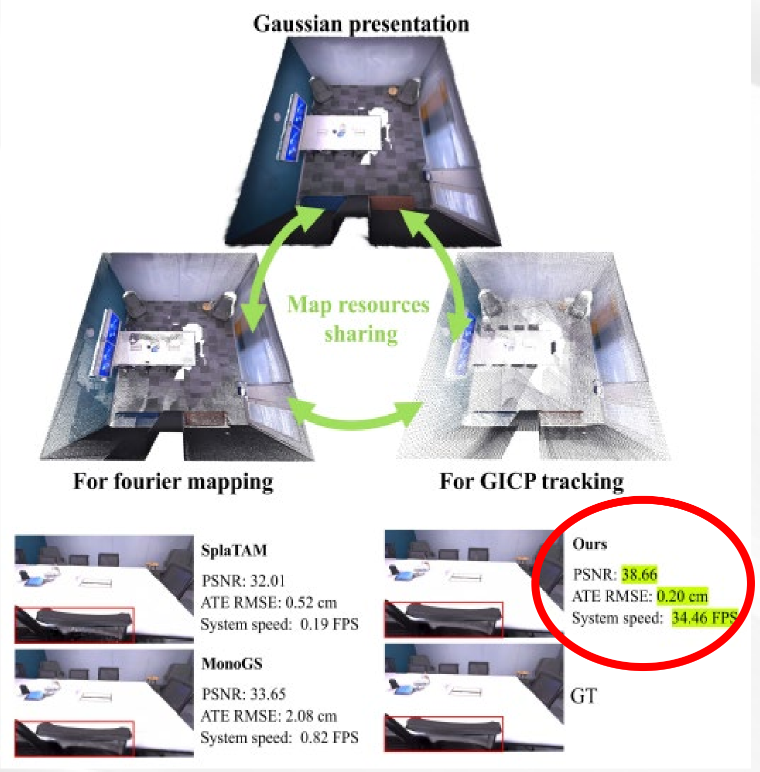


Fig. 2. System Overview. The proposed method uses RGB-D data as input to the system. Mapping: The spatial domain is transformed into the frequency domain through Fourier transforms. New gaussians are adaptively initialized based on high and low frequency regions, thereby constructing a gaussian dense map. Resource Sharing: The system simultaneously constructs both sparse and dense maps, with map points stored using gaussian attributes. Gaussian attributes and the mask of missing gaussian regions are shared between the maps. Tracking: GICP performs rapid registration using the 3D gaussian point cloud of the sparse map, and supplements the gaussian points in the sparse map. The system selects co-visibility and random keyframes, and jointly optimizes the gaussian map through gaussian rasterization rendering.



FGS-SLAM Performance

香 港 大 學
THE UNIVERSITY OF HONG KONG

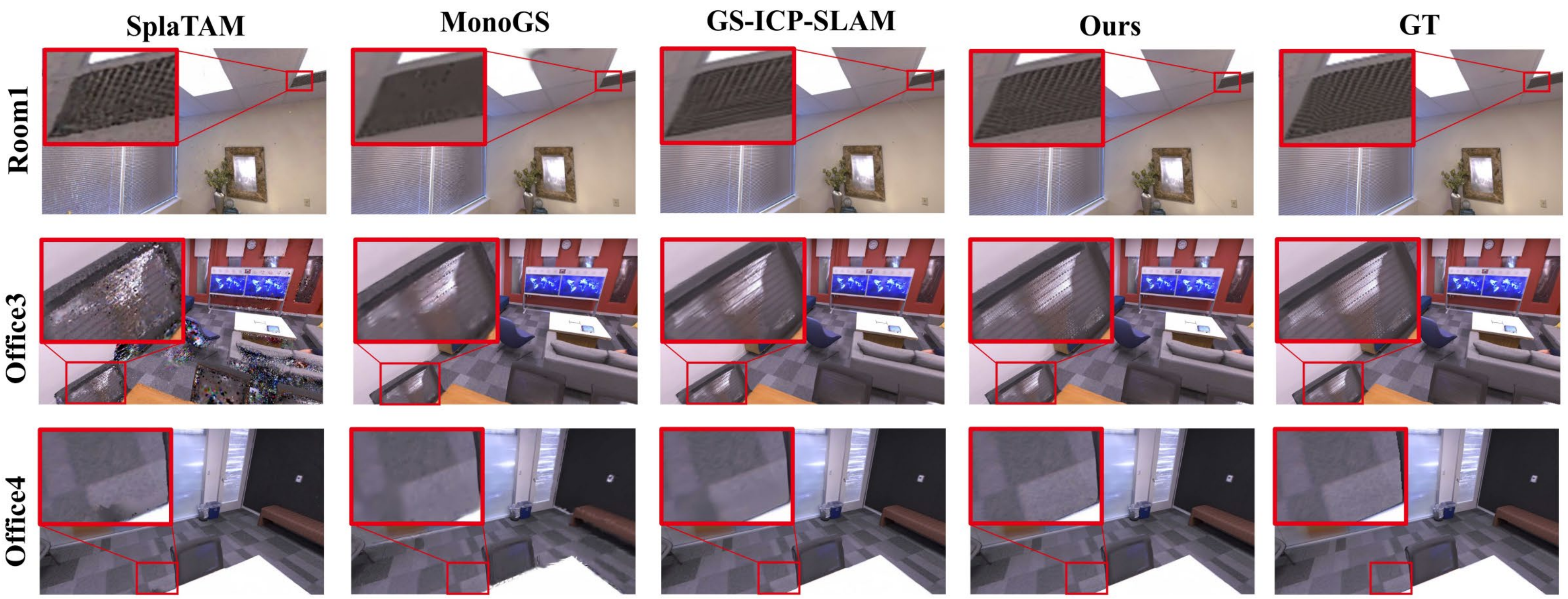


Fig. 4. Qualitative result comparison on the Replica dataset. Detail zoom-ins from three scenes are presented. Our method outperforms other frameworks in the reconstruction of map details.

TABLE I

CAMERA POSE ESTIMATION RESULTS ON THE REPLICA DATASET (ATE RMSE↓[CM]). OUR METHOD DEMONSTRATES SUPERIOR PERFORMANCE ACROSS ALL 8 SCENES, OUTPERFORMING THE CURRENT STATE-OF-THE-ART (SOTA) BASELINES. SOME BASELINE DATA IS SOURCED FROM [22].

Method	R0	R1	R2	OF0	OF1	OF2	OF3	OF4	Avg.
NICE-SLAM [8]	0.97	1.31	1.07	0.88	1.00	1.06	1.10	1.13	1.06
Point-SLAM [21]	0.56	0.47	0.30	0.35	0.62	0.55	0.72	0.73	0.54
Co-SLAM [9]	0.77	1.04	1.09	0.58	0.53	2.05	1.49	0.84	0.99
GS-SLAM [20]	0.48	0.53	0.33	0.52	0.41	0.59	0.46	0.70	0.50
SplaTAM [14]†	0.27	0.31	0.63	0.49	0.22	0.30	0.35	0.52	0.39
MonoGS (RGB-D)[15]†	0.35	0.26	0.27	0.41	0.40	0.22	0.14	2.10	0.52
CG-SLAM [22]	0.29	0.27	0.25	0.33	0.14	0.28	0.31	0.29	0.27
Ours	0.14	0.17	0.10	0.16	0.13	0.16	0.16	0.20	0.15

† denotes the reproduced results by running officially released code.

TABLE III

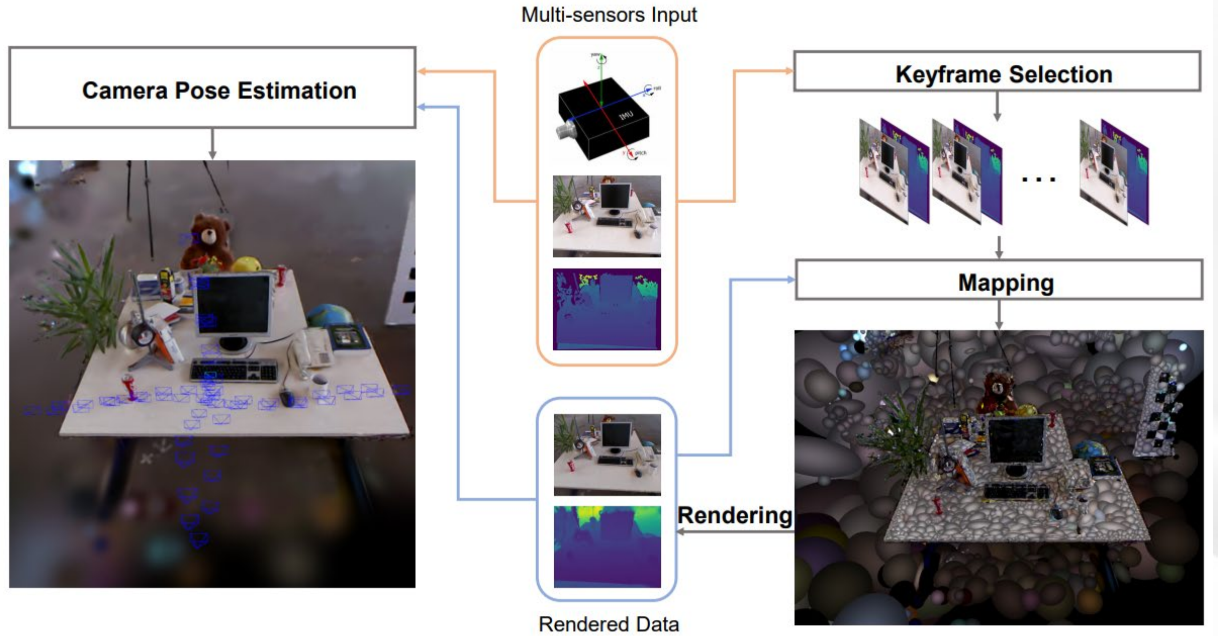
RENDERING RESULTS ON THE REPLICA DATASET. OUR METHOD ACHIEVES AN OPTIMAL BALANCE BETWEEN SYSTEM SPEED AND MAPPING QUALITY. SOME BASELINE DATA IS SOURCED FROM [21], [23].

Method	Metrics	R0	R1	R2	OF0	OF1	OF2	OF3	OF4	Avg.	FPS ↑
NICE-SLAM [8]	PSNR[dB]↑	22.12	22.47	24.52	29.07	30.34	19.66	22.23	24.94	24.42	
	SSIM ↑	0.689	0.757	0.814	0.874	0.886	0.797	0.801	0.856	0.809	-
	LPIPS ↓	0.330	0.271	0.208	0.229	0.181	0.235	0.209	0.198	0.233	
Point-SLAM [21]	PSNR[dB]↑	33.38	34.10	36.32	38.72	39.31	34.22	34.10	34.82	35.62	
	SSIM ↑	0.979	0.977	0.985	0.985	0.987	0.962	0.963	0.981	0.977	0.3
	LPIPS ↓	0.097	0.115	0.101	0.089	0.110	0.152	0.119	0.131	0.114	
GS-SLAM [20]	PSNR[dB]↑	31.56	32.86	32.59	38.70	41.17	32.36	32.03	32.92	34.27	
	SSIM ↑	0.968	0.973	0.971	0.986	0.993	0.978	0.970	0.968	0.975	8.34
	LPIPS ↓	0.094	0.075	0.093	0.050	0.033	0.094	0.110	0.112	0.082	
SplaTAM [14]†	PSNR[dB]↑	32.60	33.63	34.91	38.15	39.05	31.89	30.18	32.01	34.05	
	SSIM ↑	0.975	0.969	0.982	0.981	0.981	0.966	0.951	0.948	0.969	0.18
	LPIPS ↓	0.070	0.097	0.073	0.088	0.094	0.100	0.118	0.154	0.099	
MonoGS (RGB-D)[15]†	PSNR[dB]↑	33.21	35.88	36.86	40.49	41.39	35.62	35.48	33.65	36.57	
	SSIM ↑	0.937	0.954	0.961	0.974	0.975	0.958	0.957	0.940	0.957	0.81
	LPIPS ↓	0.081	0.092	0.075	0.061	0.053	0.071	0.059	0.112	0.076	
GS-ICP-SLAM[23]†	PSNR[dB]↑	35.11	37.28	38.11	42.38	42.76	36.77	36.80	38.54	38.55	
	SSIM ↑	0.960	0.968	0.973	0.984	0.982	0.971	0.968	0.967	0.970	29.95
	LPIPS ↓	0.053	0.051	0.053	0.032	0.036	0.048	0.047	0.049	0.045	
Ours	PSNR[dB]↑	35.27	38.05	38.63	42.73	43.18	36.42	37.04	38.66	38.75	
	SSIM ↑	0.961	0.972	0.975	0.984	0.984	0.973	0.969	0.972	0.974	32.75
	LPIPS ↓	0.045	0.043	0.045	0.028	0.035	0.045	0.040	0.046	0.041	

GI-SLAM: Gaussian-Inertial SLAM

香 港 大 學
THE UNIVERSITY OF HONG KONG

- Gaussian-inertial SLAM system which consists of an IMU-enhanced camera tracking module and 3D Gaussian-based scene representation for mapping (VIO + 3DGS);
- It seems that this work is built based on MonoGS for monocular, stereo, and RGBD cameras, both with and without IMU integration;
- Introducing IMU loss function, which, when combined with the photometric loss function, improves the camera tracking;



IMU Loss Our IMU loss function incorporates both translational and rotational constraints from 6-DOF IMU measurements. For translational constraints, we integrate the linear acceleration measurements $\mathbf{a}_t$ with the previous camera frame's linear velocity $\mathbf{v}_{t-1}$:

$$\Delta\mathbf{p}_{imu} = \mathbf{v}_{t-1}\Delta t + \frac{1}{2}\mathbf{a}_t\Delta t^2. \quad (5)$$

The translation loss $\mathcal{L}_{trans}$ is then computed as:

$$\mathcal{L}_{trans} = \|\Delta\mathbf{p}_{opt} - \Delta\mathbf{p}_{imu}\|_2^2, \quad (6)$$

where $\Delta\mathbf{p}_{opt} \in \mathbb{R}^3$ denotes the optimized displacement between consecutive frames.

For rotational constraints, we derive the relative rotation from angular velocity measurements ω_t:

$$\Delta\theta_{imu} = \omega_t\Delta t. \quad (7)$$

The rotation loss $\mathcal{L}_{rot}$ is formulated as:

$$\mathcal{L}_{rot} = \|\Delta\theta_{opt} - \Delta\theta_{imu}\|_2^2, \quad (8)$$

where $\Delta\theta_{opt} \in \mathbb{R}^3$ represents the optimized relative rotation in axis-angle form. The final IMU loss combines both components through weighted summation:

$$\mathcal{L}_{imu} = \lambda_t\mathcal{L}_{trans} + \lambda_r\mathcal{L}_{rot}, \quad (9)$$

Performance of GI-SLAM

香 港 大 學
THE UNIVERSITY OF HONG KONG

Methods	Metric	fr1/desk	fr2/xyz	fr3/office	**Avg.**
NICE-SLAM[48]	PSNR↑	13.87	17.94	15.11	15.64
	SSIM↑	0.566	0.668	0.561	0.598
	LPIPS↓	0.485	0.327	0.382	0.398
Vox-Fusion[45]	PSNR↑	15.79	16.53	17.22	16.51
	SSIM↑	0.653	0.711	0.677	0.68
	LPIPS↓	0.514	0.423	0.459	0.465
Point-SLAM[30]	PSNR↑	13.87	17.61	18.93	16.8
	SSIM↑	0.627	0.715	0.744	0.695
	LPIPS↓	0.564	0.562	0.442	0.523
SplaTAM[16]	PSNR↑	<u>22.63</u>	24.55	22.71	23.29
	SSIM↑	**0.852**	**0.935**	<u>0.876</u>	**0.888**
	LPIPS↓	<u>0.239</u>	**0.103**	0.221	**0.188**
MonoGS[21]	PSNR↑	22.56	<u>24.86</u>	<u>24.37</u>	<u>23.93</u>
	SSIM↑	0.774	0.8	0.823	0.799
	LPIPS↓	0.247	0.211	<u>0.21</u>	0.223
Ours	PSNR↑	**23.98**	**25.37**	<u>24.29</u>	**24.55**
	SSIM↑	<u>0.833</u>	<u>0.851</u>	**0.881**	<u>0.855</u>
	LPIPS↓	**0.209**	<u>0.191</u>	**0.196**	<u>0.199</u>

Table 2. Rendering performance on TUM for RGBD.

Input	Methods	fr1/desk	fr2/xyz	fr3/office	**Avg.**	fr1/desk2	fr1/room	**Avg.**
Monocular	DROID-VO[34]	5.12	9.88	7.30	7.43	-	-	-
	DepthCov-VO[8]	5.63	**1.20**	53.4	20.08	-	-	-
	MonoGS[21]	<u>3.56</u>	4.59	<u>3.50</u>	<u>3.88</u>	77.64	79.88	79.36
	Ours	**1.98**	<u>3.27</u>	**3.14**	**2.80**	50.27	61.43	55.85
RGBD	NICE-SLAM[48]	4.24	6.04	3.85	4.71	4.89	33.79	19.34
	Vox-Fusion[45]	3.48	1.53	23.7	9.57	6.00	25.73	15.87
	Point-SLAM[30]	4.11	1.43	3.19	2.91	**4.54**	33.94	19.24
	SplaTAM[16]	3.34	**1.22**	5.20	3.25	6.54	11.10	8.82
	MonoGS[21]	<u>1.59</u>	1.36	<u>1.55</u>	<u>1.50</u>	6.30	<u>6.64</u>	<u>6.47</u>
	Ours	**1.34**	<u>1.26</u>	**1.54**	**1.38**	<u>4.82</u>	**4.66**	**4.74**

Table 1. Camera tracking results on TUM for monocular and RGBD(ATE RMSE ↓ [cm]).

4D Gaussian Splatting SLAM

香 港 大 學
THE UNIVERSITY OF HONG KONG

- Incrementally tracks camera poses and establishes the 4D Gaussian radiance fields with RGB-D images;
- **Instead of treating dynamic objects as noise or distractors**, the proposed method explicitly models temporal variations of the 3D Gaussian ellipsoids;
- Integrating YoLov9 to divide the primitives into static and dynamic Gaussians, using MonoGS and static Gaussians for tracking;
- Using MLP for modeling the motion of the dynamic Gaussians, using the RAFT to provide constraint to learn the motion of the dynamic Gaussians;

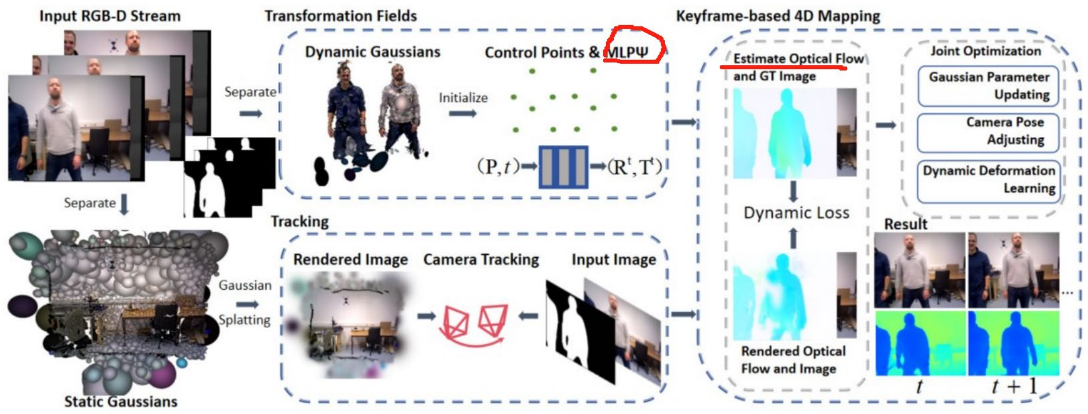
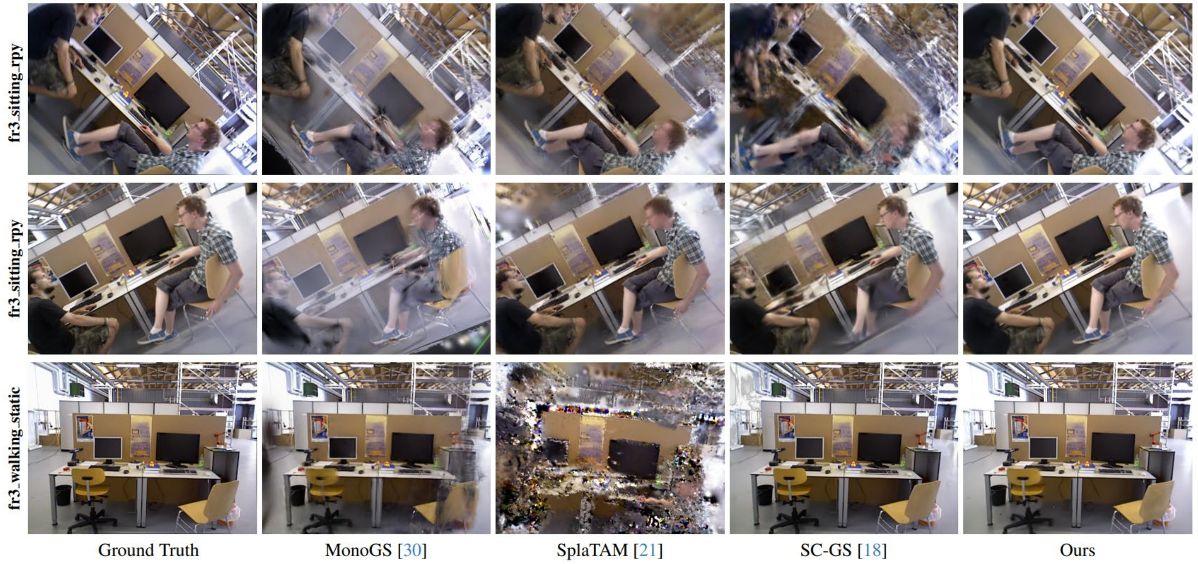


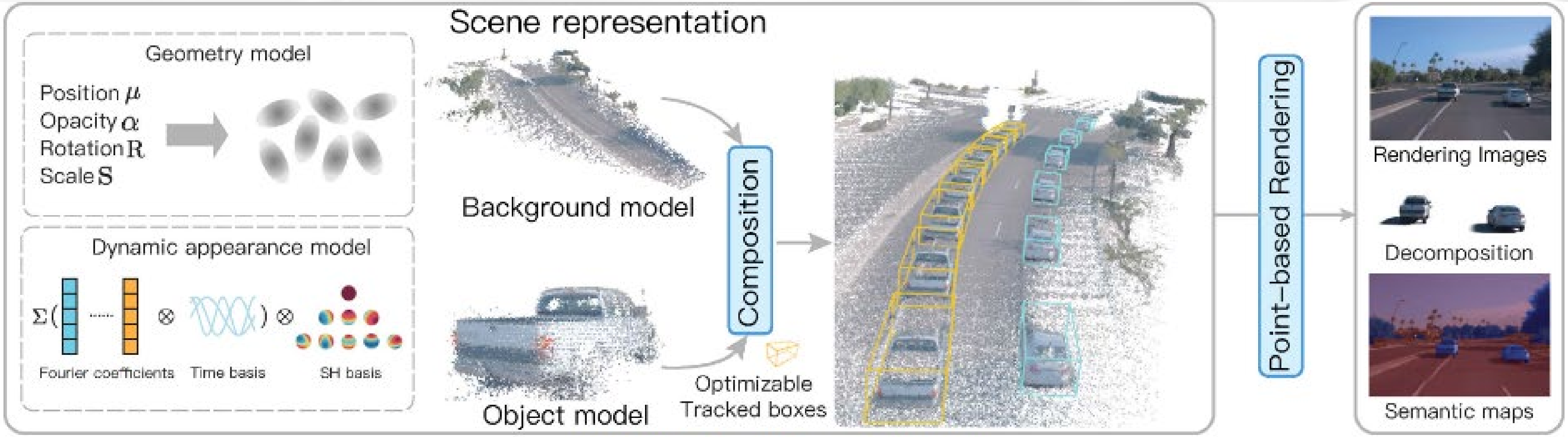
Figure 2. **Architecture of the proposed Gaussian Splatting SLAM.** The inputs to our system are temporally sequential RGB-D image sequences and motion masks.In the initial frame, dynamic and static Gaussians are independently initialized using a motion mask, and sparse control points are established according to the spatial distribution of dynamic Gaussians. The static structure is subsequently employed for camera pose estimation through photometric and geometric constraints. Following keyframe insertion, we co-optimize Gaussian attributes and camera poses while simultaneously estimating temporal motion patterns of dynamic Gaussians.



LiDAR-based 3DGS

Street Gaussians for Modeling Dynamic Urban Scenes

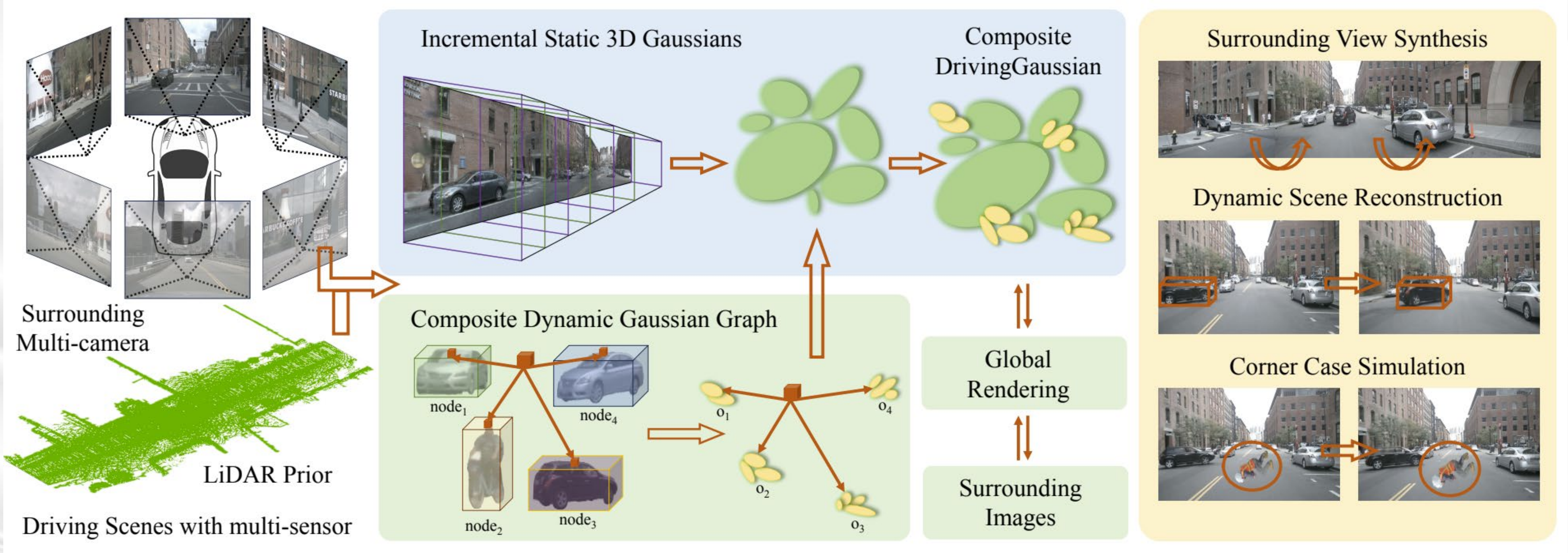
- Modeling **dynamic** urban street scenes from **monocular** videos
- The dynamic urban street is represented as a set of point clouds equipped with semantic logits and 3D Gaussians (**utilizing the point clouds to build dynamic scenes**);
- The point cloud of each foreground object vehicles is optimized with optimizable tracked poses, along with a dynamic spherical harmonics model for the dynamic appearance;
- Developing a tracked pose optimization strategy based on the proposed scene representation (need optimizable input pose);



The pipeline of Street Gaussians

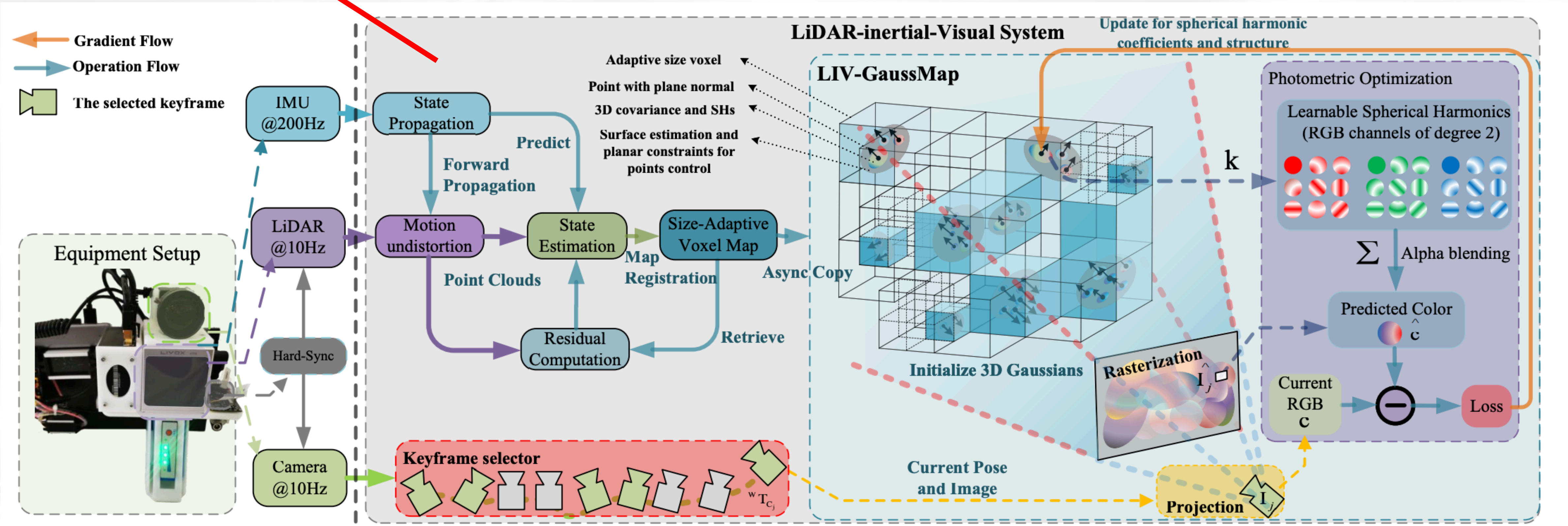
DrivingGaussian: Composite Gaussian Splatting for Surrounding Dynamic Autonomous Driving Scenes

- Incremental static 3D Gaussians and composite dynamic Gaussian graph for complex scene;
- Using **LiDAR prior for Gaussian Splatting** to reconstruct scenes with greater details and maintain panoramic consistency; this is capable of recovering more precise geometry and maintaining better multi-view consistency than utilizing point clouds generated by random initialization or SfM;

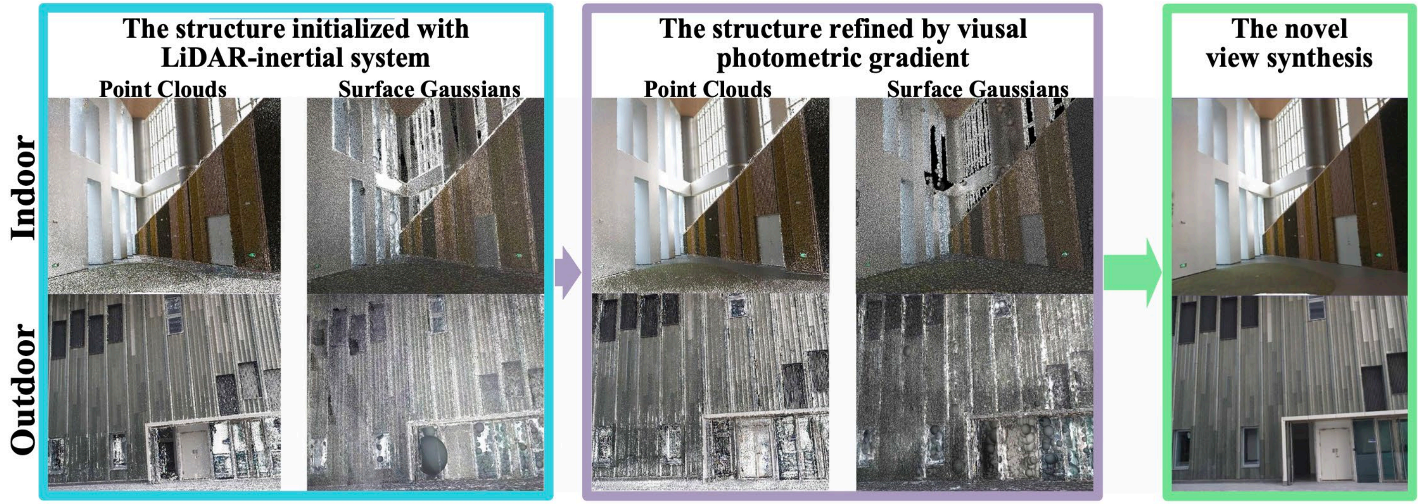


LIV-GaussMap

- The initial poses for surface Gaussian scenes are obtained using a LiDAR-inertial system with size-adaptive voxels. Then, we optimized and refined the Gaussians by visual-derived photometric gradients to optimize the quality and density of LiDAR measurements;
- VoxelMap+3DGS;

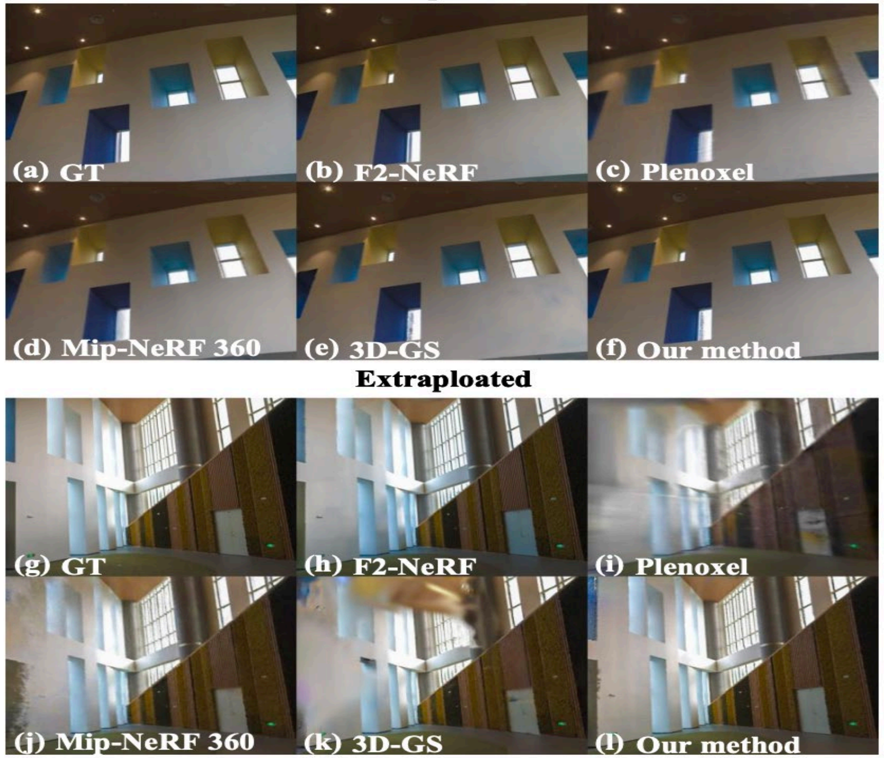


LIV-GaussMap



(a).HKU LSK(indoor)
(b).HKU Main Buliding(outdoor)
(c).HKUST Tower C2(outdoor)
(d).HKUST Maker Space(indoor)
(e).HKUST Tower C2 (indoor)
(f).HKUST Red Bird(outdoor)

(a).The small scale scene (indoor)
(b).The large scale scene (outdoor)



MM-Gaussian: 3D Gaussian-based Multi-modal Fusion for Localization and Reconstruction in Unbounded Scenes

香 港 大 學
THE UNIVERSITY OF HONG KONG

- Solid-state LiDAR+Camera+3DGS to precisely estimate the trajectory and incrementally reconstruct the 3D Gaussian map;
- Relocalization module that utilizes the capability of rendering images from Gaussians;
- All the lidar point in one frame are used to intallize the Gaussian;

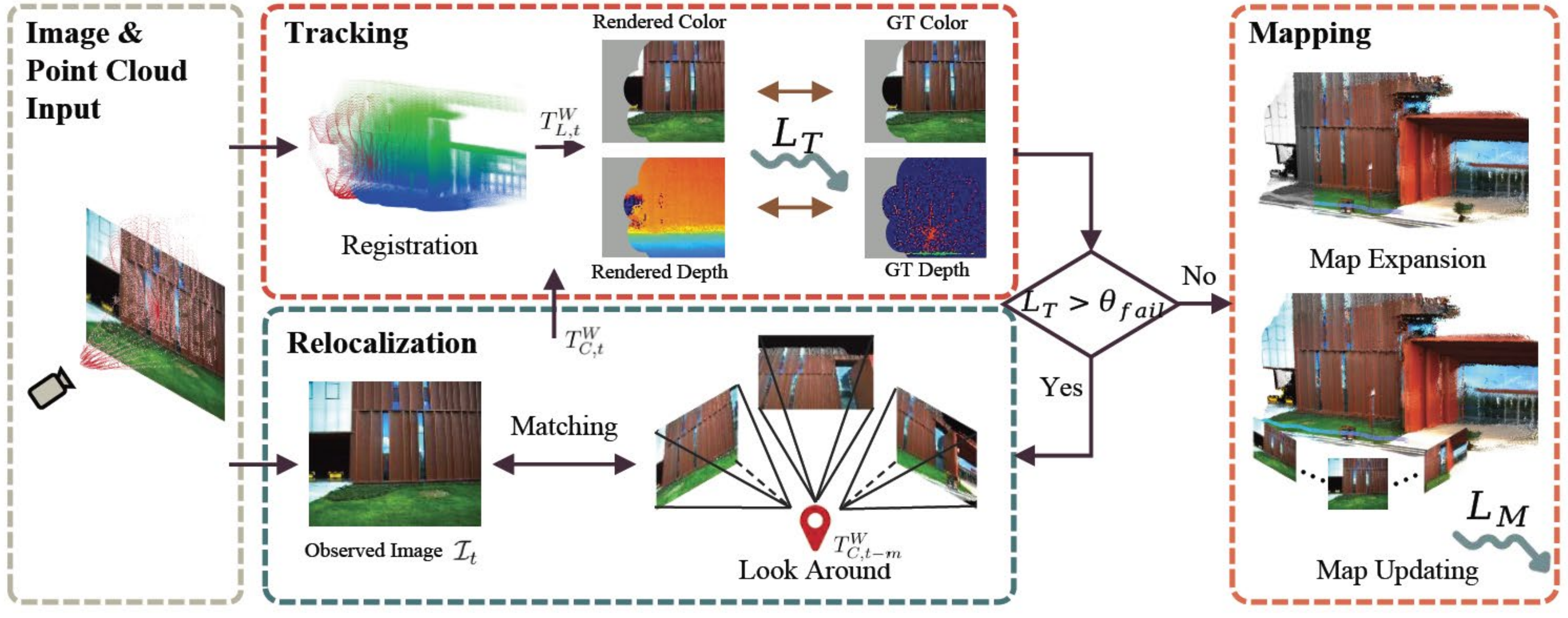
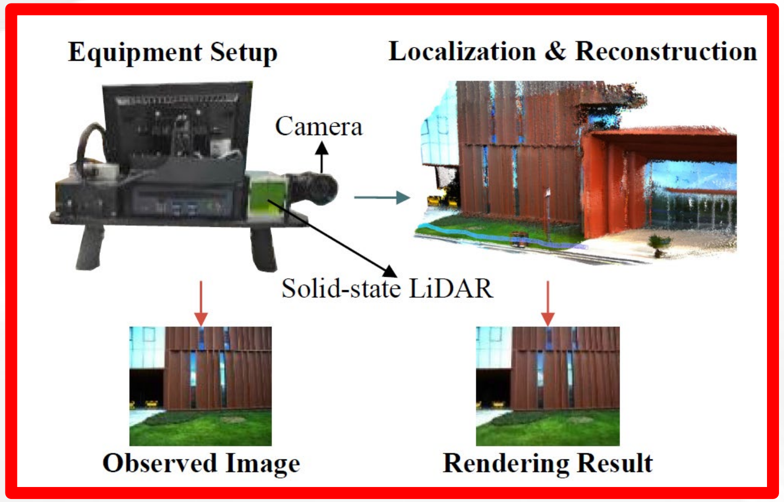


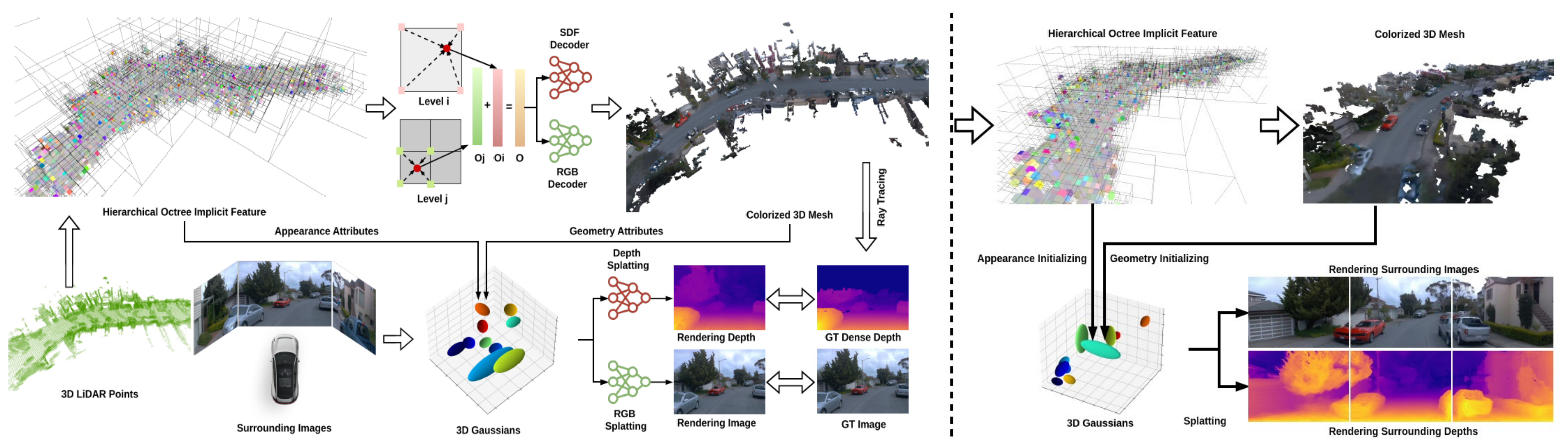
Fig. 2: Overview of MM-Gaussian.



TCLC-GS: Tightly Coupled LiDAR-Camera Gaussian Splatting for Surrounding Autonomous Driving Scenes

香 港 大 學
THE UNIVERSITY OF HONG KONG

- TCLC-GS designs a hybrid explicit (colorized 3D mesh) and implicit (hierarchical octree feature) 3D representation derived from LiDAR-camera data;
    1. First learn and store implicit features in an octree-based hierarchical structure through encoding LiDAR geometries and image colors;
    2. Then initialize 3D Gaussians in alignment with a colorized 3D mesh decoded from the implicit feature volume;

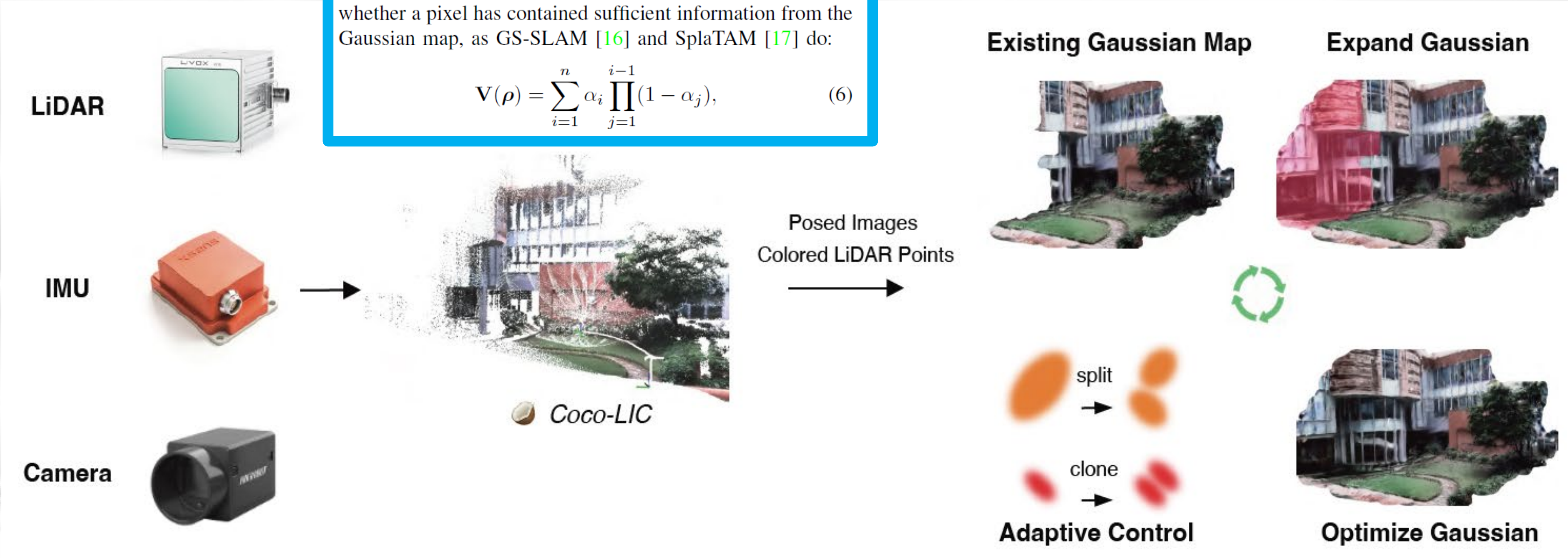


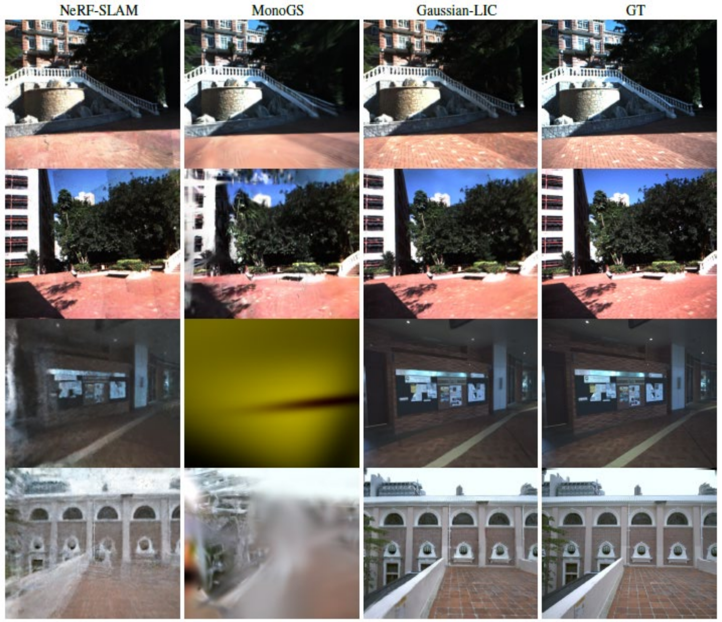
Gaussian-LIC: Photo-realistic LiDAR-Inertial-Camera SLAM with 3D Gaussian Splatting

- Leveraging robust pose estimates from LiDAR-Inertial- Camera odometry, Coco-LIC;
- Initializing 3D Gaussians from colorized LiDAR points and optimize them using differentiable rendering;
- To avoid LiDAR redundancy, first render a silhouette image from the current image view and generate a mask M to select pixels that are not reliable from the current Gaussian map and tend to observe new areas;
- The implementation details is very similar to ours;

Similarly, we also render a silhouette image to determine whether a pixel has contained sufficient information from the Gaussian map, as GS-SLAM [16] and SplaTAM [17] do:

$$\mathbf{V}(\rho) = \sum_{i=1}^{n} \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j), \tag{6}$$





GS-LIVO: Real-Time LiDAR, Inertial, and Visual Multi-sensor Fused Odometry with Gaussian Mapping

香 港 大 學 THE UNIVERSITY OF HONG KONG

- **IESKF**-based LVI-odometry (developed based on Fast-LIVO2) utilizing the **visual measurement model** based on the rendering of Gaussian maps;
- Global Gaussian map representation structured as a spatial **hash-indexed octree**;
- Incrementally maintains a **sliding window of Gaussians** with minimal graphics memory usage, significantly reducing GPU computation and memory consumption by only optimizing the map within the sliding window (**new incremental update strategy**), enabling real-time optimization (NVIDIA Jetson Orin NX1 platform);

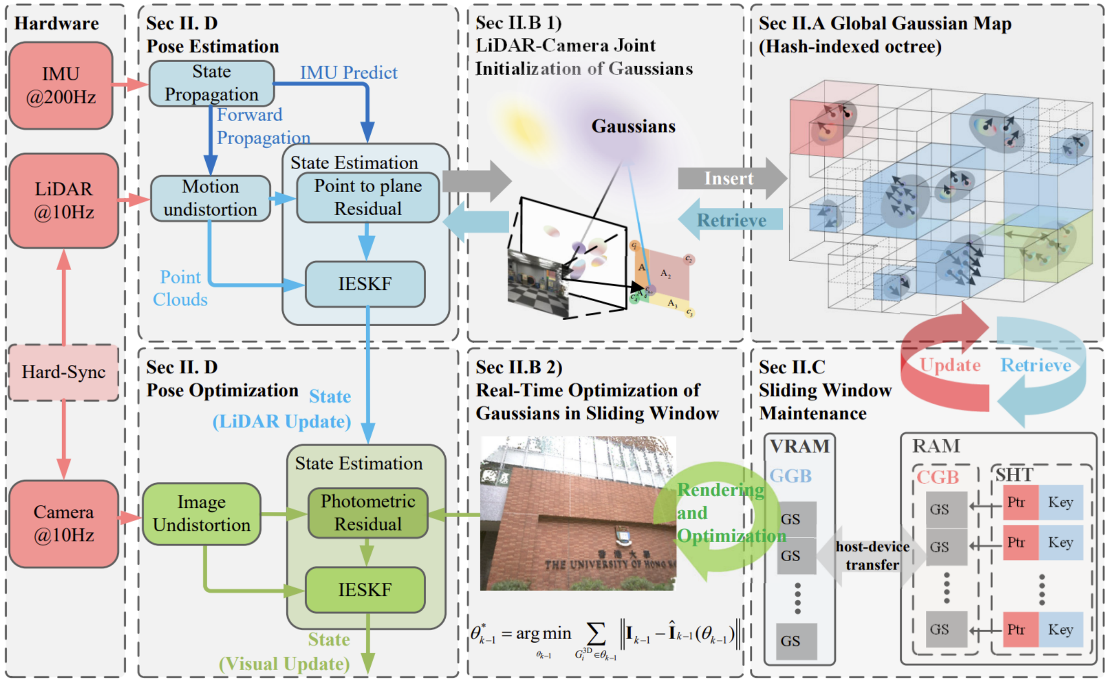


Fig. 2: System overview of GS-LIVO: a real-time LiDAR-Inertial-Visual odometry system with Gaussian Splatting-based mapping. The pipeline performs joint initialization and optimization of Gaussians using multi-sensor data, managed through a hash-indexed octree structure and sliding window mechanism.

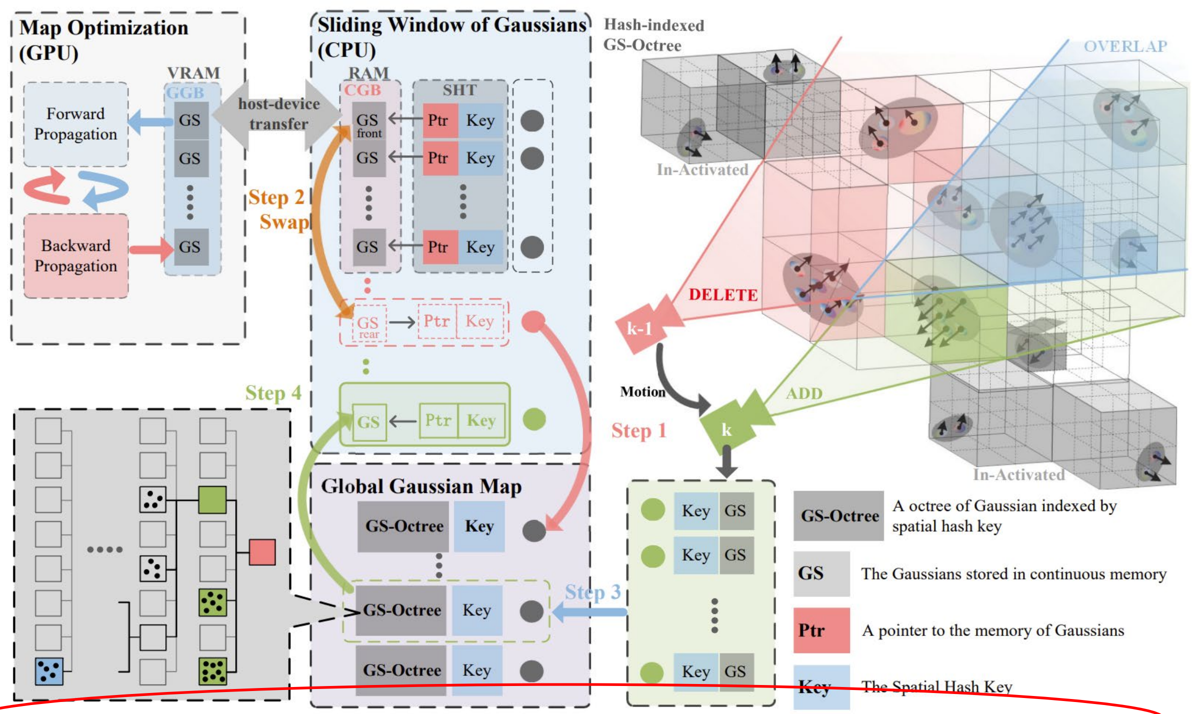


Fig. 3: An overview of the procedures for incrementally updating the sliding window of Gaussians (detailed in Sec.II-C).

Performance of GS-LIVO

香 港 大 學
THE UNIVERSITY OF HONG KONG

TABLE II: Comparative Evaluation for rendering

Sequence	Method	PSNR/dB↑	Dur./s↓	Mem./GB↓
Indoor Datasets				
HKU01 [7]	3D-GS [1]	26.22	2128.6	13.8
	SplaTAM [33]	24.06	292.2	2.6
	MonoGS [51]	23.51	258.0	3.1
	S3GS [60]	✗	✗	✗
	LetsGo [48]	24.51	3231.3	18.1
	GS-LIVO (Ours)	25.34	82.5	2.2
CBD03 [7]	3D-GS [1]	29.54	1873.8	12.5
	SplaTAM [33]	26.85	265.2	4.8
	MonoGS [51]	27.10	278.4	4.6
	S3GS [60]	24.92	3450.7	15.2
	LetsGo [48]	25.51	3573.6	20.4
	GS-LIVO (Ours)	27.52	88.4	2.2
Playground01	3D-GS [1]	29.75	763.4	8.8
	SplaTAM [33]	22.01	292.2	3.4
	MonoGS [51]	21.15	281.0	3.1
	S3GS [60]	20.50	2903.2	9.7
	LetsGo [48]	25.20	3210.3	10.2
	GS-LIVO (Ours)	24.09	48.5	1.5
Playground02	3D-GS [1]	26.54	873.8	7.5
	SplaTAM [33]	24.45	278.4	3.8
	MonoGS [51]	25.93	265.2	4.6
	S3GS [60]	22.24	2957.3	19.1
	LetsGo [48]	23.12	2967.5	18.2
	GS-LIVO (Ours)	25.52	63.4	1.2
Outdoor Datasets				
HKisland03 [9]	3D-GS [1]	17.52	3494.1	21.6
	SplaTAM [33]	12.60	790.0	10.5
	MonoGS [51]	14.22	743.7	12.3
	S3GS [60]	✗	✗	✗
	LetsGo [48]	18.32	2803.3	17.6
	GS-LIVO (Ours)	15.32	82.8	3.2
HKairport01 [9]	3D-GS [1]	16.98	3919.8	22.8
	SplaTAM [33]	12.39	915.2	11.0
	MonoGS [51]	13.87	789.6	13.4
	S3GS [60]	✗	✗	✗
	LetsGo [48]	17.32	3103.3	18.1
	GS-LIVO (Ours)	15.18	93.2	3.1

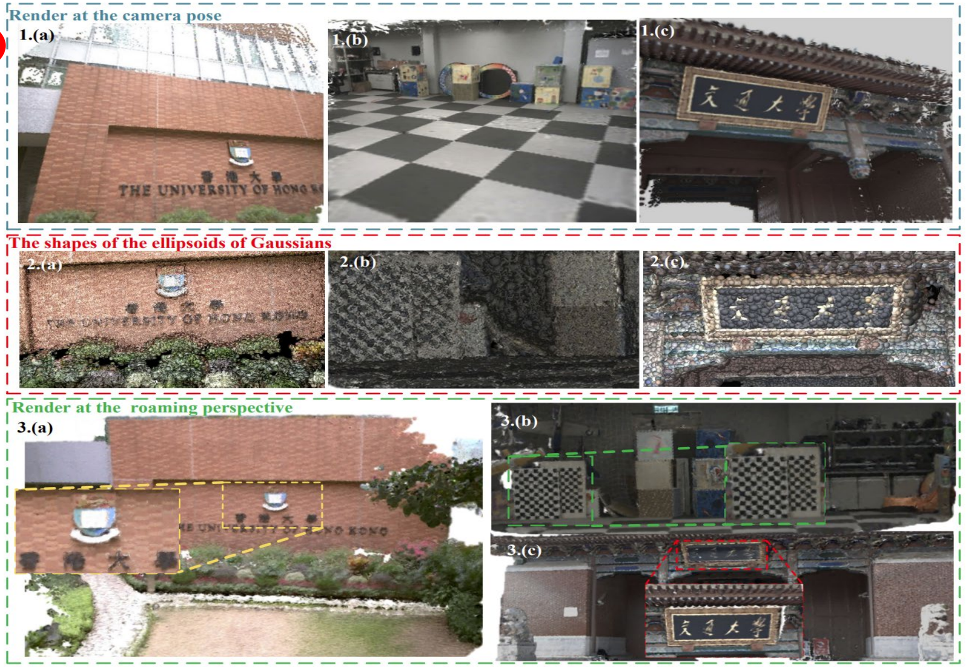


Fig. 5: Mapping results of three distinct real-world scenes (a)- (c). Top row: the rendering results from camera poses. Middle row: the rendering results from roaming perspectives. Bottom row: the shapes of scene Gaussians.

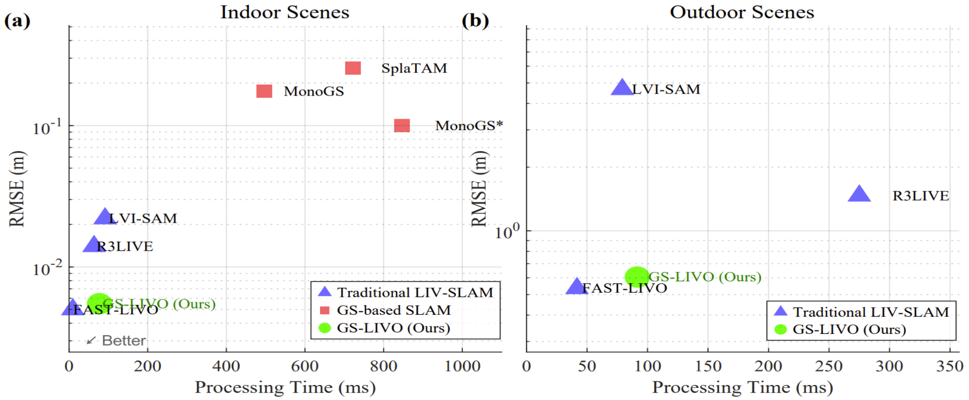
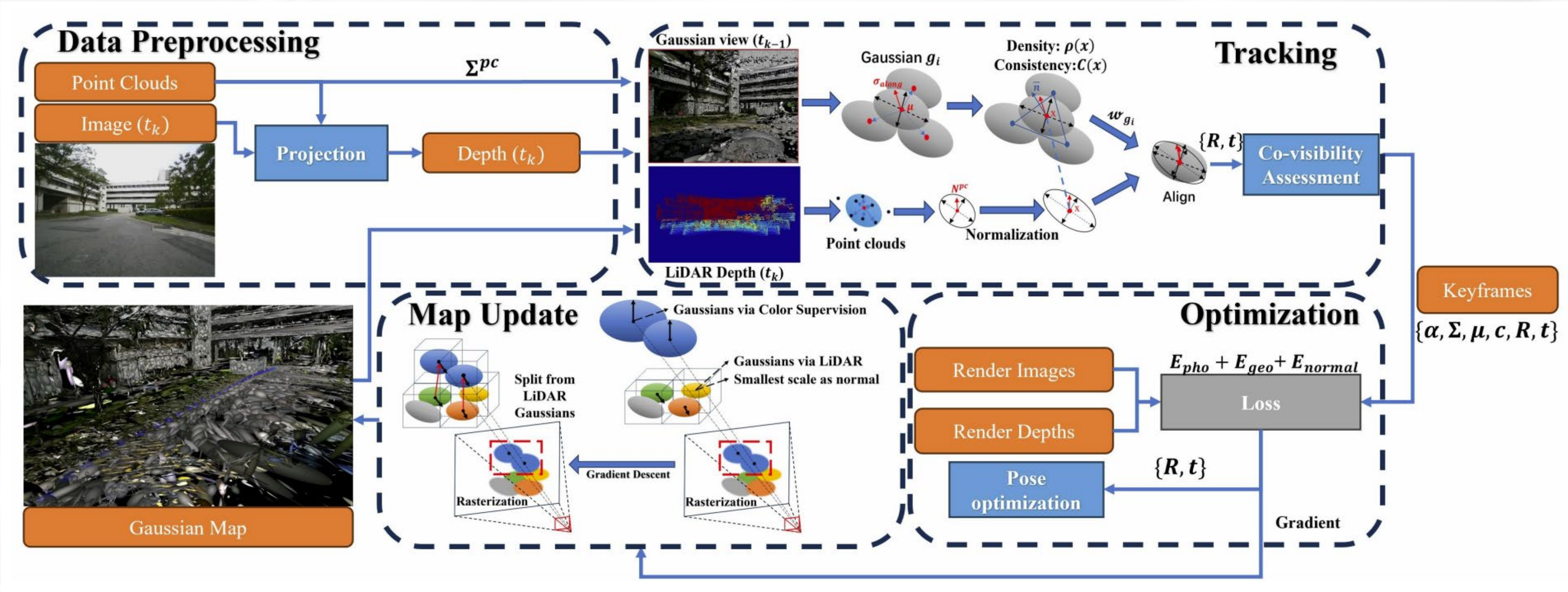


Fig. 6: Performance comparison of different SLAM systems in terms of accuracy (RMSE) and computational efficiency (processing time).

LiV-GS: LiDAR-Vision Integration for 3D Gaussian Splatting SLAM in Outdoor Environments

香 港 大 學
THE UNIVERSITY OF HONG KONG

- The first method that directly **aligns discrete and sparse LiDAR data with continuous differentiable Gaussian maps** in large-scale outdoor scenes;
- Gaussian-LiDAR alignment methods, including a normal direction constraint for stable tracking and a density- and normal-consistency-based weighting mechanism to account for the reliability of different Gaussians;
- Conditional Gaussian distribution constraint for map updates, allowing the propagation of reliable Gaussians with LiDAR priors;



Performance of LiV-GS

香 港 大 學
THE UNIVERSITY OF HONG KONG

TABLE II: Quantitative Analysis for Tracking Accuracy

Methods	cp			garden1			garden2			nyl1			nyl2			loop2		
	$t_{rel}\downarrow$	$r_{rel}\downarrow$	$t_{abs}\downarrow$	$t_{rel}\downarrow$	$r_{rel}\downarrow$	$t_{abs}\downarrow$	$t_{rel}\downarrow$	$r_{rel}\downarrow$	$t_{abs}\downarrow$	$t_{rel}\downarrow$	$r_{rel}\downarrow$	$t_{abs}\downarrow$	$t_{rel}\downarrow$	$r_{rel}\downarrow$	$t_{abs}\downarrow$	$t_{rel}\downarrow$	$r_{rel}\downarrow$	$t_{abs}\downarrow$
NeRF-LOAM	2.943	9.644	5.39	1.182	**0.559**	0.540	1.213	**0.707**	1.076	1.371	**1.140**	3.504	1.343	1.730	17.460	1.442	**2.205**	1.785
HDL-graph-SLAM	1.264	1.553	1.079	1.874	1.603	1.478	**1.186**	0.880	3.154	1.737	1.271	2.266	1.514	1.835	17.638	1.436	2.802	**0.593**
ORB-SLAM3	1.356	1.992	2.865	**1.173**	0.626	0.529	1.212	0.772	1.001	1.342	1.172	19.528	1.333	1.736	23.283	1.403	2.256	0.952
SplaTAM	-	-	2.336	-	-	0.979	-	-	1.221	-	-	12.332	-	-	17.442	-	-	2.692
MonoGS	4.171	3.472	3.440	1.179	0.754	0.664	1.163	0.765	0.708	1.382	1.175	9.595	1.371	1.701	28.553	7.375	5.708	15.357
Gaussian-SLAM	1.249	3.047	1.040	-	-	-	-	-	-	-	-	-	-	-	-	1.399	2.384	1.136
GS-ICP-SLAM	5.471	4.041	6.33	1.249	0.764	2.082	1.824	1.316	5.507	1.662	1.771	23.331	2.101	1.070	23.915	3.236	2.644	13.819
Ours	**0.234**	**1.216**	**0.464**	1.183	0.716	**0.366**	1.236	0.962	**0.679**	**1.240**	1.307	**0.580**	**1.106**	**1.369**	**0.771**	**1.393**	2.239	0.843

TABLE III: Quantitative Analysis for Rendering [SSIM↑ PSNR↑ LPIPS↓]

Methods	cp			garden1			garden2			nyl1			nyl2			loop2		
	SSIM	PSNR	LPIPS	SSIM	PSNR	LPIPS	SSIM	PSNR	LPIPS	SSIM	PSNR	LPIPS	SSIM	PSNR	LPIPS	SSIM	PSNR	LPIPS
3DGS	0.718	21.95	0.617	0.598	20.495	0.557	0.662	20.402	0.536	0.726	20.224	0.575	0.571	18.040	0.657	0.594	17.794	0.620
GS-ICP-SLAM	0.552	17.336	0.772	0.511	15.782	0.661	0.417	13.946	0.606	-	-	-	0.374	11.533	0.882	0.492	12.335	0.771
NeRF++	0.566	19.226	0.698	0.529	15.772	0.685	0.475	13.736	0.675	0.574	13.662	0.726	0.320	10.130	0.879	-	-	-
SplaTAM (Odom)	0.629	18.772	0.585	0.515	18.788	0.545	0.540	18.402	0.569	0.688	19.770	0.574	0.480	17.212	0.644	0.475	13.736	0.675
SplaTAM (GT)	0.535	17.501	0.691	0.498	17.936	0.607	0.497	18.402	0.533	0.597	18.960	0.520	0.432	16.933	0.721	0.438	11.225	0.753
MonoGS (Odom)	0.655	20.962	0.444	0.607	23.124	0.474	0.649	20.627	0.535	0.813	**22.800**	0.373	0.560	17.812	0.665	0.578	17.993	0.503
MonoGS (GT)	0.733	22.118	0.364	0.717	**23.723**	0.522	0.598	20.112	0.552	0.748	21.969	0.415	0.511	17.006	0.738	0.505	17.567	0.616
Gaussian-SLAM (Odom)	0.665	20.924	0.595	-	-	-	-	-	-	-	-	-	-	-	-	0.534	16.509	0.656
Gaussian-SLAM (GT)	0.647	21.026	0.688	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Ours (Odom)	**0.775**	22.274	0.336	**0.773**	23.569	**0.386**	**0.772**	**22.141**	**0.336**	**0.833**	22.559	**0.274**	0.725	**20.835**	0.559	**0.686**	18.037	0.556
Ours (GT)	0.763	**22.368**	**0.319**	0.721	23.353	0.404	0.758	21.991	0.348	0.799	22.136	0.308	**0.744**	20.630	**0.548**	0.631	**18.472**	**0.417**

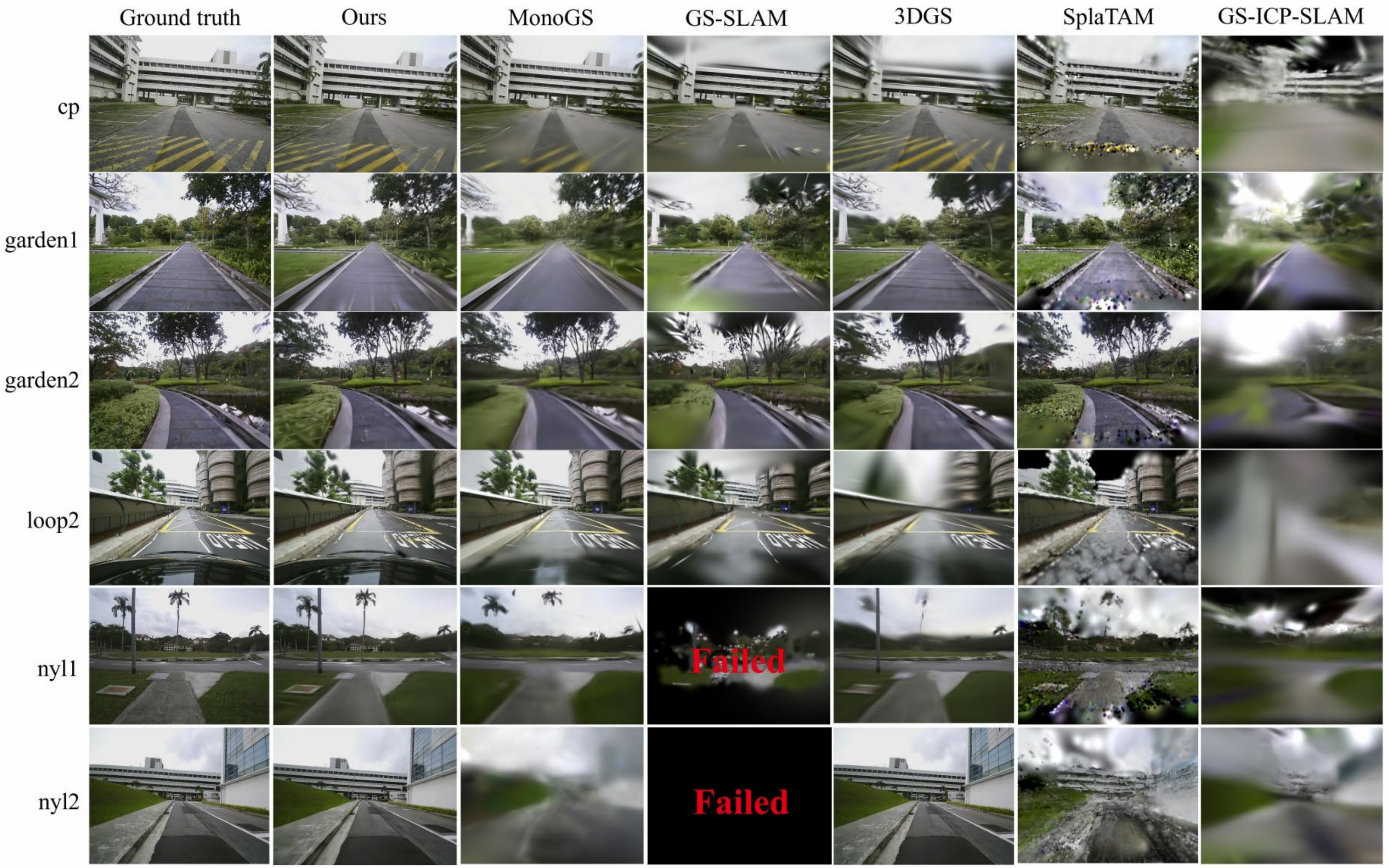
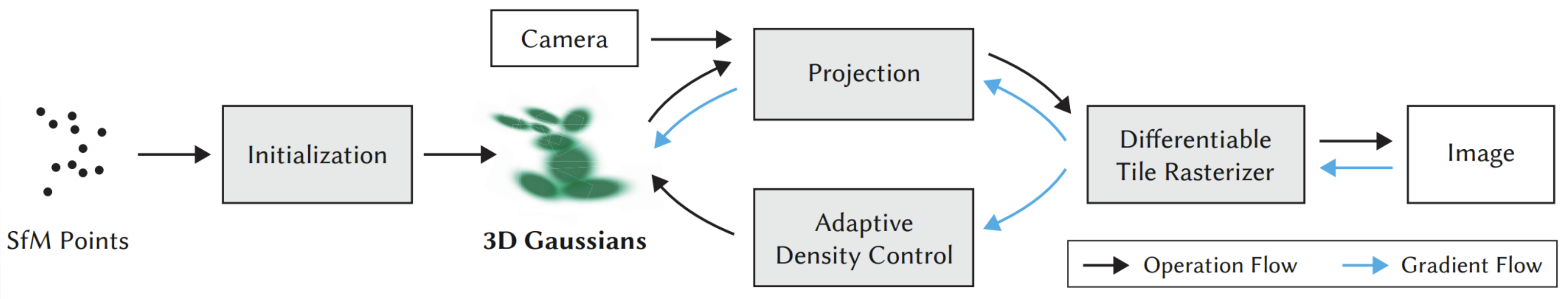
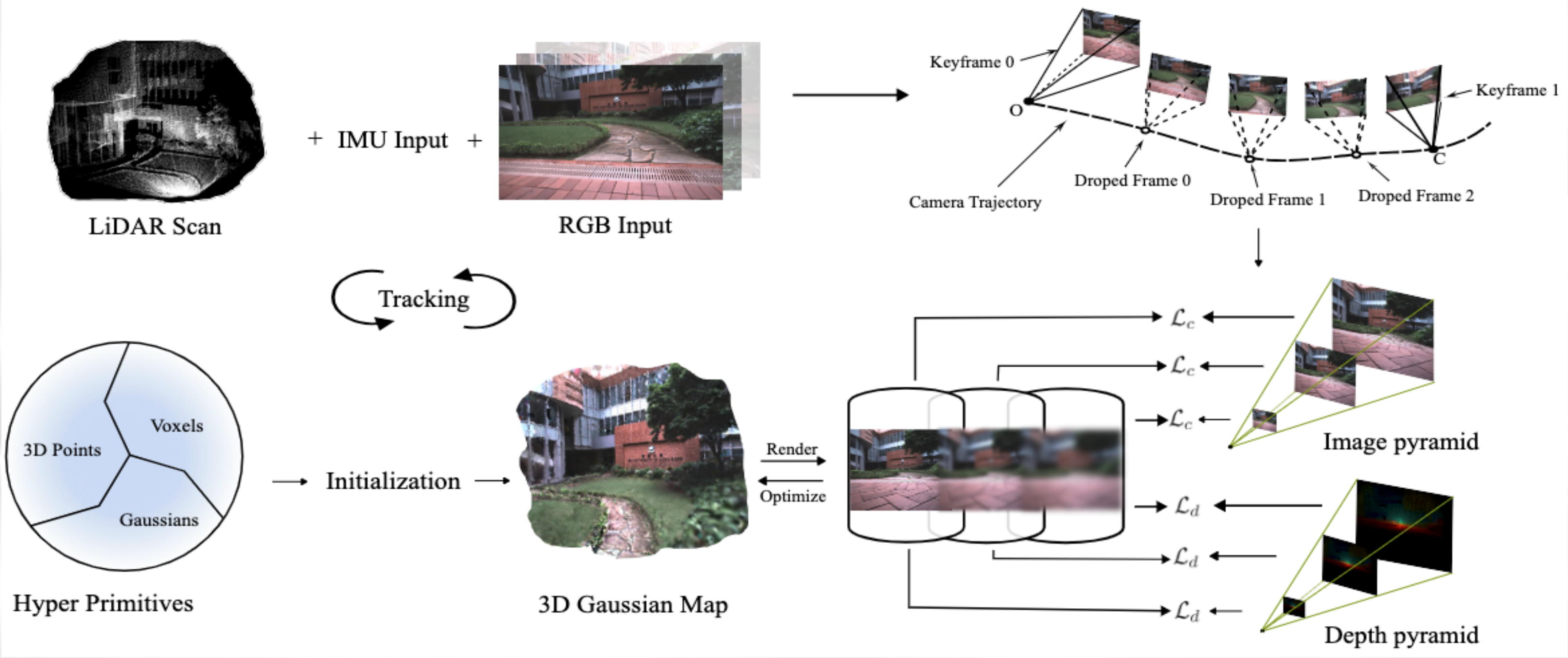


Fig. 6: **Comparison of Rendering Results.**

LVI-GS: Tightly-coupled LiDAR-Visual-Inertial SLAM using 3D Gaussian Splatting

香 港 大 學
THE UNIVERSITY OF HONG KONG





$$f^{3D}(p) = \text{sigmoid}(o)\exp\left(-\frac{1}{2}(p-\mu)^T\Sigma^{-1}(p-\mu)\right)$$

$$\Sigma = RSS^TR^T$$

$$\mu^{2D} = K((E\mu)/(E\mu)_z),$$
$$\Sigma^{2D} = JE\Sigma E^TJ^T,$$

Project to the image plane

$$C_{\text{pix}} = \sum_{i\in V} c_i f^{2D}_{i,\text{pix}} \prod_{j=1}^{i-1}(1 - f^{2D}_{j,\text{pix}})$$

$$\boxed{D_{\text{pix}} = \sum_{i\in S} z_i f^{2D}_{i,\text{pix}} \prod_{j}^{i-1}(1 - f^{2D}_{j,\text{pix}})}$$

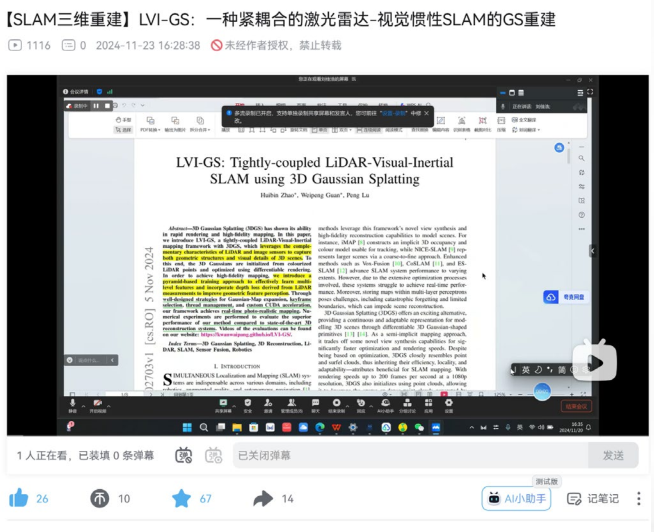
$$\mathcal{L}_c = (1-\lambda)\mathcal{L}_1(I, \mathcal{C}(\mathcal{G}, T_c)) + \lambda\mathcal{L}_{ssim}$$

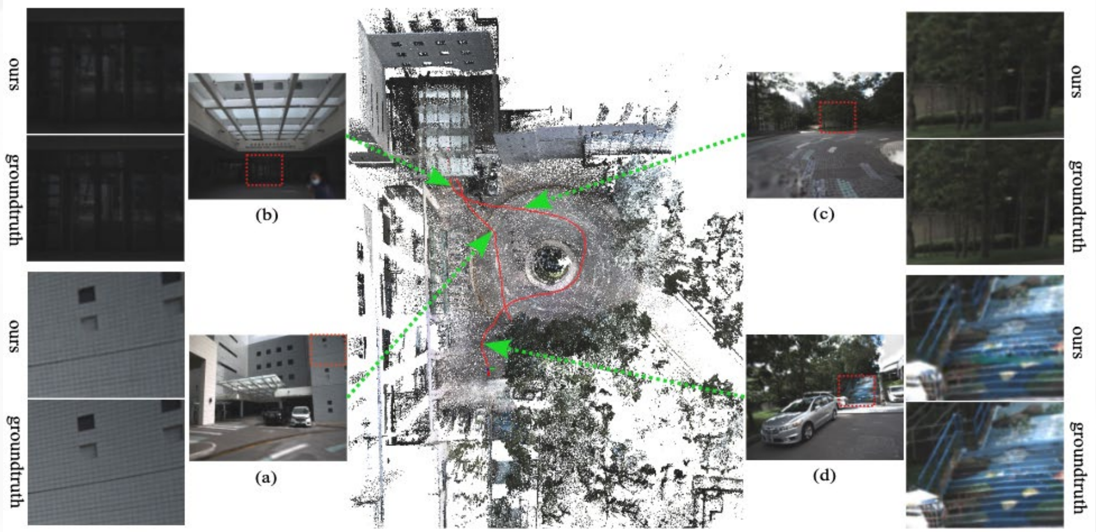
$$\mathcal{L} = \mathcal{L}_c + \lambda_d\mathcal{L}_d$$

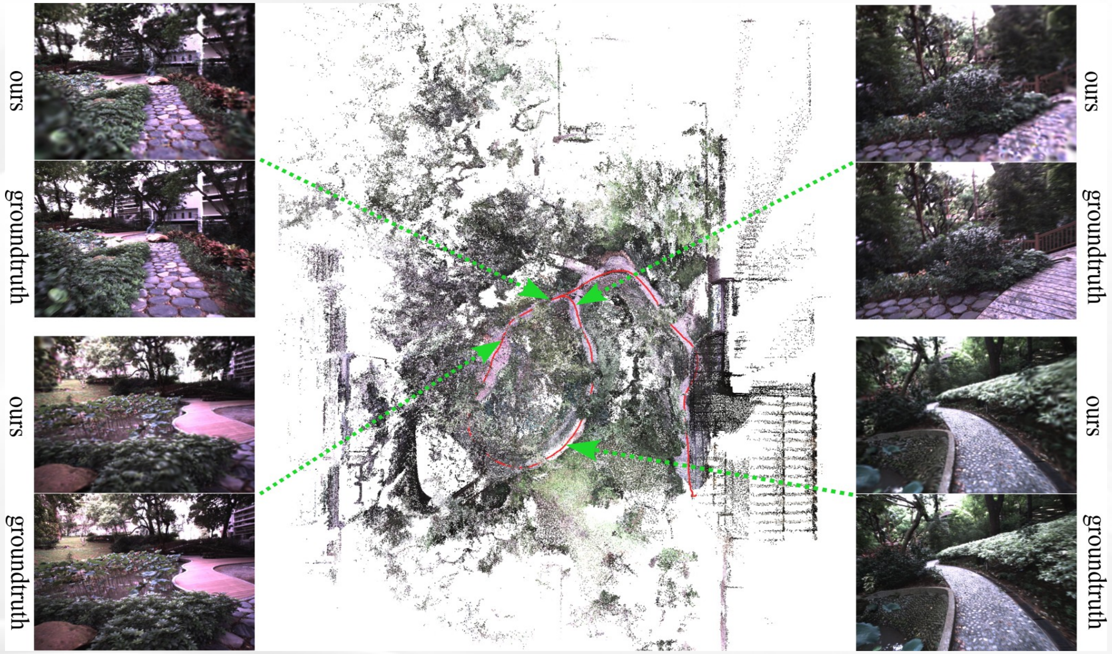
$$\mathcal{L}_d = \sum \|\mathcal{D} - \mathcal{D}_{lidar}\|$$

Our LVI-GS

- Link: https://kwanwaipang.github.io/LVI-GS/







Splat-LOAM: Gaussian Splatting LiDAR Odometry and Mapping

香 港 大 學
THE UNIVERSITY OF HONG KONG

- LiDAR odometry and mapping pipeline that exclusively relies on **2D Gaussian** primitives for its scene representation;
- Employing **spherical projection** to encode LiDAR measurements into an image-like representation so that it can be used to guide the Gaussian primitives optimization;
- Rely on keyframing to optimize local maps, and Frame-To-Model registration for tracking;

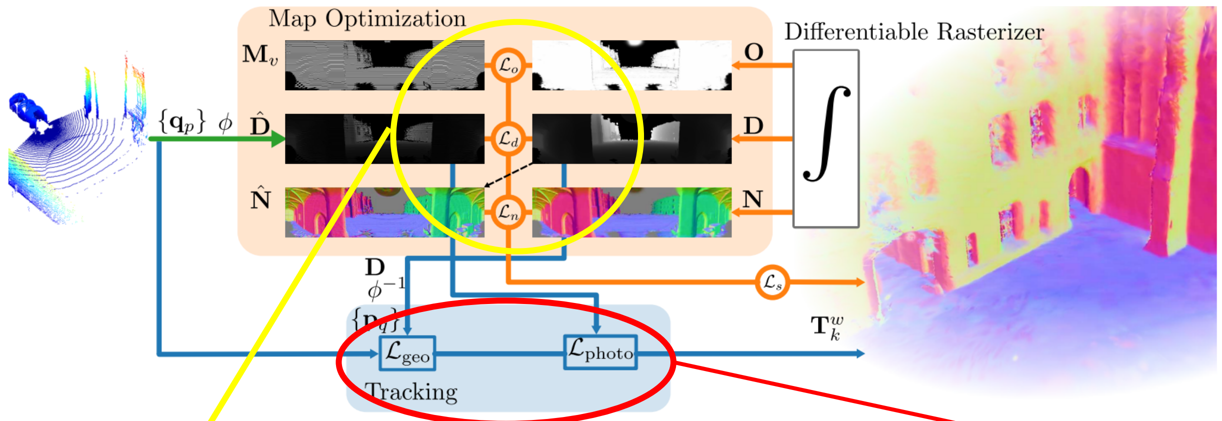


Figure 2. **Splat-LOAM Overview.** Given a LiDAR point cloud, we leverage the spherical projection to generate an image-like representation. Moreover, using an ad-hoc differentiable rasterizer, we guide the optimization for structural parameters of 2D Gaussians. The underlying representation is concurrently used to incrementally register new measurements.

, we integrate them using α-blending from front to back to obtain a range d, normal $\mathbf{n}$ and opacity o values, as follows:

$$d = \sum_{i=1}^{T} o_i \mathcal{G}_i d_i \prod_{j=1}^{i-1} (1 - o_j \mathcal{G}_j) \tag{5}$$

$$\mathbf{n} = \sum_{i=1}^{T} o_i \mathcal{G}_i \mathbf{t}_{n_i} \prod_{j=1}^{i-1} (1 - o_j \mathcal{G}_j) \tag{6}$$

$$o = \sum_{i=1}^{T} o_i \mathcal{G}_i \prod_{j=1}^{i-1} (1 - o_j \mathcal{G}_j) \tag{7}$$

To optimize the geometric consistency of the local model, we employ a loss term that minimizes the L_1 error:

$$\mathcal{L}_d = \sum_{\mathbf{u} \in \mathbf{M}_v} \rho_d \|\mathbf{D}(\mathbf{u}, \mathbf{x}) - \hat{\mathbf{D}}(\mathbf{u})\|, \tag{14}$$

where ρ_d is a weight function dependent on the measurement's range. In addition, we employ a self-regularization term to align the splat's normals to the surface normals $\mathbf{N}$ estimated by the gradients of the range map $\mathbf{D}$ [15]:

$$\mathcal{L}_n = \sum_{\mathbf{u} \in \mathbf{M}_v} 1 - \mathbf{n}^T \mathbf{N}(\mathbf{D}(\mathbf{u}, \mathbf{x})) \tag{15}$$

Furthermore, to promote the expansion of splats over uniform surfaces, we introduce an additional term that operates on the opacity channel of the rasterized images. Specifically, we drive the splats to cover the areas of the image containing valid measurements by correlating the opacity image $\mathbf{O}$ with the valid mask $\mathbf{M}_v$.

$$\mathcal{L}_o = \sum_{\mathbf{u} \in \mathbf{M}_v} -\log\left(\mathbf{O}(\mathbf{u}, \mathbf{x})\right). \tag{16}$$

$$\mathcal{L}_{\text{geo}} = \sum_{p,q \in \{a\}} \rho_{\text{Huber}}\left((\mathbf{T}_w^k \mathbf{n}_{l_q})^T (\mathbf{T}_w^k \mathbf{q}_p - \mathbf{p}_{q_i})\right), \tag{20}$$

$$\mathcal{L}_{\text{photo}} = \sum_{\mathbf{u}} \left\| \rho_{\text{Huber}}\left(\mathbf{D}(\mathbf{u}) - \hat{\mathbf{D}}\underbrace{\left(\phi(\mathbf{T}_w^k \phi^{-1}(\mathbf{u}, \hat{d}))\right)}_{\mathbf{u}'}\right)\right\|^2. \tag{21}$$

Performance of Splat-LOAM

香 港 大 學
THE UNIVERSITY OF HONG KONG

- The evaluation of pose tracking is no obvious, without the ATE or quantitative comparison;
- While the mapping performance seems to be better than some baseline, but without comparison with the LVI-odometry series (like R3LIVE, FAST-LVIO2);
- Nonetheless, the inspiration of this work is impressive;

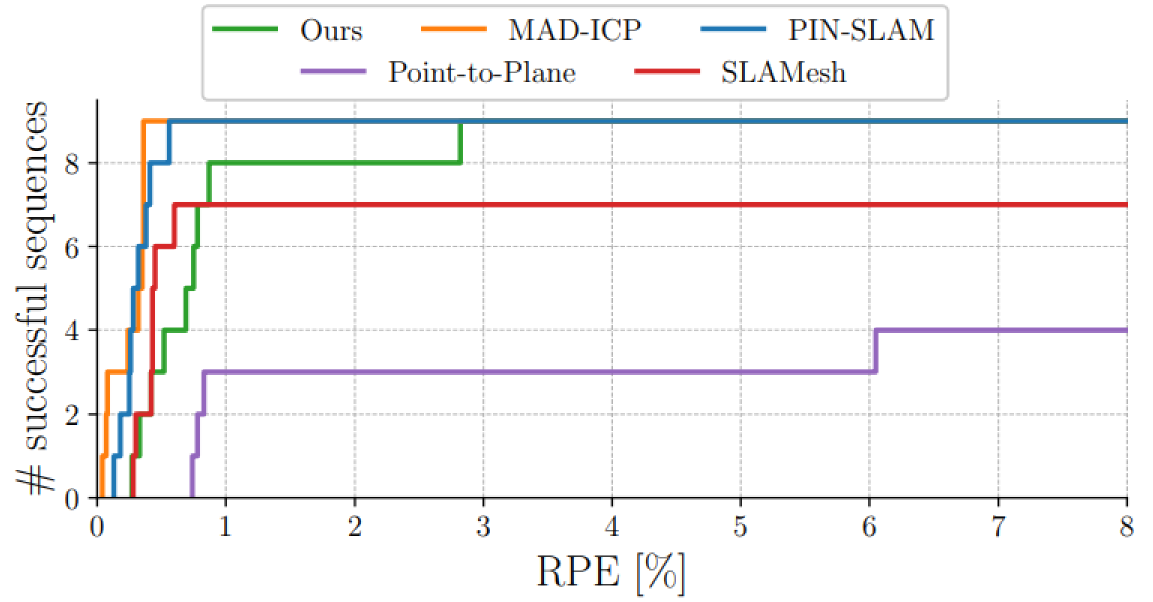


Figure 4. **RPE evaluation.** Number of successful sequences across RPE thresholds. It includes the sequences of Newer College [49], VBR [4], Oxford Spires [41] and Mai City [43].

Dataset	Newer College[50]				Oxford Spires[41]											
	quad-easy				keble-college02				bodleian-library-02				observatory-01			
Approach	Acc↓	Com↓	C-l1↓	F-score↑	Acc↓	Com↓	C-l1↓	F-score↑	Acc↓	Com↓	C-l1↓	F-score↑	Acc↓	Com↓	C-l1↓	F-score↑
OpenVDB[27]	11.45	4.38	7.92	88.85	7.46	**6.92**	7.19	91.74	10.34	4.68	**7.51**	89.68	9.58	**9.60**	9.59	86.16
VoxBlox[29]	20.36	12.64	16.5	64.63	15.81	14.25	15.03	71.63	18.92	11.56	15.24	58.77	15.09	15.15	15.12	70.45
N^3-Mapping[39]	**6.32**	9.75	8.04	94.54	6.21	7.82	**7.01**	93.47	**10.16**	5.62	7.89	**90.36**	**8.27**	10.44	**9.35**	**87.94**
PIN-SLAM[30]	15.28	10.5	12.89	88.05	13.73	9.94	11.83	79.65	14.34	7.14	10.74	82.71	16.91	12.07	14.49	72.31
Ours	6.64	**4.09**	**5.37**	**96.74**	**6.18**	8.69	7.43	**94.41**	10.87	**4.33**	7.6	90.09	9.35	11.76	10.56	83.04

Table 1. **Reconstruction quality evaluation.** The pipelines were run with ground-truth poses. Voxel size is set to 20 cm and F-score is computed with a 20 cm error threshold. Splat-LOAM yields competitive mapping performance on both the Newer College[50] and Oxford Spires[41] datasets and outperforms most competitive approaches.

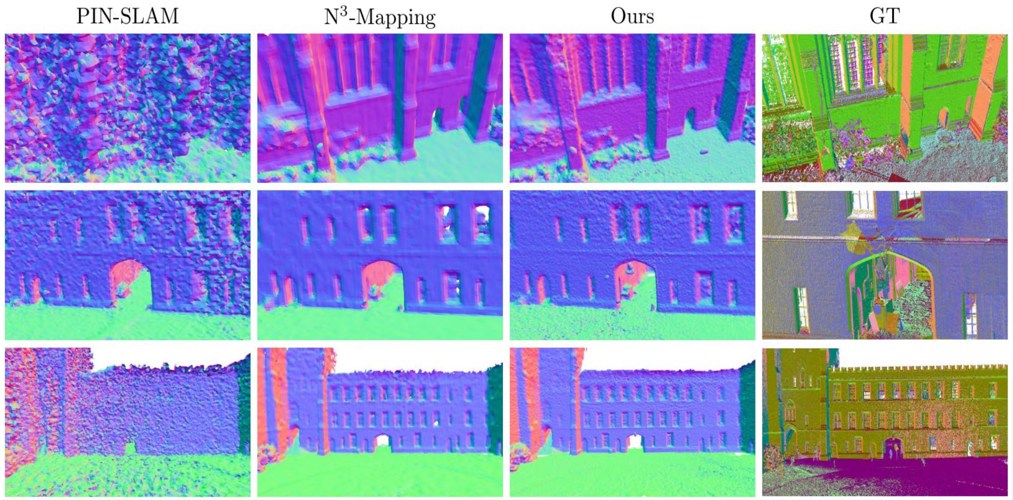
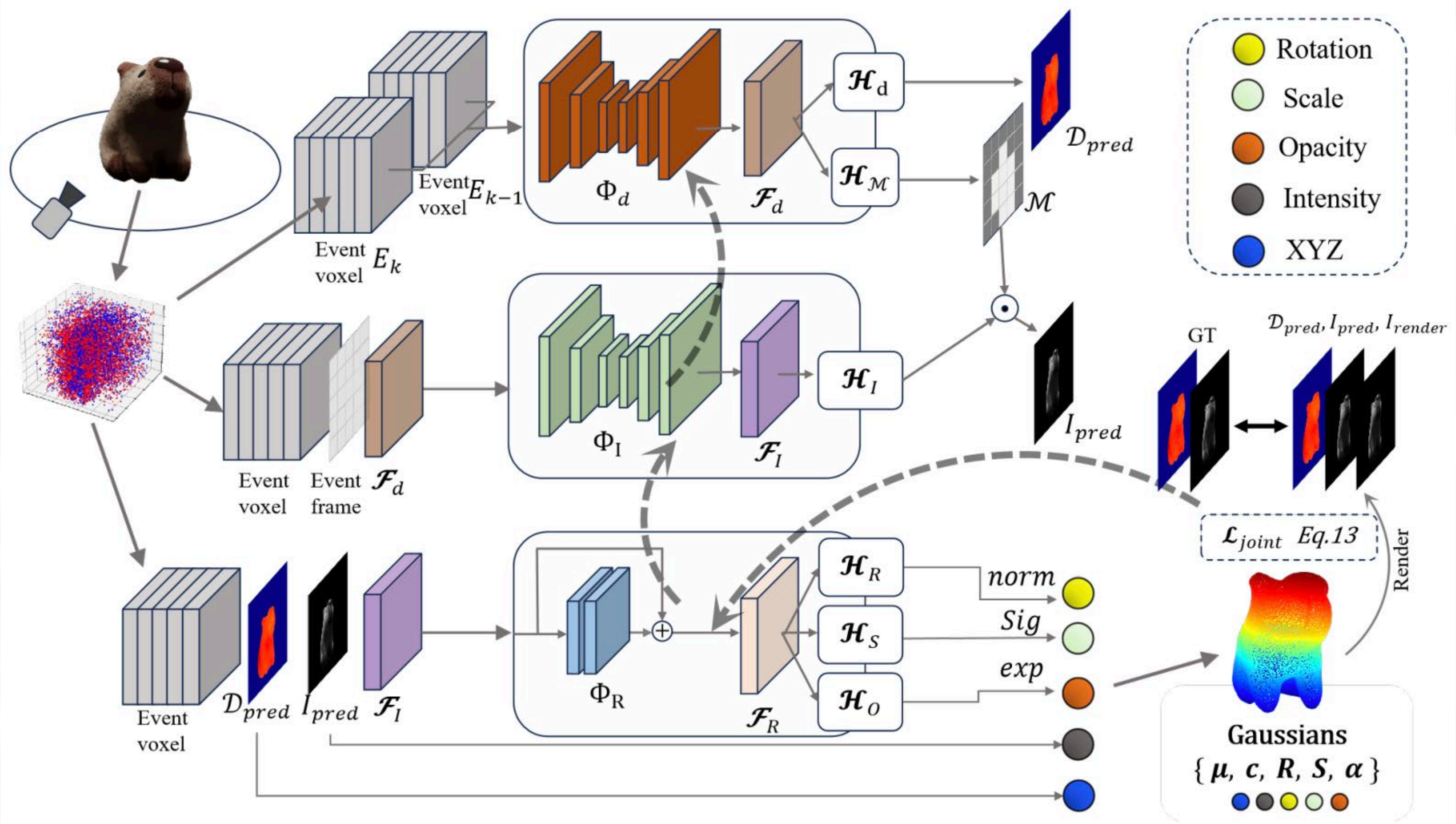


Figure 6. **Comparison of Mesh Reconstruction.** The figure shows reconstruction results for *quad-easy* sequence from the newer college dataset. Our method recovers a geometry with much higher data fidelity. PIN-SLAM lacks many details and exhibits a large level of noise. N^3-Mapping performs more similar to ours, but oversmoothes fine geometric details.

Event-based 3DGS

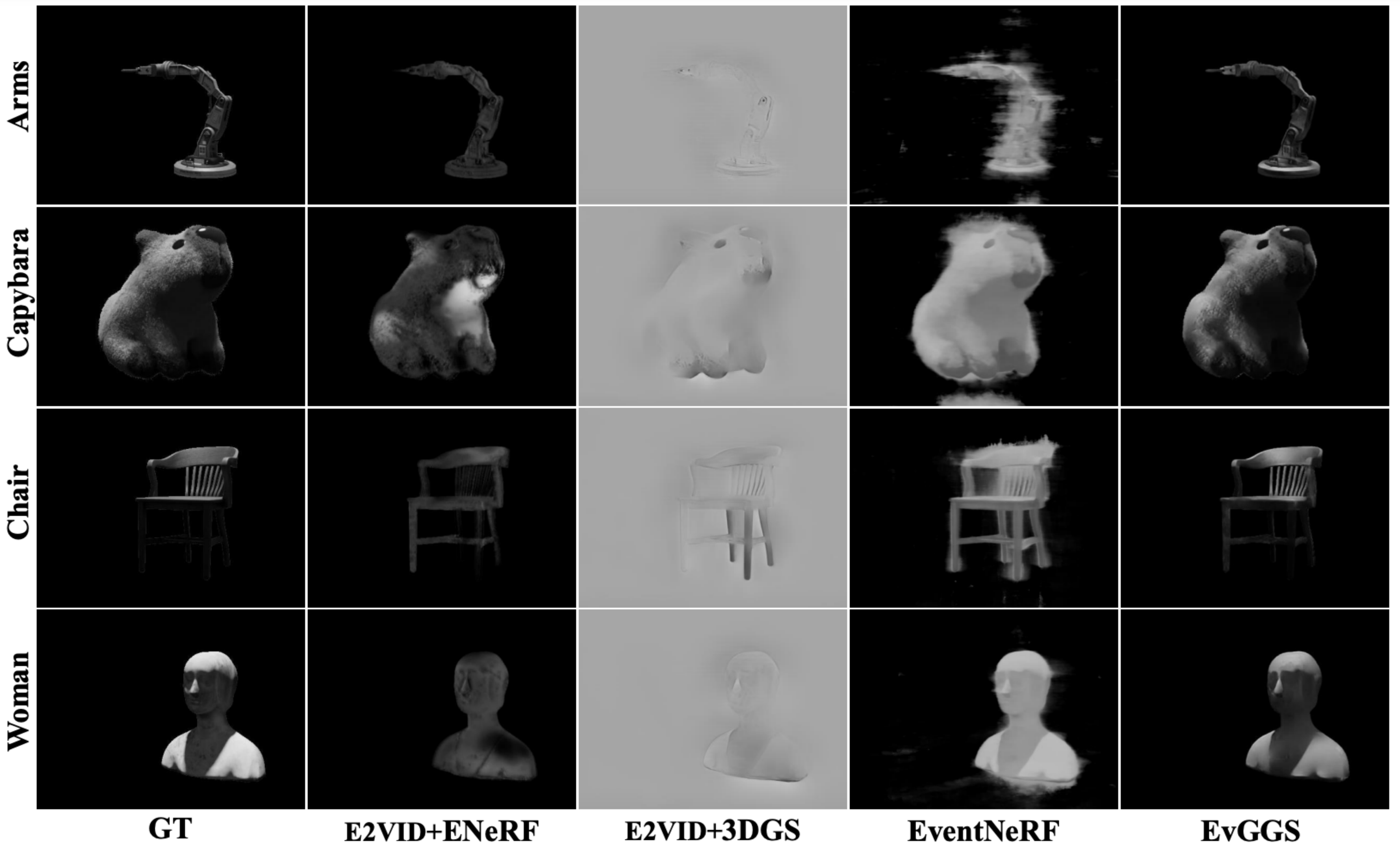
EvGGS: A Collaborative Learning Framework for Event-based Generalizable Gaussian Splatting

- Reconstructing scenes as 3D Gaussians from only event input in a feedforward manner;
- This framework includes a depth estimation module, an intensity reconstruction module, and a Gaussian regression module;
- Given a 360-degree event stream and target viewpoints, employ two submodules to extract the depth and intensity information, which serve as the 3D position and color maps;



EvGGS: A Collaborative Learning Framework for Event-based Generalizable Gaussian Splatting

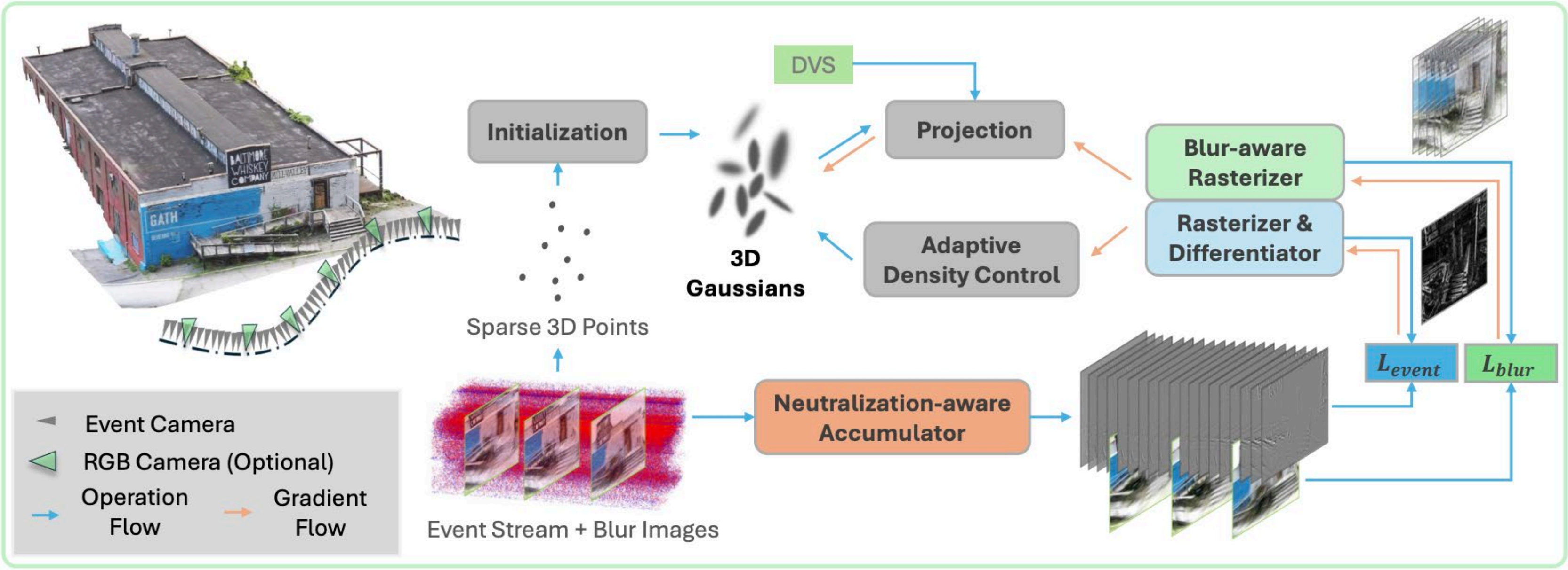
香港大學
THE UNIVERSITY OF HONG KONG



Event3DGS: Event-based 3D Gaussian Splatting for Fast Egomotion

香 港 大 學 THE UNIVERSITY OF HONG KONG

- The first work to learn Gaussian Splatting solely from raw event streams;
- Within radiance field rendering, the **inherent capability of event cameras** to precisely capture scene information at high temporal resolutions seamlessly aligns with the demands posed by radiance field rendering in fast ego motion scenarios;
- It can reconstruct 3D structures under fast ego-motion through "**just saying**", **without any evaluations** or even using the real event data for evaluation;
- New event slicing strategy and handle the uniform radiance region where do not trigger events;
- Using **colorful event** (or add the blur images) as input, SfM-event for initialization, RGB as target;



Event3DGS: Event-based 3D Gaussian Splatting for Fast Egomotion

香 港 大 學
THE UNIVERSITY OF HONG KONG

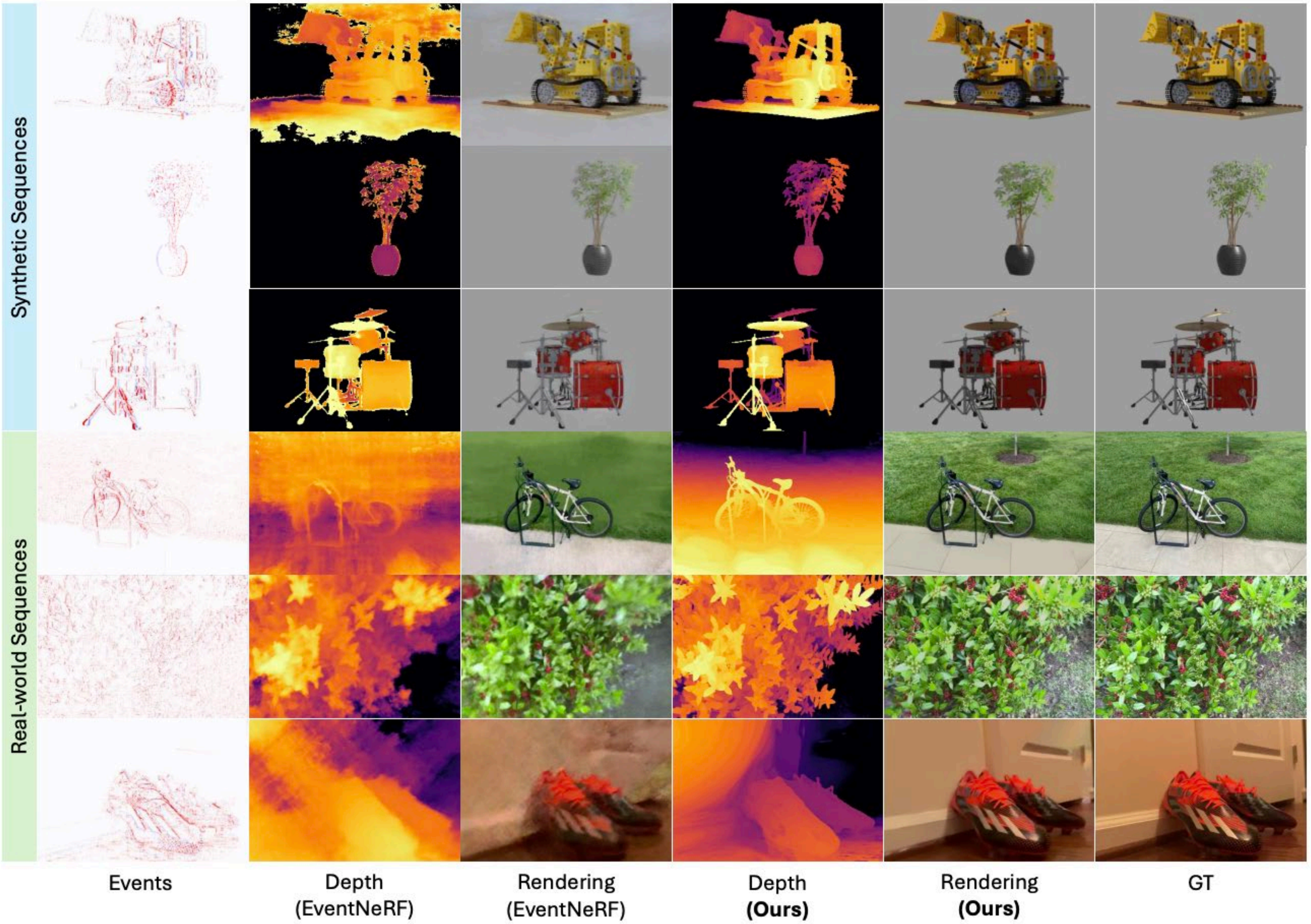


Figure 5: Qualitative comparison on synthetic and real-world sequences (event-only).

Synthetic Sequences Table 1 demonstrates that our Event3DGS method consistently outperforms both baselines across almost all synthetic scenes in all metrics. On average, our method achieves a $+2.61dB$ higher PSNR, a 2.15% higher SSIM ,and a 50% lower LPIPS. Notably, our training time is significantly shorter than both baselines (see Sec. 4.3).

Scene	E2VID[54] + 3DGS[2]			EventNeRF[42]			**Event3DGS (event-only)**		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Drums	16.52	0.74	0.24	27.43	0.91	0.07	29.37	0.94	0.04
Lego	16.11	0.75	0.23	25.84	0.89	0.13	29.57	0.93	0.05
Chair	20.64	0.87	0.13	30.62	0.94	0.05	31.59	0.95	0.03
Ficus	23.33	0.88	0.12	31.94	0.94	0.05	32.47	0.95	0.03
Mic	20.47	0.89	0.14	31.78	0.96	0.03	33.83	0.98	0.02
Hotdog	22.45	0.90	0.12	30.26	0.94	0.04	32.35	0.96	0.03
Materials	18.62	0.85	0.15	24.10	0.94	0.07	31.03	0.96	0.03
Average	19.73	0.84	0.16	28.85	0.93	0.06	**31.46**	**0.95**	**0.03**

Table 1: Quantitative comparison on synthetic event-sequences (event-only)

Event data with high temporal resolution can provide supervision signals with sharp scene structure, allowing 3D gaussian splatting (3DGS) to perform fine-grained reconstruction of scene structure under fast egomotion The multi-view consistency of event sequence guarantee the learnable Gaussians to continuously converge to the ground truth geometric structure and logarithmic color field of the scene during optimization. Our event rendering loss $\mathcal{L}_{event}(t_s, t_e)$ compares the recorded events with the differential signal generated by adjacent view renderings according to the event formation model. Following [2], it primarily comprises two components: the $\mathcal{L}_1$ loss, which measures the absolute log-radiance change difference at each pixel, and the structural dissimilarity loss $\mathcal{L}_{DSSIM}$ [50], which accounts for the structural information between adjacent pixels. We define them as follows:

$$\mathcal{L}_1(t_s, t_e) = \left\| \frac{\mathbf{F} \odot (\log \tilde{\mathbf{C}}(t_e) - \log \tilde{\mathbf{C}}(t_s))}{g} - \mathbf{F} \odot \mathbf{E}(\mathbf{t_s}, \mathbf{t_e}) \right\|_1 \quad (4)$$

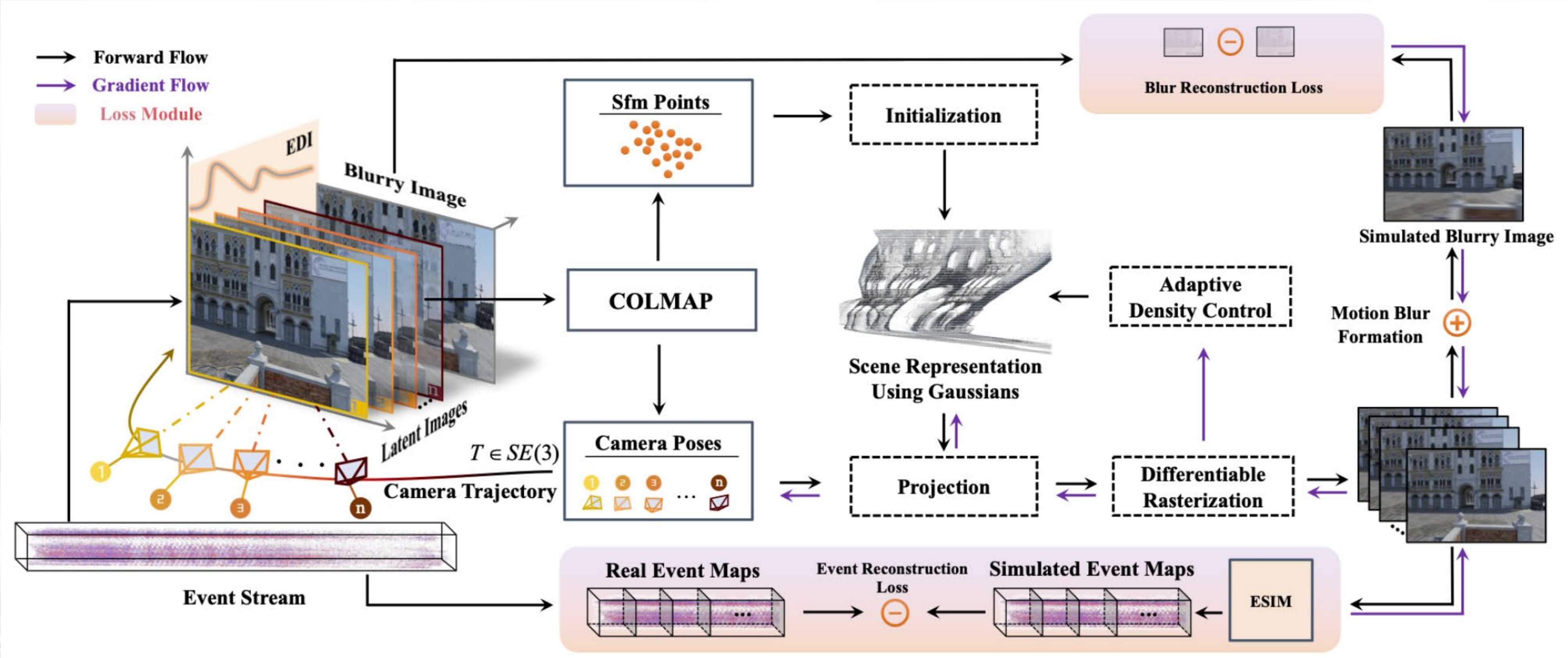
$$\mathcal{L}_{DSSIM}(t_s, t_e) = DSSIM\left(\frac{\mathbf{F} \odot (\log \tilde{\mathbf{C}}(t_e) - \log \tilde{\mathbf{C}}(t_s))}{g}, \mathbf{F} \odot \mathbf{E}(\mathbf{t_s}, \mathbf{t_e})\right) \quad (5)$$

where $\tilde{\mathbf{C}}(t)$ denotes the 2D rendering under the view at time t, g is a gamma correction value initialized to 2.2 in our experiments which can be adjust in appearance refinement stage (see Sec. 3.5), $\mathbf{E}$ represents the accumulation of all event polarities triggered within the field of view (FOV), $\mathbf{F}$ is the RGGB Bayer filter [42], which only apply for colour events. The total loss can be written as, we set λ_{DSSIM} to 0.2 in our experiments:

$$\mathcal{L}_{event} = (1 - \lambda_{DSSIM})\mathcal{L}_1 + \lambda_{DSSIM}\mathcal{L}_{DSSIM} \quad (6)$$

EvaGaussians: Event Stream Assisted Gaussian Splatting from Blurry Images

- Integrating event streams to assist in reconstructing high-quality 3D-GS from **blurry images**;
- https://drexubery.github.io/EvaGaussians/ Novel synthetic dataset using **Color DAVIS346**;
- **Event-based double integral (EDI)** model achieves model-based image deblurring by explicitly modeling the relationship between events triggered during the exposure time and the captured blurry frames;



EvaGaussians: Event Stream Assisted Gaussian Splatting from Blurry Images

香 港 大 學
THE UNIVERSITY OF HONG KONG

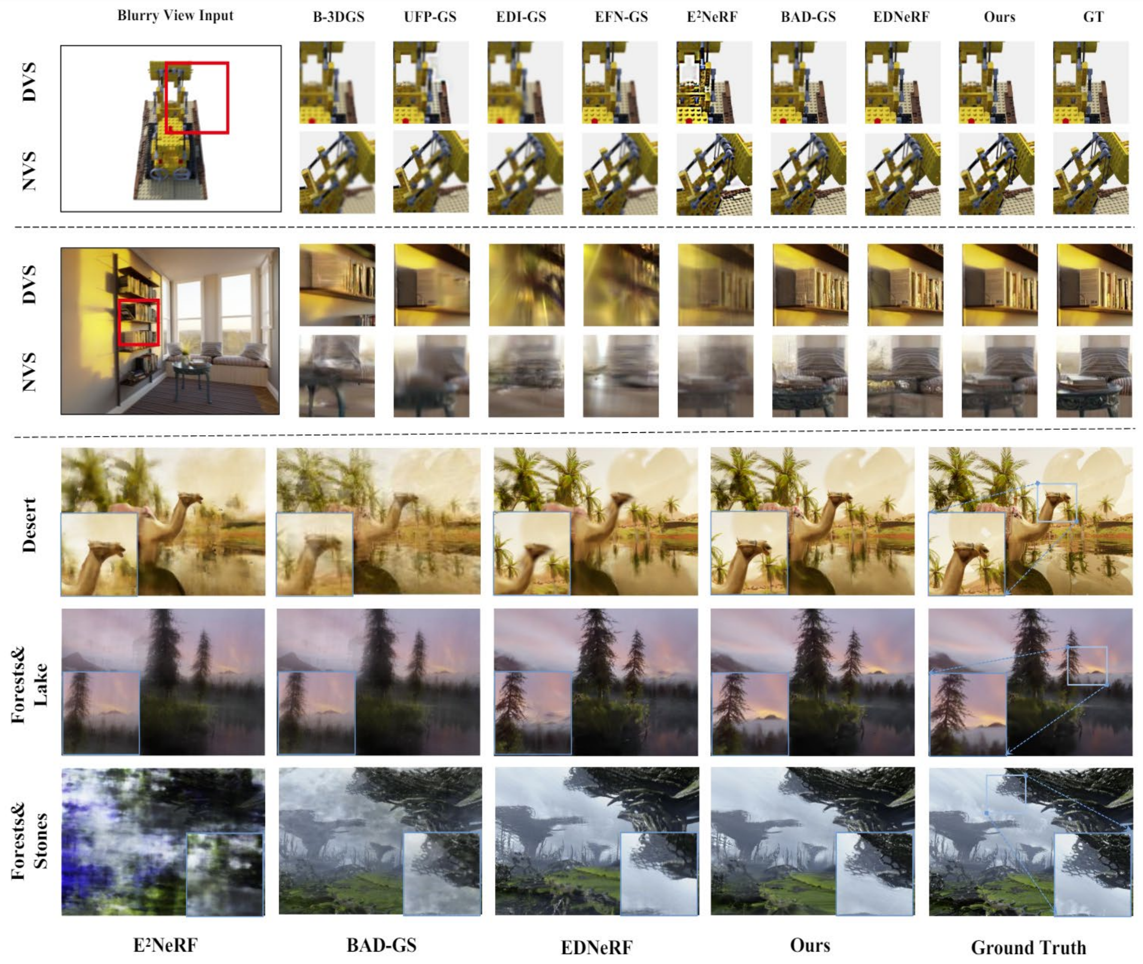


Figure 2: Qualitative comparision on the synthetic dataset. We show both novel view synthesis (NVS) results and input view deblurring (DVS) results on the top two rows. It shows that our method achieves better performance in recovering the training blurry views as well as rendering novel views. More results are presented in Appendix. B.

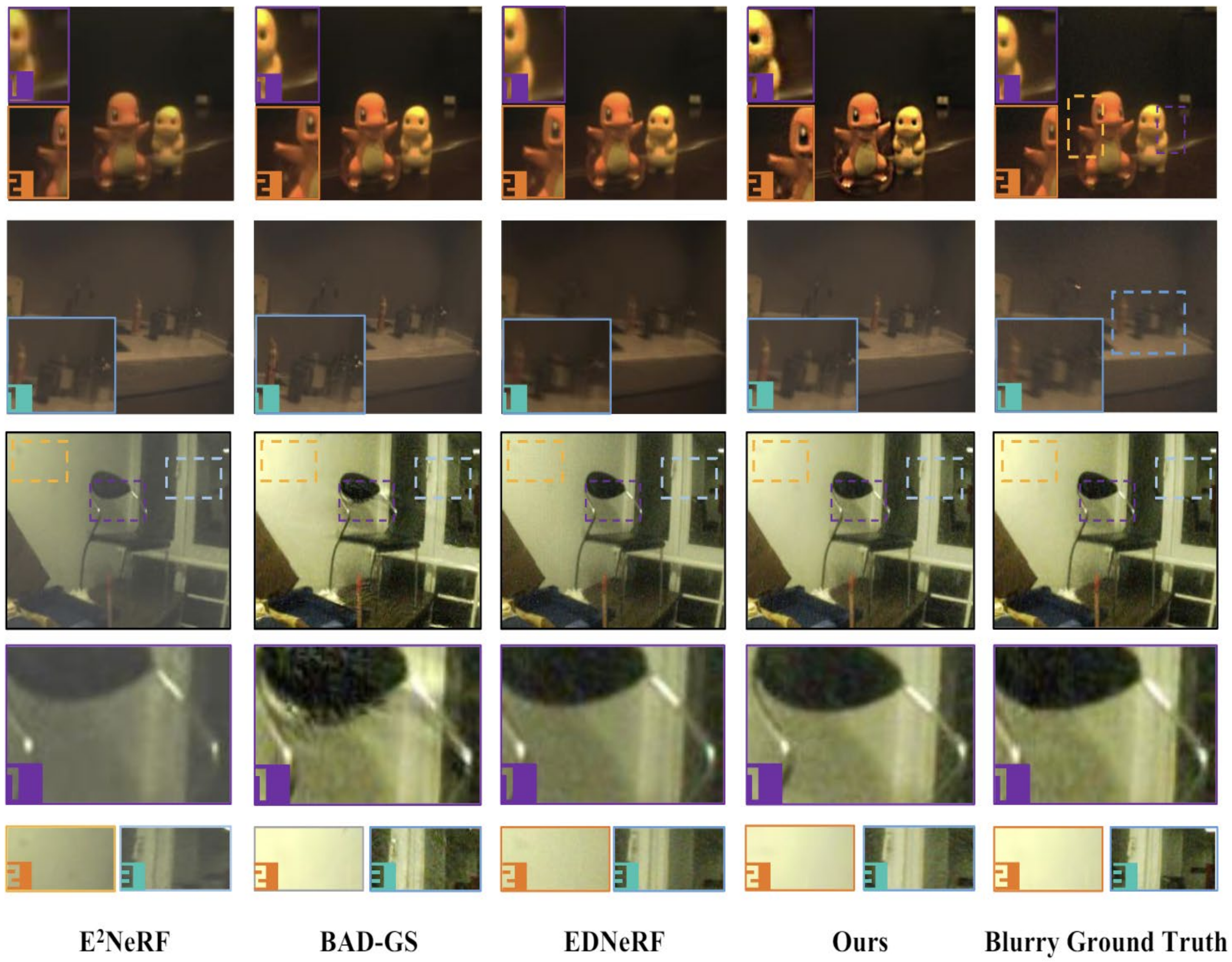
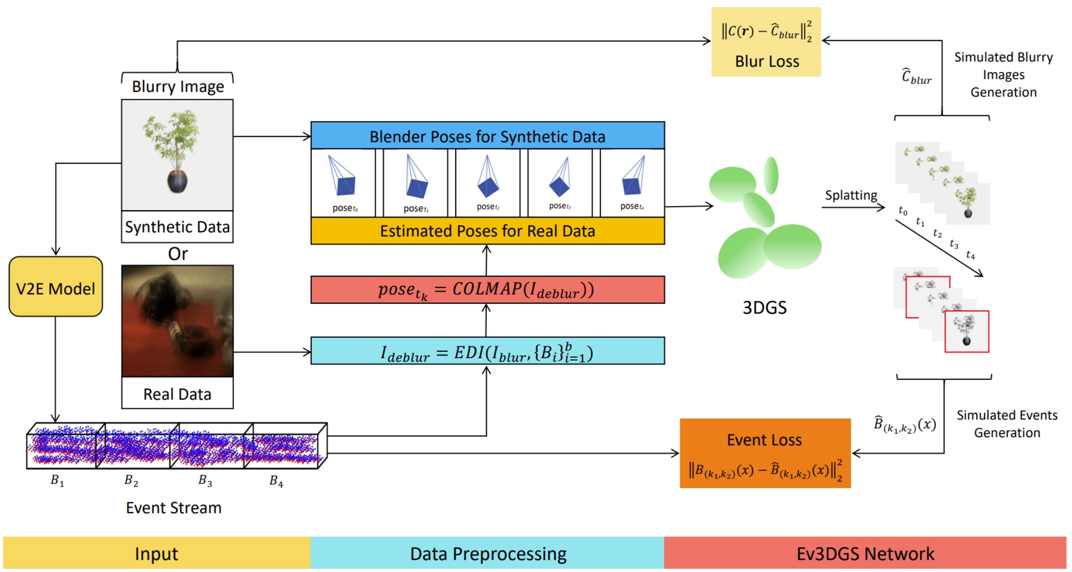


Figure 3: Qualitative results on the real-world dataset. It can be found that our method outperforms the baselines in synthesizing sharper novel views. More results are presented in Appendix. B.

Ev3DGS: Event Enhanced 3D Gaussian Splatting from Blurry Images

香 港 大 學
THE UNIVERSITY OF HONG KONG

- Utilizing the combined data from event cameras and standard RGB cameras to achieve image deblurring and realize high-quality of novel view synthesis;
- **Blur rendering loss**: superimpose the clear images obtained by rendering multiple predicted poses at equal time intervals under one viewpoint as the predicted blurred images, and compare them with the input blurred images as the blur rendering loss (**learn texture details**);
- **Event rendering loss**: the generation process of predicted event data is simulated based on the brightness change caused by the change of camera position and compared with the real event data to get the event rendering loss (**learn the motion information**);
- Developed based on E2NeRF (ICCV2023) with simulated image blur and event data, also has the real-world dataset captured by DAVIS 346;



$$\mathcal{L} = \mathcal{L}_{blur} + \omega \mathcal{L}_{event},$$

$$\mathcal{L}_{event} = \sum_{\boldsymbol{x} \in \chi} \left\| \hat{B}_{(k_1,k_2)}(\boldsymbol{x}) - B_{(k_1,k_2)}(\boldsymbol{x}) \right\|_2^2,$$

$$B_{(k_1,k_2)}(\boldsymbol{x}) = \sum_{k=k_1+1}^{k_2} B_k(\boldsymbol{x}),$$

$$\hat{B}_{(k_1,k_2)}(\boldsymbol{x}) = \begin{cases} \left[\frac{\log(L_{k_2}) - \log(L_{k_1})}{\theta_{neg}} \right], L_{k_2} < L_{k_1} \\ \left[\frac{\log(L_{k_2}) - \log(L_{k_1})}{\theta_{pos}} \right], L_{k_2} \geq L_{k_1} \end{cases}$$

$$\mathcal{L}_{blur} = \sum_{\boldsymbol{r} \in \mathcal{R}} \left[\left\| \hat{C}^c_{blur} - C(\boldsymbol{r}) \right\|_2^2 + \left\| \hat{C}^f_{blur} - C(\boldsymbol{r}) \right\|_2^2 \right]$$

Performance for both Ev3DGS and E2NeRF

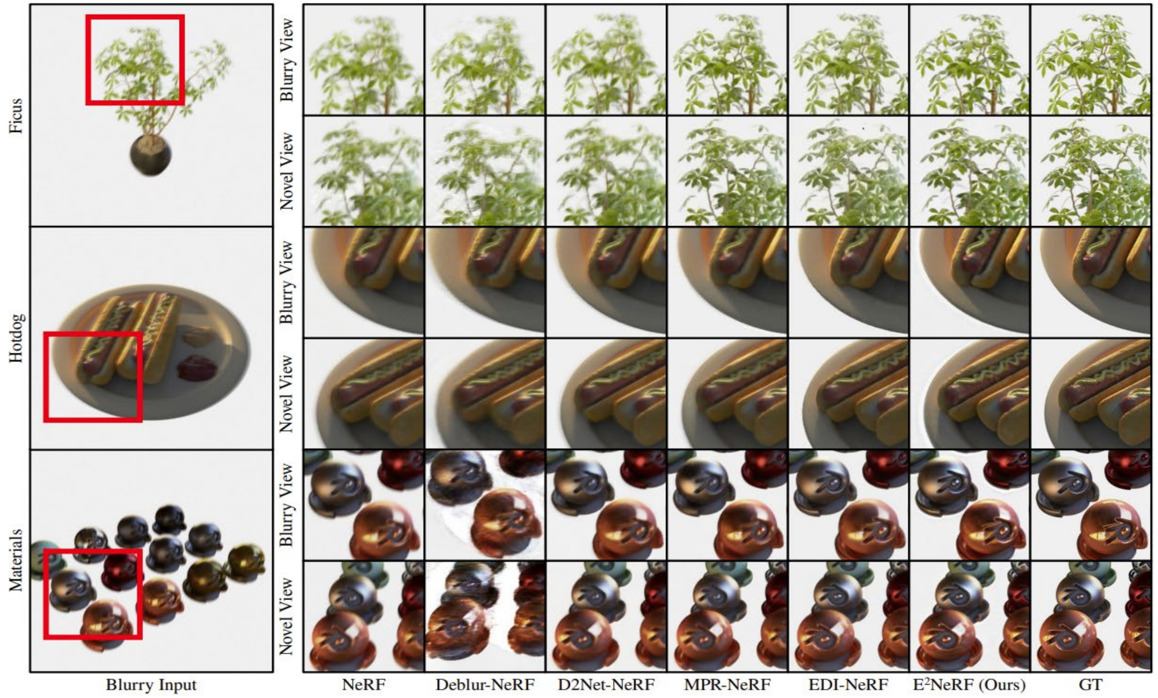


Figure 8: Qualitative comparison on synthetic data.

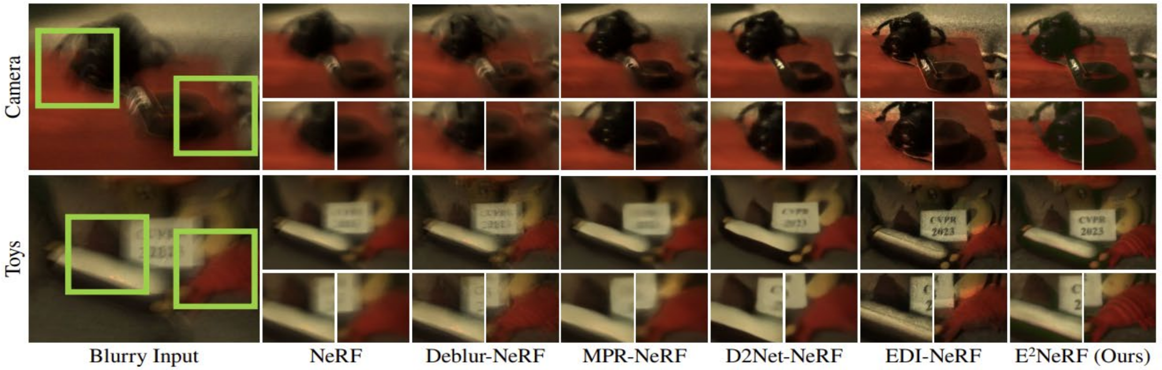


Figure 9: Qualitative comparison on real-world data.

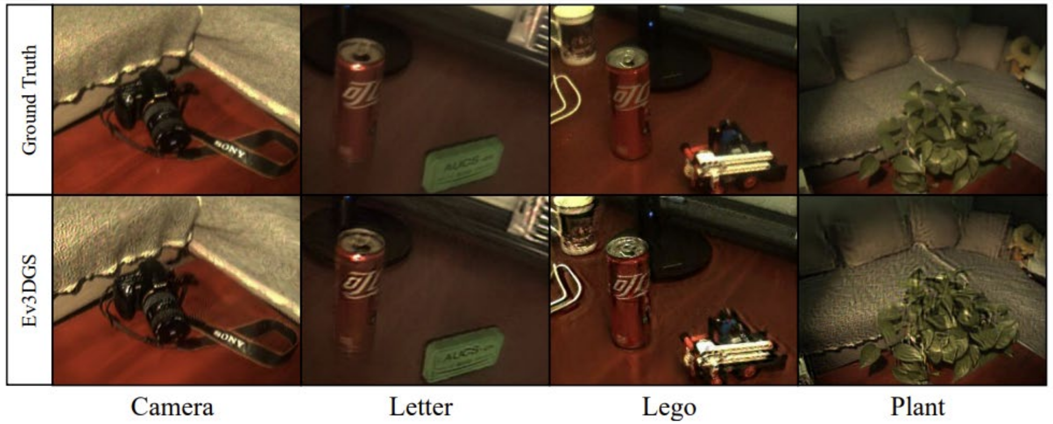


Fig. 3. Real-world data rendering results of Ev3DGS method.

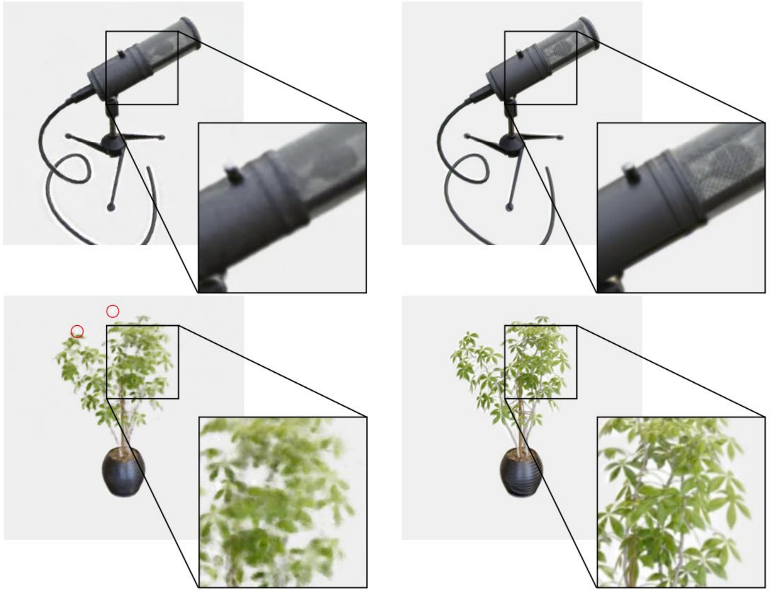


Fig. 2. Comparison of E2NeRF (left) and Our Ev3DGS (right) rendering results on synthetic data of mic (top) and ficus (bottom) scene.

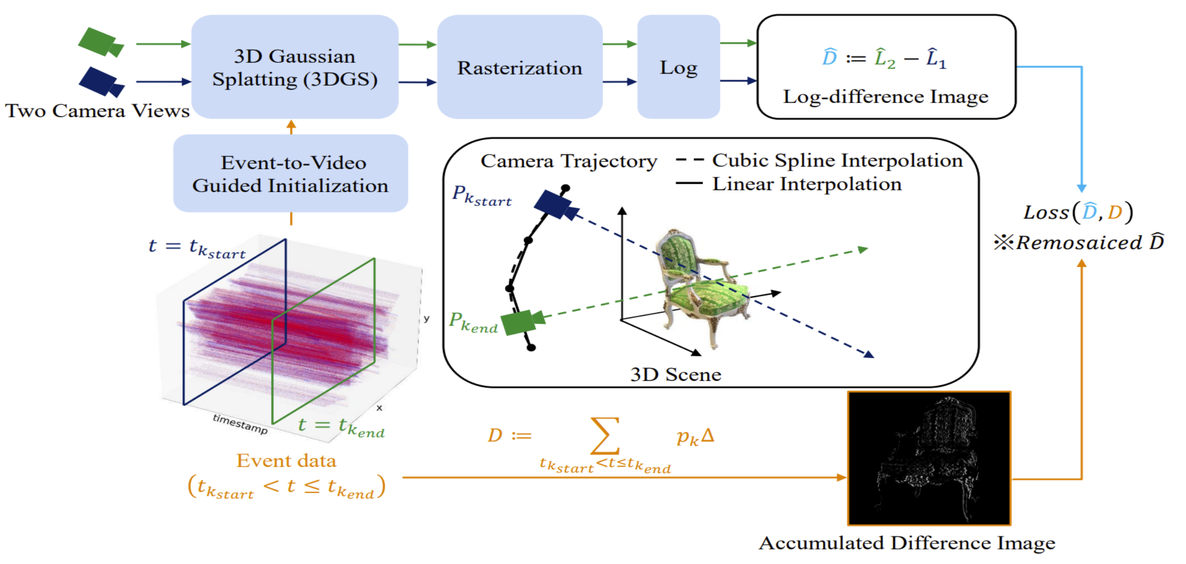
EventSplat: 3D Gaussian Splatting from Moving Event Cameras for Real-time Rendering

香 港 大 學
THE UNIVERSITY OF HONG KONG

- Address the novel view synthesis challenge in the presence of **fast motion**, using **event-only** rather than RGB images;
- An **event-to-video guided SfM approach** for initializing the 3DGS optimization process;
- Use of cubic spline trajectory interpolation for assigning camera poses to events at high rates;
- The key idea is to model changes in logarithmic image intensity (aka. Accumulated event);

$$E(a,b) := \int_a^b \log\left(I'(t)\right) \mathrm{d}t. \longrightarrow D_{x,y} := \sum_{\substack{k \in \{k_{\text{start}}, \dots, k_{\text{end}}\} \\ x_k = x,\, y_k = y}} p_k \Delta$$

- Computing the corresponding log-intensity changes by rasterizing two views from the Gaussian scene representation;



Training 3DGS using event accumulated images between two viewpoints, which represent relative intensity images. Consequently, the model cannot directly estimate absolute intensity images, **necessitating a linear transformation using evaluation data as a reference.**

Since event cameras capture variations in log-radiance rather than absolute log-radiance values, the predicted intensity $I(t)$ from the 3D Gaussian Splatting has an unknown offset. To rectify this limitation, a linear color transformation is designed to adjust our predictions in the logarithmic domain [42, 59]. This transformation is both necessary and adequately effective for aligning our predictions with the reference data. It ensures that the reconstructed intensity values are properly calibrated and aligned with the observed event data.

sian scene representation. Next, we perform a standard remosaicing operation because events are triggered per pixel asynchronously, not allowing us to use conventional demosaicing methods [5, 29, 35, 36, 41, 44]

$$\text{Remosaicing} : \mathbb{R}^{H \times W \times 3} \to \mathbb{R}^{H \times W} \quad (7)$$

which reintroduces the Bayer RGB pattern. It consists of two steps: First, a set of color-channel specific matrices of size 2×2 are Hadamard multiplied with each 2×2 pixel block in the image. The channel specific matrices $R, G, B \in \mathbb{R}^{2\times 2}$ are given by

$$R = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad G = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}. \quad (8)$$

The thus modified channels have non-zero entries at non-overlapping pixel locations. From this, the single-channel remosaiced image is obtained by addition across the channels. Finally, entry-wise logarithm computation yields the desired images $\hat{L}_1$, $\hat{L}_2$ and thus $\hat{D}$.

Evaluation in the EDS and TUM-VIE dataset

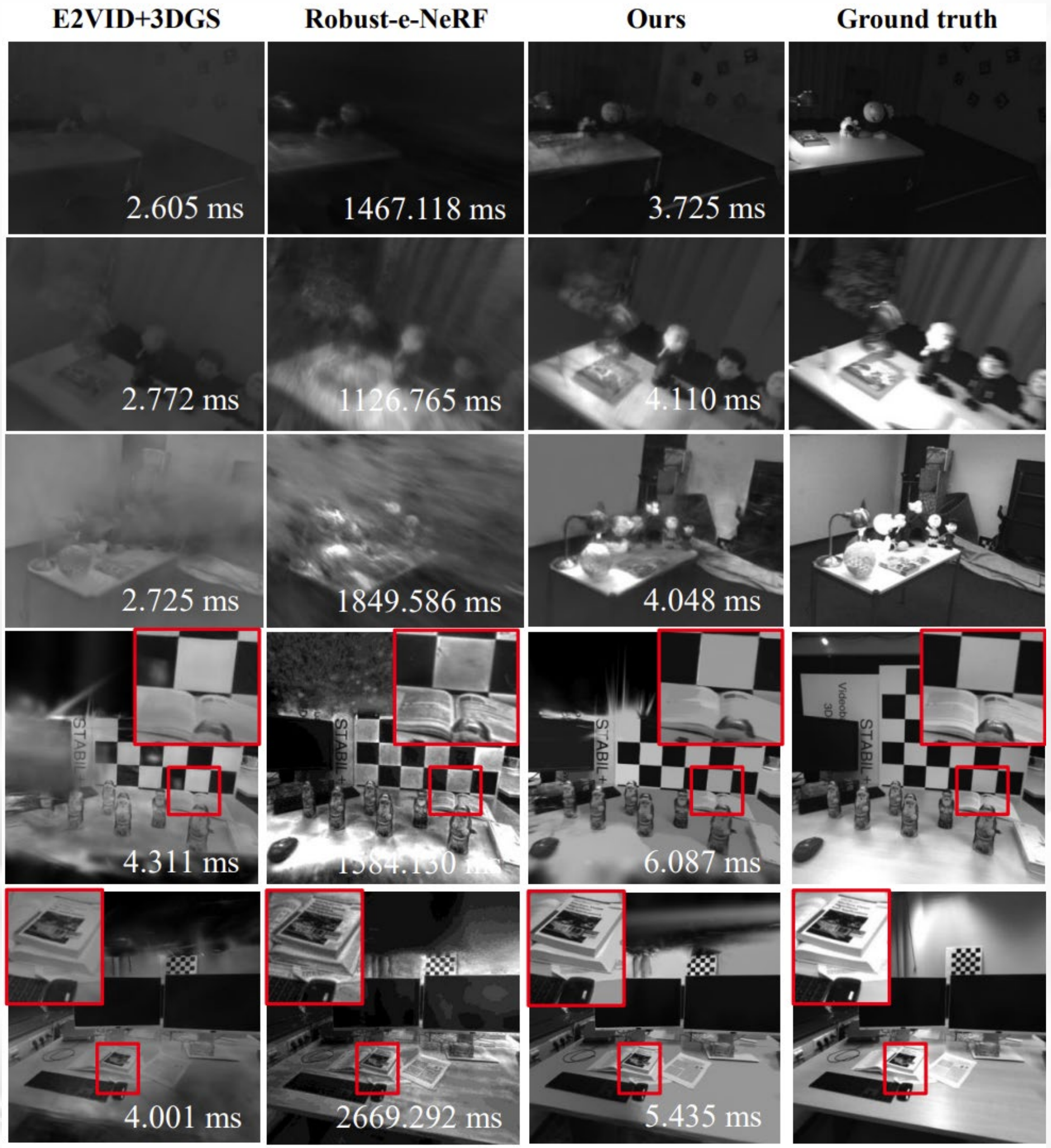


Table 2. Real scenes in comparison between our work, previous event-based NeRF and E2VID+3DGS quantitatively.

Metric	Method	Real Scene 03	07	08	11	13	Mean
PSNR ↑	E2VID + 3DGS	15.67	15.05	14.03	13.83	18.96	15.51
	Robust *e*-NeRF	19.19	14.78	14.75	14.43	18.10	16.25
	Ours	**20.78**	**19.14**	**17.53**	**17.79**	**19.05**	**18.86**
SSIM ↑	E2VID + 3DGS	0.716	0.689	0.642	0.691	0.723	0.692
	Robust *e*-NeRF	**0.846**	0.815	0.735	0.569	0.729	0.739
	Ours	0.835	**0.816**	**0.745**	**0.789**	**0.774**	**0.792**
LPIPS ↓	E2VID + 3DGS	0.266	0.378	**0.402**	0.415	0.415	0.375
	Robust *e*-NeRF	0.324	0.476	0.567	0.700	0.650	0.543
	Ours	**0.239**	**0.351**	0.424	**0.391**	**0.407**	**0.363**

Thank you