

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

Huajian Huang¹ Longwei Li² Hui Cheng² Sai-Kit Yeung¹

¹The Hong Kong University of Science and Technology ²Sun Yat-sen University

hhuangbg@connect.ust.hk, lilw23@mail2.sysu.edu.cn, chengh9@mail.sysu.edu.cn, saikit@ust.hk

Abstract

The integration of neural rendering and the SLAM system recently showed promising results in joint localization and photorealistic view reconstruction. However, existing methods, fully relying on implicit representations, are so resource-hungry that they cannot run on portable devices, which deviates from the original intention of SLAM. In this paper, we present Photo-SLAM, a novel SLAM framework with a hyper primitives map. Specifically, we simultaneously exploit explicit geometric features for localization and learn implicit photometric features to represent the texture information of the observed environment. In addition to actively densifying hyper primitives based on geometric features, we further introduce a Gaussian-Pyramid-based training method to progressively learn multi-level features, enhancing photorealistic mapping performance. The extensive experiments with monocular, stereo, and RGB-D datasets prove that our proposed system Photo-SLAM significantly outperforms current state-of-the-art SLAM systems for online photorealistic mapping, e.g., PSNR is 30% higher and rendering speed is hundreds of times faster in the Replica dataset. Moreover, the Photo-SLAM can run at real-time speed using an embedded platform such as Jetson AGX Orin, showing the potential of robotics applications. Project Page and code: <https://huajianup.github.io/research/Photo-SLAM/>.

1. Introduction

Simultaneous Localization and Mapping (SLAM) using cameras is a fundamental problem in both computer vision and robotics, seeking to enable autonomous systems to navigate and comprehend their surroundings. Traditional SLAM systems [7–9, 24] primarily focus on geometric mapping, providing accurate but visually simplistic representations of the environment. However, recent developments in neural rendering [35, 40] have demonstrated the potential of integrating photorealistic view reconstruction



Figure 1. Rendering and trajectory results. Photo-SLAM can reconstruct high-fidelity views of scenes using monocular, stereo, and RGB-D cameras while render speed is up to 1000 FPS.

into the SLAM pipeline, enhancing the perception capabilities of robotic systems.

Despite the promising results achieved through the integration of neural rendering and SLAM, existing methods simply and heavily rely on implicit representations, making them computationally intensive and unsuitable for deployment on resource-constrained devices. For example, Nice-SLAM [46] leverages a hierarchical grid [42] to store learnable features representing the environment while ESLAM [16] utilizes multi-scale compact tensor components [3]. They then jointly estimate the camera poses and optimize features by minimizing the reconstruction loss of a batch of ray sampling [21]. Such an optimization process is time-consuming. Consequently, it is indispensable for them to incorporate corresponding depth information obtained from various sources such as RGB-D cameras, dense optical flow estimators [33], or monocular depth es-

timators [12] to ensure efficient convergence. Additionally, since the implicit features are decoded by the multi-layer perceptrons (MLPs), it is typically necessary to carefully define a bounding area to normalize ray sampling for optimal performance, as discussed in [14]. It essentially limits the scalability of the system. These limitations imply that they cannot provide real-time exploration and mapping capabilities in the unknown environment using portable platforms, which is one of the main objectives of SLAM.

In this paper, we propose Photo-SLAM, an innovative framework that addresses the scalability and computational resource constraints of existing methods, while achieving precise localization and online photorealistic mapping. We maintain a hyper primitives map which is composed of point clouds storing ORB features [26], rotation, scaling, density, and spherical harmonic (SH) coefficients [10, 38]. 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 [18] rather than ray sampling. Although the introduction of a 3D Gaussian splatting renderer can reduce view reconstruction costs, it does not enable the generation of high-fidelity rendering for online incremental mapping, in particular in monocular scenarios. To achieve high-quality mapping without reliance on dense depth information, we further propose a geometry-based densification strategy and a Gaussian-Pyramid-based (GP) learning method. Importantly, GP learning facilitates the progressive acquisition of multi-level features which effectively enhances the mapping performance of our system.

To evaluate the efficacy of our proposed approach, we conduct extensive experiments employing diverse datasets captured by monocular, stereo, and RGB-D cameras. These experiment results unequivocally demonstrate that Photo-SLAM attains state-of-the-art performance in terms of localization efficiency, photorealistic mapping quality, and rendering speed. Furthermore, the real-time execution of the Photo-SLAM system on the embedded devices showcases its potential for practical robotics applications. The schematic overview of Photo-SLAM is demonstrated in Fig. 1 and Fig. 2b.

In summary, the main contributions of this work include:

- We developed the first simultaneous localization and photorealistic mapping system based on hyper primitives map. The novel framework supports monocular, stereo, and RGB-D cameras in indoor and outdoor environments.
- We proposed Gaussian-Pyramid-based learning allowing the model to efficiently and effectively learn multi-level features realizing high-fidelity mapping.
- The system, fully implemented in C++ and CUDA, achieves start-of-the-art performance and can run at real-time speed even on embedded platforms.

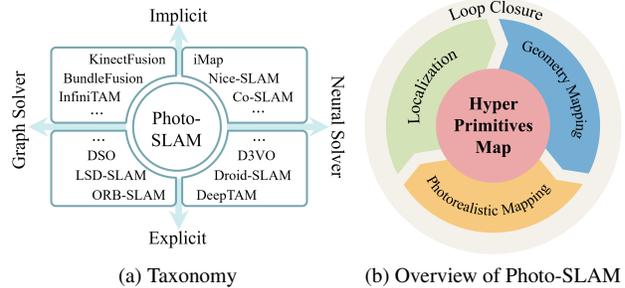


Figure 2. The Photo-SLAM contains four main components, including localization, explicit geometry mapping, implicit photorealistic mapping, and loop closure components, while maintaining a map with hyper primitives.

2. Related Work

Visual localization and mapping is a problem that aims to build a proper representation of an unknown environment via cameras while estimating their poses within that environment. In contrast to SfM techniques, visual SLAM techniques typically pursue a better trade-off between accuracy and real-time performance. In this section, we focus on visual SLAM and conduct a brief review.

Graph Solver vs Neural Solver. Classical SLAM methods widely adopt factor graphs to model complex optimization problems between variables (i.e., poses and landmarks) and measurements (i.e., observations and constraints). To achieve real-time performance, SLAM methods incrementally propagate their pose estimations while avoiding expensive operations. For example, ORB-SLAM series methods [2, 23, 24] rely on extracting and tracking lightweight geometric features across consecutive frames, which perform bundle adjustment locally instead of globally. Moreover, direct SLAMs like LSD-SLAM [7] and DSO [8] operate on raw image intensities, without the cost of geometric feature extractions. They maintain a sparse or semi-dense map represented by point clouds online, even on the resource-constraint system. Benefiting from the success of deep-learning models, learnable parameters and models are introduced into SLAM making the pipeline differentiable. Some methods such as DeepTAM [45] predict camera poses by the neural network [17] end-to-end, while the accuracy is limited. To enhance performance, some methods, e.g., D3VO [41] and Droid-SLAM [34], introduce monocular depth estimation [12] or dense optical flow estimation [33] models into the SLAM pipeline as supervision signals. Therefore, they can generate depth maps that explicitly represent the scene geometry. With the large-scale synthetic SLAM dataset, TartanAir [37], available for training, Droid-SLAM building upon RAFT [33] achieves state-of-the-art performance. However, the pure neural-based solver is computationally expensive and their performance would significantly degrade on the unseen scenes.

Explicit Representation vs Implicit Representation. In order to obtain dense reconstruction, some methods including KinectFusion [15], BundleFusion [6], and InfiniTAM [25] utilize the implicit representation, Truncated Signed Distance Function (TSDF) [5], to integrate the incoming RGB-D images and reconstruct a continuous surface, which can run in real time on GPU. Although they can obtain dense reconstruction, view rendering quality is limited. Recently, neural rendering techniques represented by neural radiance field (NeRF) [21] have achieved breathtaking novel view synthesis. Given camera poses, NeRF implicitly models the scene geometry and color by multi-layer perceptrons (MLP). The MLP is optimized by minimizing the loss of rendering images and training views. iMAP [30] then adapts NeRF for incremental mapping, optimizing not only MLP but also camera poses. The following work Nice-SLAM [46] introduces multi-resolution grids [42] to store features reducing the cost of deep MLP query. Co-SLAM [36] and ESLAM [16] explore Instant-NGP [22] and TensorRF [3] respectively to further accelerate the mapping speed. However, implicitly joint optimization of camera poses and geometry representation is still ill-conditioned. Inevitably, they rely on explicit depth information from RGB-D cameras or additional model predictions for fast convergence of the radiance field.

Our proposed Photo-SLAM seeks to recover a concise representation of the observed environment for immersive exploration rather than reconstructing a dense mesh. It maintains a map with hyper primitives online which capitalizes on explicit geometric feature points for accurate and efficient localization while leveraging implicit representations to capture and model the texture information. Please refer to Fig. 2a for the taxonomy of existing systems. Since Photo-SLAM achieves high-quality mapping without reliance on dense depth information, it can support RGB-D cameras as well as monocular and stereo cameras.

3. Photo-SLAM

Photo-SLAM contains four main components, including localization, geometry mapping, photorealistic mapping, and loop closure, shown in Fig. 2b. Each component runs in a parallel thread and jointly maintains a hyper primitives map.

3.1. Hyper Primitives Map

In our system, hyper primitives are defined as a set of point clouds $\mathbf{P} \in \mathbb{R}^3$ associated with ORB features [26] $\mathbf{O} \in \mathbb{R}^{256}$, rotation $\mathbf{r} \in SO(3)$, scaling $\mathbf{s} \in \mathbb{R}^3$, density $\sigma \in \mathbb{R}^1$, and spherical harmonic coefficients $\mathbf{SH} \in \mathbb{R}^{16}$. ORB features extracted from image frames take responsibility for establishing 2D-to-2D and 2D-to-3D correspondences. Once the system successfully estimates the transformation matrix based on sufficient 2D-to-2D correspondences between adjacent frames, the hyper primitives map is initialized via tri-

angulation, and pose tracking gets started. During tracking, the localization component processes the incoming images and makes use of 2D-to-3D correspondence to calculate current camera poses. In addition, the geometry mapping component will incrementally create and initialize sparse hyper primitives. Finally, the photorealistic component progressively optimizes and densifies hyper primitives.

3.2. Localization and Geometry Mapping

The localization and geometry mapping components provide not only efficient 6-DoF camera pose estimations of the input images, but also sparse 3D points. The optimization problem is formulated as a factor graph solved by the Levenberg–Marquardt (LM) algorithm.

In the localization thread, we use a motion-only bundle adjustment to optimize the camera orientation $\mathbf{R} \in SO(3)$ and position $\mathbf{t} \in \mathbb{R}^3$ in order to minimize the reprojection error between matched 2D geometric keypoint \mathbf{p}_i of the frame and 3D point \mathbf{P}_i . Let $i \in \mathcal{X}$ be the index of set of matches \mathcal{X} , what we are trying to optimize with LM is

$$\{\mathbf{R}, \mathbf{t}\} = \arg \min_{\mathbf{R}, \mathbf{t}} \sum_{i \in \mathcal{X}} \rho \left(\|\mathbf{p}_i - \pi(\mathbf{R}\mathbf{P}_i + \mathbf{t})\|_{\Sigma_g}^2 \right), \quad (1)$$

where Σ_g is the scale-associated covariance matrix of the keypoint, $\pi(\cdot)$ is the 3D-to-2D projection function, and ρ denotes the robust Huber cost function.

In the geometry mapping thread, we perform a local bundle adjustment on a set of covisible points \mathcal{P}_L and keyframes \mathcal{K}_L . The keyframes are selected frames from the input camera sequence and provide good visual information. We construct a factor graph where each keyframe is a node, and the edges represent constraints between keyframes and matched 3D points. We iteratively minimize the reprojection residual by refining the keyframe poses and 3D points using the first-order derivatives of the error function. We fix the poses of keyframes \mathcal{K}_F which are also observing \mathcal{P}_L but not in \mathcal{K}_L . Let $\mathcal{K} = \mathcal{K}_L \cup \mathcal{K}_F$, and \mathcal{X}_k be the set of matches between 2D keypoints in a keyframe k and 3D points in \mathcal{P}_L . The optimization process aims to reduce the geometric inconsistency between \mathcal{K} and \mathcal{P}_L , and is defined as

$$\{\mathbf{P}_i, \mathbf{R}_l, \mathbf{t}_l | i \in \mathcal{P}_L, l \in \mathcal{K}_L\} = \arg \min_{\mathbf{P}_i, \mathbf{R}_l, \mathbf{t}_l} \sum_{k \in \mathcal{K}} \sum_{j \in \mathcal{X}_k} \rho(E(k, j)), \quad (2)$$

with reprojection residual

$$E(k, j) = \|\mathbf{p}_j - \pi(\mathbf{R}_k \mathbf{P}_j + \mathbf{t}_k)\|_{\Sigma_g}^2.$$

3.3. Photorealistic Mapping

The photorealistic mapping thread is responsible for optimizing hyper primitives that are incrementally created by the geometry mapping thread. The hyper primitives can be

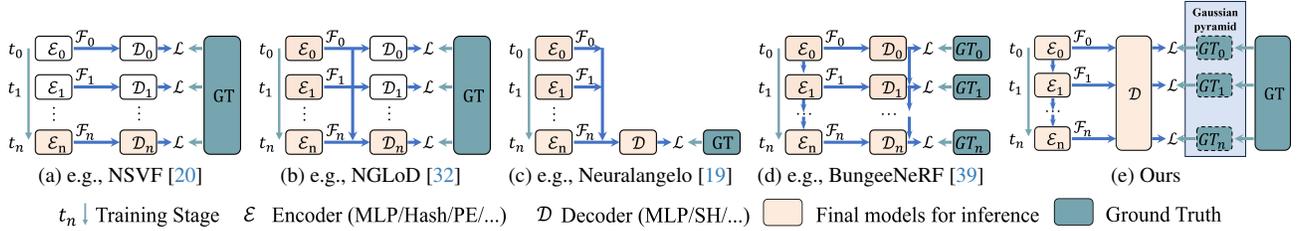


Figure 3. Comparison of different progressive training methods. The encoder \mathcal{E}_n here represents a structure to regress features \mathcal{F}_n which can be an MLP, voxel grid, hash table, positional encoding, etc. The decoder \mathcal{D}_n here represents a structure converting \mathcal{F}_n into density, color, or other information. We proposed a new method based on the Gaussian pyramid to efficiently learn multi-level features.

rasterized by a tile-based renderer to synthesize corresponding images with keyframe poses. The rendering process is formulated as

$$C(\mathbf{R}, \mathbf{t}) = \sum_{i \in N} \mathbf{c}_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j), \quad (3)$$

where N is the number of hyper primitives, \mathbf{c}_i denotes the color converted from $\mathbf{SH} \in \mathbb{R}^{16}$, and α_i is equal to $\sigma_i \cdot \mathcal{G}(\mathbf{R}, \mathbf{t}, \mathbf{P}_i, \mathbf{r}_i, \mathbf{s}_i)$, \mathcal{G} denotes 3D Gaussian splatting algorithm [18]. The optimization in terms of position \mathbf{P} , rotation \mathbf{r} , scaling \mathbf{s} , density σ , and spherical harmonic coefficients \mathbf{SH} is performed by minimizing the photometric loss \mathcal{L} between rendering image I_r and ground truth image I_{gt} , denoted as

$$\mathcal{L} = (1 - \lambda) |I_r - I_{gt}|_1 + \lambda(1 - \text{SSIM}(I_r, I_{gt})), \quad (4)$$

where $\text{SSIM}(I_r, I_{gt})$ denotes structural similarity between two images and λ is a weight factor for balance.

3.3.1 Geometry-based Densification

If we consider photorealistic mapping as a regression model of the scene, denser hyper primitives, i.e., more parameters, generally can better model the complexity of the scene for higher rendering quality. To meet the demand for real-time mapping, the geometry mapping component only establishes sparse hyper primitives. Therefore, the coarse hyper primitives created by the geometry mapping need to be densified during the optimization of photorealistic mapping. Apart from splitting or cloning hyper primitives with large loss gradients similar to [18], we introduce an additional geometry-based densification strategy.

Experimentally, less than 30% of 2D geometric feature points of frames are active and have corresponding 3D points, especially for non-RGB-D scenarios, as shown in Fig. 4. We argue that 2D geometric feature points spatially distributed in the frames essentially represent the region with a complex texture that requires more hyper primitives. Therefore, we actively create additional temporary hyper primitives based on the inactive 2D feature points once the keyframe is created for photorealistic mapping. When we

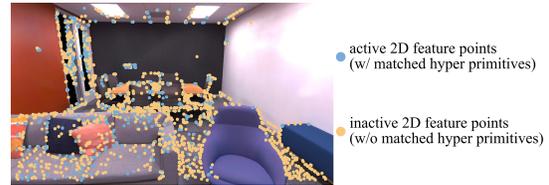


Figure 4. We make use of initial geometric information to densify hyper primitives.

use RGB-D cameras, we can directly project the inactive 2D feature points with depth to create temporary hyper primitives. As for monocular scenarios, we estimate the depth of inactive 2D feature points by interpreting the depth of their nearest neighborhood’s active 2D feature points. In stereo scenarios, we rely on a stereo-matching algorithm to estimate the depth of inactive 2D feature points.

3.4. Gaussian-Pyramid-Based Learning

Progressive training is a widely used technology in neural rendering to accelerate the optimization process. Some methods have been proposed to reduce training time while achieving better rendering quality. A basic method is to progressively increase the structure resolution and the number of model parameters. For example, NSVF [20] and DVGO [31] progressively increase the feature grid resolution during training which significantly improves training efficiency compared to previous work. The lower-resolution model is used to initialize the higher-resolution model but is not retained for final inference, as shown in Fig. 3a. To enhance performance with multi-resolution features, NGLoD [32] progressively trains multiple MLPs as encoders and decoders, while only retaining the final decoder to decode integrated multi-resolution features, as shown in Fig. 3b. Furthermore, Neuralangelo [19] only maintains a single MLP during training, as shown in Fig. 3c. It progressively activates different levels of hash tables [22] achieving better performance in large-scale scene reconstruction. Similarly, 3D Gaussian Splatting [18] progressively densifies 3D Gaussian achieving top performance on radiance field rendering. Training different level mod-

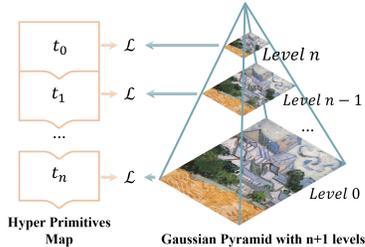


Figure 5. Training process based on the Gaussian pyramid.

els in these methods is supervised by the same training images. Conversely, the fourth method (Fig. 3d) used in BungeeNeRF [39] is to apply different models to tackle different-resolution images. BungeeNeRF demonstrates the efficiency of explicitly grouping multi-resolution training images for models to learn multi-level features. However, such a method is not universal since multi-resolution images are not available for most scenarios.

To make full use of various merits, we propose Gaussian-Pyramid-based (GP) learning (Fig. 3e), a new progressive training method. As illustrated in Fig. 5, a Gaussian pyramid is a multi-scale representation of an image containing different levels of detail. It is constructed by repeatedly applying Gaussian smoothing and downsampling operations to the original image. At the beginning training step, the hyper primitives are supervised by the highest level of the pyramid, *i.e.* level n . As training iteration increases, we not only densify hyper primitives as described in Sec. 3.3.1 but also reduce the pyramid level and obtain a new ground truth until reaching the bottom of the Gaussian pyramid. The optimization process using a Gaussian pyramid with $n+1$ levels can be denoted as

$$\begin{aligned}
 t_0 &: \arg \min \mathcal{L}(I_r^n, \text{GP}^n(I_{\text{gt}})), \\
 t_1 &: \arg \min \mathcal{L}(I_r^{n-1}, \text{GP}^{n-1}(I_{\text{gt}})), \\
 &\dots \\
 t_n &: \arg \min \mathcal{L}(I_r^0, \text{GP}^0(I_{\text{gt}})),
 \end{aligned} \tag{5}$$

where $\mathcal{L}(I_r, \text{GP}(I_{\text{gt}}))$ is Eq. 4, while $\text{GP}^n(I_{\text{gt}})$ denotes the ground image in the level n of the Gaussian pyramid. In the experiment, we prove that GP learning significantly improves the performance of photorealistic mapping particularly for monocular cameras.

3.5. Loop Closure

Loop Closure [11] is crucial in SLAM because it helps address the problem of accumulated errors and drift that can occur during the localization and geometry mapping process. After detecting a closing loop, we can correct local keyframes and hyper primitives by similarity transformation. With corrected camera poses, the photorealistic mapping component can further get rid of the ghosting caused by odometry drifts and improve the mapping quality.

4. Experiment

In this section, we compare Photo-SLAM to other state-of-the-art (SOTA) SLAM and real-time 3D reconstruction systems in various scenarios encapsulating monocular, stereo, RGB-D cameras, and indoor and outdoor environments. In addition, we evaluate Photo-SLAM performance on various hardware configurations to demonstrate its efficiency. Finally, we conduct an ablation study to verify the effectiveness of the proposed algorithms.

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 of photorealistic mapping is performed with the Stochastic Gradient Descent algorithm while we use a fixed learning rate and $\lambda = 0.2$. Considering image resolution of the testing dataset, the level of the Gaussian pyramid is set to three, *i.e.* $n = 2$ by default. The compared baseline includes a SOTA classical SLAM system ORB-SLAM3 [2], a real-time RGB-D dense reconstruction system BundleFusion [6], a deep-learning based system DROID-SLAM [34], and recent SLAM systems supporting view synthesis, *i.e.* Nice-SLAM [46], OrbeeZ-SLAM [4], ESLAM [16], CoSLAM [36], and Point-SLAM [27] and Go-SLAM [44].

Hardware. We ran Photo-SLAM and all compared methods using their official code in a desktop with an NVIDIA RTX 4090 24 GB GPU, an Intel Core i9-13900K CPU, and 64 GB RAM. We further tested Photo-SLAM on a laptop and a Jetson AGX Orin Developer Kit. The laptop is equipped with an NVIDIA RTX 3080ti 16 GB Laptop GPU, an Intel Core i9-12900HX, and 32 GB RAM.

Datasets and Metrics. We performed tests for monocular and RGB-D sensor types on the well-known RGB-D datasets: the Replica dataset [28, 30] and the TUM RGB-D dataset [29]. As for stereo tests, we used the EuRoC MAV dataset [1]. Besides indoor scenes, we utilize a ZED 2 stereo camera to collect outdoor scenes for extra evaluation.

Following the convention, we used the Absolute Trajectory Error (ATE) metric [13] to estimate the accuracy of localization, while the RMSE and STD of ATE are reported. Quantitative measurements in terms of PSNR, SSIM, and LPIPS [43] are adopted to analyze the performance of photorealistic mapping. We also report the requirement of computing resources by showing the tracking FPS, rendering FPS, and GPU memory usage. The evaluation regarding mesh reconstruction is out of the range of this work. Moreover, to lower the effect of the nondeterministic nature of multi-threading and machine-learning systems, we ran each sequence five times and reported the average results for each metric. Please refer to the supplementary for details.

On Replica Dataset		Localization (cm)		Mapping			Resources			
Cam	Method	RMSE ↓	STD ↓	PSNR ↑	SSIM ↑	LPIPS ↓	Operation Time ↓	Tracking FPS ↑	Rendering FPS ↑	GPU Memory Usage ↓
Mono	ORB-SLAM3 [2]	3.942	3.115	-	-	-	<1 mins	58.749	-	0
	DROID-SLAM [34]	0.725	0.308	-	-	-	<2 mins	35.473	-	11 GB
	Nice-SLAM* [46]	99.9415	35.336	16.311	0.720	0.439	>10 mins	2.384	0.944	12 GB
	Orbee-SLAM [4]	-	-	23.246	0.790	0.336	<5 mins	49.200	1.030	6 GB
	Go-SLAM [44]	71.054	24.593	21.172	0.703	0.421	<5 mins	25.366	0.821	22 GB
	Ours (Jetson)	1.235	0.756	29.284	0.883	0.139	<5 mins	18.315	95.057	4 GB
	Ours (Laptop)	0.713	0.524	33.049	0.926	0.086	<5 mins	19.974	353.504	4 GB
	Ours	1.091	0.892	33.302	0.926	0.078	<2 mins	41.646	911.262	6 GB
RGB-D	ORB-SLAM3 [2]	1.833	1.478	-	-	-	<1 mins	52.209	-	0
	DROID-SLAM [34]	0.634	0.248	-	-	-	<2 mins	36.452	-	11 GB
	BundleFusion [6]	1.606	0.969	23.839	0.822	0.197	<5 mins	8.630	-	5 GB
	Nice-SLAM [46]	2.350	1.590	26.158	0.832	0.232	>10 mins	2.331	0.611	12 GB
	Orbee-SLAM [4]	0.888	0.562	32.516	0.916	0.112	<5 mins	41.333	1.401	6 GB
	ESLAM [16]	0.568	0.274	30.594	0.866	0.162	<5 mins	6.687	2.626	21 GB
	Co-SLAM [36]	1.158	0.602	30.246	0.864	0.175	<5 mins	14.575	3.745	4 GB
	Go-SLAM [44]	0.571	0.218	24.158	0.766	0.352	<5 mins	19.437	0.444	24 GB
	Point-SLAM [27]	0.596	0.249	34.632	0.927	0.083	>2 hrs	0.345	0.510	24 GB
	Ours (Jetson)	0.581	0.289	31.978	0.916	0.101	<5 mins	17.926	116.395	4 GB
	Ours (Laptop)	0.590	0.289	34.853	0.944	0.062	<5 mins	20.597	396.082	4 GB
	Ours	0.604	0.298	34.958	0.942	0.059	<2 mins	42.485	1084.017	5 GB

Table 1. Quantitative results on the Replica dataset. We mark the best two results with **first** and **second**. Nice-SLAM* means the depth supervision is disabled. “-” denotes the system does not support view rendering or fails to track camera poses. The results of Photo-SLAM running on the laptop and Jetson platform are denoted as “Ours (Laptop)” and “Ours (Jetson)” respectively.

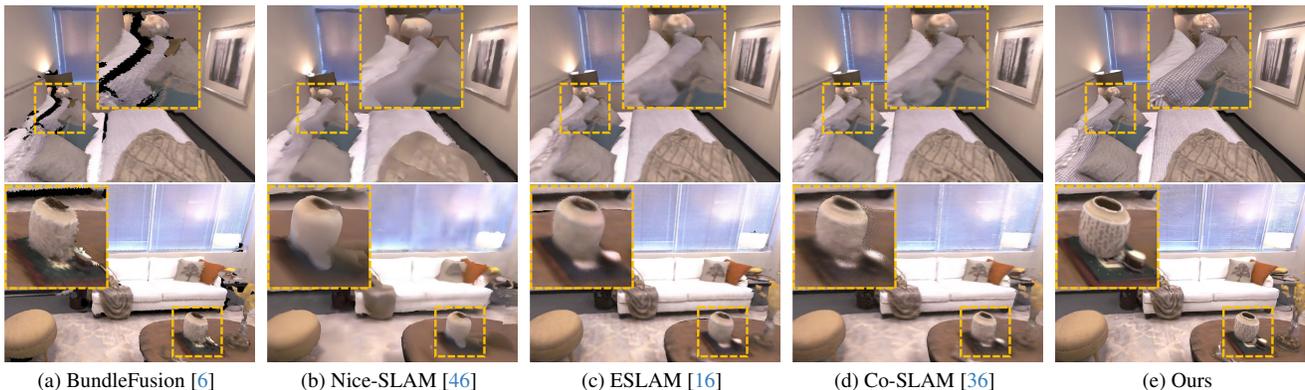


Figure 6. Qualitative comparison of diverse systems using RGB-D images from dataset Replica. Photo-SLAM can reconstruct high-fidelity scenes while others are over-smooth and have obvious artifacts. Zoom in for better views.

4.2. Results and Evaluation

On Replica. As quantitative comparison demonstrated in Table 1, Photo-SLAM achieves top performance in terms of mapping quality. With competitive localization accuracy, Photo-SLAM can track the camera poses in real time. Moreover, Photo-SLAM renders hundreds of photorealistic views in a resolution of 1200×680 per second with less GPU memory usage. Even on the embedded platform, the rendering speed of Photo-SLAM is about 100 FPS.

In monocular scenarios, Photo-SLAM significantly suppresses other methods. When we disabled the depth supervision of Nice-SLAM [46], its accuracy of localization dramatically decreased while the mapping was of 16.311 PSNR. We conduct a qualitative comparison in Fig. 7. The mapping results of Photo-SLAM are photorealistic.

In RGB-D scenarios, we ran BundleFusion [6] with

RGB-D sequences and then extracted textured mesh. And then we used a mesh renderer to render corresponding images for comparison. As shown in Fig. 6, the mesh reconstructed by the classical method is likely to be aliasing and hollow. ESLAM [16] and Go-SLAM [44] have the best localization accuracy, but the mapping lacks high-frequency details. By contrast, Photo-SLAM can render high-fidelity images and the rendering speed is about **three hundred times faster**.

On TUM. We provide quantitative analyses on the three sequences of the TUM dataset in Table 2. Compared to learning-based methods, *e.g.*, DROID-SLAM [34] and Go-SLAM [44], ORB-SLAM3 runs faster without the requirement of GPU and has higher accuracy regarding localization. It is shown that the classical method still has advantages in terms of robustness and generalization. Fig. 8 is a gallery of Photo-SLAM mapping.

On Stereo. Stereo cameras can provide more robust track-

On TUM Dataset		fr1-desk				fr2-xyz				fr3-office			
Cam	Method	RMSE (cm) ↓	PSNR ↑	SSIM ↑	LPIPS ↓	RMSE (cm) ↓	PSNR ↑	SSIM ↑	LPIPS ↓	RMSE (cm) ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Mono	ORB-SLAM3 [2]	1.534	-	-	-	0.720	-	-	-	1.400	-	-	-
	DROID-SLAM [34]	78.245	-	-	-	36.050	-	-	-	154.383	-	-	-
	Go-SLAM [44]	33.122	11.705	0.406	0.614	28.584	14.807	0.443	0.572	105.755	13.572	0.480	0.643
	Ours (Jetson)	1.757	18.811	0.681	0.329	0.558	21.347	0.727	0.187	1.687	18.884	0.672	0.289
	Ours (Laptop)	1.549	20.515	0.733	0.241	0.852	21.575	0.739	0.157	1.542	19.138	0.680	0.259
	Ours	1.539	20.972	0.743	0.228	0.984	21.072	0.726	0.166	1.257	19.591	0.692	0.239
RGB-D	ORB-SLAM3 [2]	1.724	-	-	-	0.385	-	-	-	1.698	-	-	-
	DROID-SLAM [34]	91.985	-	-	-	41.833	-	-	-	160.141	-	-	-
	Nice-SLAM [46]	19.317	12.003	0.417	0.510	36.103	18.200	0.603	0.313	25.309	16.341	0.548	0.386
	ESLAM [16]	3.359	17.497	0.561	0.484	31.448	22.225	0.727	0.233	25.808	19.113	0.616	0.359
	Co-SLAM [36]	3.094	16.419	0.482	0.591	31.347	19.176	0.595	0.374	25.374	17.863	0.547	0.452
	Go-SLAM [44]	2.119	15.794	0.531	0.538	31.788	16.118	0.534	0.419	26.802	16.499	0.566	0.569
	Ours (Jetson)	4.571	18.273	0.663	0.338	0.360	23.127	0.780	0.149	1.874	19.781	0.701	0.235
	Ours (Laptop)	1.891	20.403	0.728	0.251	0.361	22.570	0.777	0.158	1.315	21.569	0.749	0.184
	Ours	2.603	20.870	0.743	0.239	0.346	22.094	0.765	0.169	1.001	22.744	0.780	0.154

Table 2. Quantitative results on the TUM RGB-D dataset. We mark the best two results with **first** and **second**.

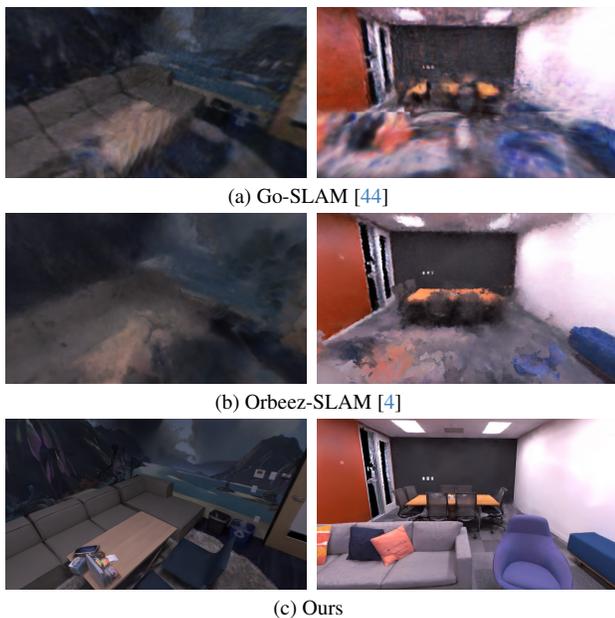


Figure 7. Mapping comparison with other monocular camera systems on *Replica office3* scene.

ing but have hardly been supported by former real-time dense SLAM systems. However, Photo-SLAM has been designed to be compatible with stereo cameras. We provide quantitative results on the EuRoC dataset in Table 3 and qualitative results in supplementary. The results show that our system could still perform decently in stereo scenes. Further, we used a hand-held stereo camera to collect some outdoor scenes, and the mapping results of Photo-SLAM are illustrated in Fig. 9.

4.3. Ablation Study

We proposed geometry-based densification (Geo) and Gaussian-Pyramid-based (GP) learning to boost the system performance of real-time photorealistic mapping. In this

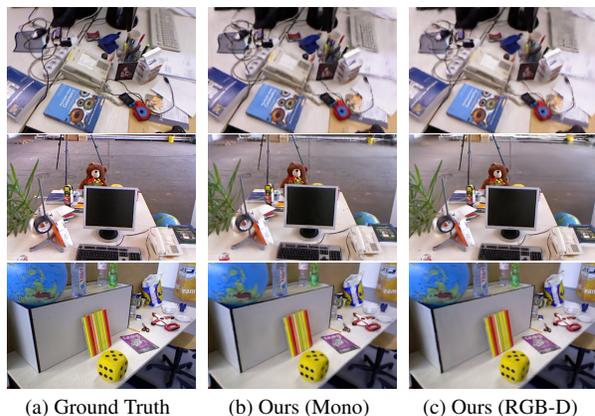


Figure 8. Qualitative results of Photo-SLAM on dataset TUM.

On Euroc Stereo	ORB-SLAM3	DROID-SLAM	Ours (Jetson)	Ours (Laptop)	Ours	
MH-01	RMSE (cm) ↓	4.379	39.514	4.207	4.049	4.109
	PSNR ↑	-	-	13.979	13.962	13.952
	SSIM ↑	-	-	0.426	0.421	0.420
	LPIPS ↓	-	-	0.428	0.378	0.366
MH-02	RMSE (cm) ↓	4.525	39.265	4.193	4.731	4.441
	PSNR ↑	-	-	14.210	14.254	14.201
	SSIM ↑	-	-	0.436	0.436	0.430
	LPIPS ↓	-	-	0.447	0.373	0.356
V1-01	RMSE (cm) ↓	8.940	21.646	8.830	8.836	8.821
	PSNR ↑	-	-	16.933	17.025	17.069
	SSIM ↑	-	-	0.626	0.622	0.618
	LPIPS ↓	-	-	0.321	0.284	0.266
V2-01	RMSE (cm) ↓	26.904	15.344	26.643	26.736	26.609
	PSNR ↑	-	-	16.038	16.052	15.677
	SSIM ↑	-	-	0.643	0.635	0.622
	LPIPS ↓	-	-	0.347	0.314	0.323

Table 3. Quantitative results on the EuRoC MAV dataset, using stereo inputs. Our Photo-SLAM is the first system to support on-line photorealistic mapping with stereo cameras.

section, we constructed an ablation study to measure the efficacy of each algorithm, which can be quantified by PSNR, rendering speed (FPS), and final model size in megabytes (MB). The quantitative results are demonstrated in Table 4.



Figure 9. Mapping results of Photo-SLAM using a hand-held stereo camera in an outdoor unbounding scene.

On Replica			Mono			RGB-D		
#	Geo	GP	PSNR \uparrow	FPS \uparrow	MB	PSNR \uparrow	FPS \uparrow	MB
(1)	w/o	n=2	31.274	994.2	10.742	33.296	923.0	18.199
(2)	w/	w/o	20.002	353.2	44.100	33.696	860.0	31.856
(3)	w/o	w/o	22.913	645.0	5.782	32.551	1010.8	13.901
(4)	w/	n=1	30.903	803.8	21.819	34.634	953.7	31.552
(5)	w/	n=3	31.563	877.6	22.510	33.305	946.2	31.039
default	w/	n=2	33.302	911.3	31.419	34.958	1084.0	35.211

Table 4. Ablation study on the effect of geometry-based densification (Geo) and Gaussian-Pyramid-based (GP) learning.

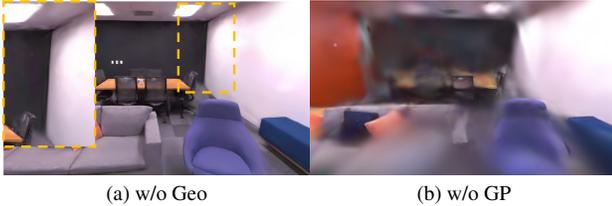


Figure 10. Mapping results of different ablated systems on the monocular Replica scene.

The relationship is tangled between rendering quality or speed, and the number of hyper primitives represented by the model size. The models are typically small without actively densifying hyper primitives based on the inactive 2D geometric features (Geo). However, without Geo, PSNR suffers degradation by 2.028 and 1.662 on monocular and RGB-D scenarios respectively, as indicated in Table 4(1). Compared to Fig. 7c, the rendering image without Geo (Fig. 10a) exhibits artifacts, such as on the ceiling. In RGB-D scenarios, more hyper primitives can get higher PSNR. However, without Gaussian-Pyramid-based learning, more hyper primitives are densified by Geo, and thus lead to a decrease in mapping quality and rendering speed especially in monocular scenarios, as visualized in Fig. 10b and reported in Table 4(2) and (3). This is because densified hyper primitives without precise depth information have inaccurate positions. Without thorough optimization, inaccurate

hyper primitives become encumbrances. It is noticeable that the systems with GP learning can generally perform better, highlighting the effectiveness of GP learning. Additionally, increasing the Gaussian pyramid levels improves mapping quality, and our Photo-SLAM with default 3-level GP learning achieves the best results. However, we found that the results of using a Gaussian pyramid with 4 levels deteriorated, as shown in Table 4(5), possibly due to overfitting low-level features during incremental mapping. Since the image is rendered by splatting hyper primitives, the rendering speed theoretically is correlated to the number of visible hyper primitives in the current view rather than the model size of the whole scene. Although a smaller model does not necessarily imply a higher average rendering speed, a precise reconstructed model should accurately capture its essential details while using concise parameters, which is the premise for high-speed rendering. Moreover, reducing the time required for rendering enables the optimization of hyper primitives with more iterations during online mapping, ultimately leading to improved accuracy and quality.

In conclusion, the ablation study verifies that the geometry-based densification strategy allows the system to obtain sufficient hyper primitives while the Gaussian-Pyramid-based learning guarantees hyper primitives optimized thoroughly, enhancing online photorealistic mapping performance. There is no doubt that the default Photo-SLAM can reconstruct a map with more appropriate hyper primitives, and achieve better rendering quality and high rendering speed than ablated systems.

5. Conclusion

In this paper, we have proposed a novel SLAM framework called Photo-SLAM for simultaneous localization and photorealistic mapping. Instead of highly relying on resource-intensive implicit representations and neural solvers, we introduced a hyper primitives map. It enables our system to leverage explicit geometric features for localization and implicitly capture the texture information of the scenes. In addition to geometry-based densification, we proposed Gaussian-Pyramid-based learning, a new progressive training method, to further enhance mapping performance. Extensive experiments have demonstrated that Photo-SLAM significantly outperforms existing SOTA SLAMs for online photorealistic mapping. Furthermore, our system verifies its practicality by achieving real-time performance on an embedded platform, highlighting its potential for advanced robotics applications in real-world scenarios.

Acknowledgements: This work is partially supported by the Innovation and Technology Support Programme of the Innovation and Technology Fund (Ref: ITS/200/20FP), the Marine Conservation Enhancement Fund (MCEF20107 & MCEF23EG01), and an internal grant from HKUST (R9429).

References

- [1] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*, 2016. **5**
- [2] Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM Montiel, and Juan D Tardós. Orb-slam3: An accurate open-source library for visual, visual-inertial, and multi-map slam. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021. **2, 5, 6, 7**
- [3] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision (ECCV)*, 2022. **1, 3**
- [4] Chi-Ming Chung, Yang-Che Tseng, Ya-Ching Hsu, Xiang-Qian Shi, Yun-Hung Hua, Jia-Fong Yeh, Wen-Chin Chen, Yi-Ting Chen, and Winston H Hsu. Orbeez-slam: A real-time monocular visual slam with orb features and nerf-realized mapping. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9400–9406. IEEE, 2023. **5, 6, 7**
- [5] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312, 1996. **3**
- [6] Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Transactions on Graphics (ToG)*, 36(4): 1, 2017. **3, 5, 6**
- [7] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *European conference on computer vision*, pages 834–849. Springer, 2014. **1, 2**
- [8] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):611–625, 2017. **2**
- [9] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 15–22. IEEE, 2014. **1**
- [10] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5501–5510, 2022. **2**
- [11] Dorian Gálvez-López and J. D. Tardós. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, 2012. **5**
- [12] Clément Godard, Oisín Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 270–279, 2017. **2**
- [13] Michael Grupp. evo: Python package for the evaluation of odometry and slam. <https://github.com/MichaelGrupp/evo>, 2017. **5**
- [14] Huajian Huang, Yingshu Chen, Tianjian Zhang, and Sai-Kit Yeung. 360roam: Real-time indoor roaming using geometry-aware 360° radiance fields. *arXiv preprint arXiv:2208.02705*, 2022. **2**
- [15] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568, 2011. **3**
- [16] Mohammad Mahdi Johari, Camilla Carta, and François Fleuret. Eslam: Efficient dense slam system based on hybrid representation of signed distance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17408–17419, 2023. **1, 3, 5, 6, 7**
- [17] Alex Kendall, Matthew Grimes, and Roberto Cipolla. PoseNet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision*, pages 2938–2946, 2015. **2**
- [18] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (ToG)*, 42(4):1–14, 2023. **2, 4, 5, 1**
- [19] Zhaoshuo Li, Thomas Müller, Alex Evans, Russell H Taylor, Mathias Unberath, Ming-Yu Liu, and Chen-Hsuan Lin. Neuralangelo: High-fidelity neural surface reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8456–8465, 2023. **4**
- [20] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *Advances in Neural Information Processing Systems*, 33:15651–15663, 2020. **4**
- [21] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer, 2020. **1, 3**
- [22] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multi-resolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022. **3, 4**
- [23] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE transactions on robotics*, 33(5):1255–1262, 2017. **2**
- [24] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015. **1, 2**
- [25] V. A. Prisacariu, O. Kahler, M. M. Cheng, C. Y. Ren, J. Valentin, P. H. S. Torr, I. D. Reid, and D. W. Murray. A Framework for the Volumetric Integration of Depth Images. *ArXiv e-prints*, 2014. **3**

- [26] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. *IEEE International Conference on Computer Vision*, 58(11):2564–2571, 2011. 2, 3
- [27] Erik Sandström, Yue Li, Luc Van Gool, and Martin R. Oswald. Point-slam: Dense neural point cloud-based slam. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. 5, 6
- [28] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard Newcombe. The Replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019. 5
- [29] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, 2012. 5
- [30] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew J Davison. imap: Implicit mapping and positioning in real-time. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6229–6238, 2021. 3, 5
- [31] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5459–5469, 2022. 4
- [32] Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Neural geometric level of detail: Real-time rendering with implicit 3d shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11358–11367, 2021. 4
- [33] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 402–419. Springer, 2020. 1, 2
- [34] Zachary Teed and Jia Deng. Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras. *Advances in neural information processing systems*, 34:16558–16569, 2021. 2, 5, 6, 7
- [35] Ayush Tewari, Justus Thies, Ben Mildenhall, Pratul Srinivasan, Edgar Tretschk, Wang Yifan, Christoph Lassner, Vincent Sitzmann, Ricardo Martin-Brualla, Stephen Lombardi, et al. Advances in neural rendering. In *Computer Graphics Forum*, pages 703–735. Wiley Online Library, 2022. 1
- [36] Hengyi Wang, Jingwen Wang, and Lourdes Agapito. Co-slam: Joint coordinate and sparse parametric encodings for neural real-time slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13293–13302, 2023. 3, 5, 6, 7, 1
- [37] Wenshan Wang, DeLong Zhu, Xiangwei Wang, Yaoyu Hu, Yuheng Qiu, Chen Wang, Yafei Hu, Ashish Kapoor, and Sebastian Scherer. Tartanair: A dataset to push the limits of visual slam. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4909–4916. IEEE, 2020. 2
- [38] Suttisak Wizadwongsa, Pakkapon Phongthawee, Jiraphon Yenphraphai, and Supasorn Suwajanakorn. Nex: Real-time view synthesis with neural basis expansion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8534–8543, 2021. 2
- [39] Yuanbo Xiangli, Linning Xu, Xingang Pan, Nanxuan Zhao, Anyi Rao, Christian Theobalt, Bo Dai, and Dahua Lin. Bungeenerf: Progressive neural radiance field for extreme multi-scale scene rendering. In *European conference on computer vision*, pages 106–122. Springer, 2022. 4, 5
- [40] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. In *Computer Graphics Forum*, pages 641–676. Wiley Online Library, 2022. 1
- [41] Nan Yang, Lukas von Stumberg, Rui Wang, and Daniel Cremers. D3vo: Deep depth, deep pose and deep uncertainty for monocular visual odometry. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1281–1292, 2020. 2
- [42] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenotrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5752–5761, 2021. 1, 3
- [43] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 5
- [44] Youmin Zhang, Fabio Tosi, Stefano Mattoccia, and Matteo Poggi. Go-slam: Global optimization for consistent 3d instant reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3727–3737, 2023. 5, 6, 7, 1
- [45] Huizhong Zhou, Benjamin Ummenhofer, and Thomas Brox. DeepTam: Deep tracking and mapping. In *Proceedings of the European conference on computer vision (ECCV)*, pages 822–838, 2018. 2
- [46] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R Oswald, and Marc Pollefeys. Nice-slam: Neural implicit scalable encoding for slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12786–12796, 2022. 1, 3, 5, 6, 7

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

Supplementary Material



Figure 11. The proposed system has real-time performance on embedded platforms, such as Jetson AGX Orin Developer Kit.

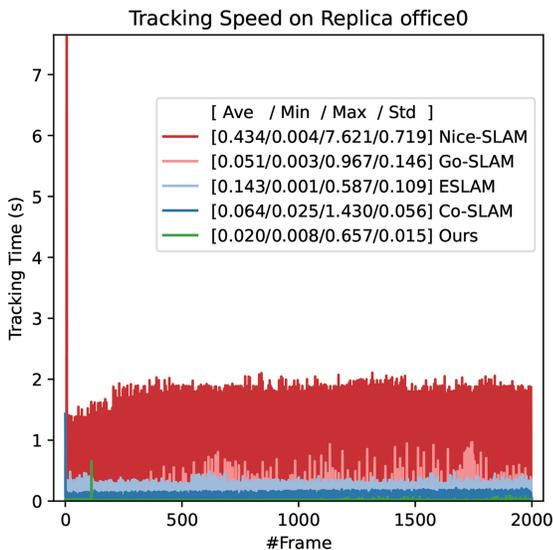


Figure 12. Tracking speed comparisons on scene office0 using an RGB-D camera. The vertical axis denotes the processing time of each frame while the horizontal axis denotes the frame number. [Ave/Min/Max/Std] represent the average, minimum, and maximum tracking time and its standard deviation respectively.

Photo-SLAM is a novel system for simultaneous localization and photorealistic mapping, which can even run on embedded platforms at real-time speed, as demonstrated in Fig. 11. In this supplementary, we provide additional results regarding localization and mapping performance.

6. Localisation

Stability. As online systems, SLAMs are required to process the incoming frames and estimate current camera poses in time. Therefore, tracking stability regarding latency and the average processing time is an important factor in evaluating system performance in addition to pose estimation accuracy. As reported in Table 1 of the main paper, Photo-SLAM is capable of processing more than 40 frames per second with accurate pose estimation. The average tracking speed is about six times faster than ESLAM [16] and three times faster than Co-SLAM [36]. Here, we provide additional analysis on tracking stability while an example plotted in Fig. 12.

Although the average tracking time of Go-SLAM [44] is less than Co-SLAM and ESLAM, the processing latency is high due to frequently conducting expensive global optimization. As shown in Fig. 12, Go-SLAM often takes about 1 second to process the frame and estimate the pose. Moreover, both Nice-SLAM [46] and Co-SLAM need a longer time to accurately initialize the tracking. Obviously, our system can rapidly and stably process the incoming frames, having minimum average tracking time and standard deviation. The peak processing time of our system occurs when loop closure is detected for correcting pose estimation drift. **Accuracy.** Some qualitative tracking results of Photo-SLAM are demonstrated in Fig. 13.

7. Discussion

Online Mapping vs Offline Mapping. For online mapping, the mapping process occurs simultaneously with the localization process. Therefore it requires continuous and prompt updates with each new observation as the robot or camera moves and observes its surroundings. In general, online photorealistic mapping is more challenging than offline photorealistic mapping, since it is crucial to balance the trade-off between computational efficiency and rendering quality. As mentioned in the main paper, we proposed a geometry-based densification strategy and a Gaussian-Pyramid-based (GP) learning method to achieve high-quality online mapping. To further support this statement, we compared the photorealistic mapping performance between our Photo-SLAM and 3D Gaussian splatting (3DG) [18]. 3DG is the SOTA offline method which takes a set of images with known poses and a sparse point cloud as input to learn a radiance field for view synthesis. During the experiments, 3DG used the keyframe poses esti-



(a) Replica office0



(b) Replica room0



(c) TUM fr3-office

Figure 13. Trajectory in the reconstructed map. Green points denote ground truth trajectory while red denotes the estimated trajectory of Photo-SLAM.

mated by Photo-SLAM and performed training for the same duration as Photo-SLAM. The required point cloud input is initialized in three different ways: 1) randomly initializing 100 points; 2) randomly initializing 10,000 points; and 3) initializing from the hyper primitives map of Photo-SLAM. The results are reported in Table 5. Without inputting fine-grained point clouds, 3DG needs more time for optimization such that the rendering quality decreases. In addition, to enhance rendering quality, 3DG tends to densify point clouds leading to larger model size and slower rendering speed. Whether using monocular cameras or RGB-D cameras, Photo-SLAM consistently delivers compelling rendering quality and faster rendering speeds, owing to the effectiveness of the proposed algorithms.

method	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	Rendering FPS \uparrow	Model Size (MB)
1) 3DG	27.844	0.861	0.213	745.480	36.141
2) 3DG	34.555	0.942	0.065	483.904	144.196
3) 3DG	37.055	0.962	0.032	448.109	219.470
Ours using Mono	33.302	0.926	0.078	911.262	31.419
Ours using RGB-D	34.958	0.942	0.059	1084.017	35.211

Table 5. Comparison of mapping performance between 3D Gaussian splatting (3DG) and our system Photo-SLAM with different settings on the Replica dataset.

On TUM Dataset			Resources		
Scene	Cam	Method	Tracking FPS \uparrow	Rendering FPS \uparrow	Model Size (MB)
fr1-desk	Mono	Ours (Jetson)	28.267	340.507	4.610
		Ours (Laptop)	28.330	1105.062	7.421
		Ours	57.781	2016.690	10.027
	RGB-D	Ours (Jetson)	27.970	380.622	5.743
		Ours (Laptop)	28.930	1061.040	8.432
		Ours	58.378	2083.896	9.963
fr2-xyz	Mono	Ours (Jetson)	24.005	169.321	14.286
		Ours (Laptop)	24.922	619.554	16.102
		Ours	58.241	1405.797	20.380
	RGB-D	Ours (Jetson)	21.032	274.718	6.319
		Ours (Laptop)	22.665	701.590	13.850
		Ours	52.904	1790.120	21.399
fr3-office	Mono	Ours (Jetson)	36.700	291.398	10.669
		Ours (Laptop)	38.929	824.658	16.249
		Ours	81.575	1522.120	19.211
	RGB-D	Ours (Jetson)	18.039	291.907	12.726
		Ours (Laptop)	19.636	764.342	15.349
		Ours	43.650	1540.757	17.009

Table 6. Additional results of Photo-SLAM with different platforms on the TUM dataset.

On EuRoC Dataset		Resources		
Scene	Method	Tracking FPS \uparrow	Rendering FPS \uparrow	Model Size (MB)
MH-01	Ours (Jetson)	21.359	93.762	43.385
	Ours (Laptop)	25.019	316.403	89.700
	Ours	44.977	613.958	123.528
MH-02	Ours (Jetson)	22.355	101.021	36.263
	Ours (Laptop)	26.189	332.174	81.569
	Ours	46.556	675.508	113.116
V1-01	Ours (Jetson)	21.332	106.008	28.444
	Ours (Laptop)	25.403	367.903	55.263
	Ours	44.763	835.119	74.457
V2-01	Ours (Jetson)	23.872	99.988	27.840
	Ours (Laptop)	27.556	307.025	62.588
	Ours	48.911	595.234	82.600

Table 7. Additional results of Photo-SLAM with different platforms on the EuRoC MAV stereo dataset.

8. More Results

The results of each scene of the replica dataset are detailed in Table 8. Additional qualitative results on the TUM dataset are demonstrated on Fig. 14 and Fig. 15, while Fig. 16 illustrates qualitative results of Photo-SLAM on EuRoC Stereo

On Replica Dataset			Localization		Mapping			Resources		
Scene	Cam	Method	Trajectory (RMSE _{cm}) ↓	Rotation (RMSE) ↓	PSNR ↑	SSIM ↑	LPIPS ↓	Tracking FPS ↑	Rendering FPS ↑	Model Size (MB)
office0	Mono	Ours (Jetson)	0.467	0.00334	34.415	0.940	0.949	18.328	123.551	17.703
		Ours (Laptop)	0.587	0.00343	36.227	0.954	0.071	20.422	413.645	21.703
		Ours	0.575	0.00369	36.989	0.955	0.061	42.487	930.598	24.975
	RGB-D	Ours (Jetson)	0.499	0.00356	35.447	0.949	0.086	19.076	154.087	18.281
		Ours (Laptop)	0.519	0.00317	38.219	0.965	0.053	22.446	497.917	18.826
		Ours	0.522	0.00307	38.477	0.964	0.050	48.588	1447.887	19.740
office1	Mono	Ours (Jetson)	5.586	0.31140	32.382	0.904	0.113	18.312	80.048	22.904
		Ours (Laptop)	0.379	0.00463	37.970	0.954	0.060	19.968	322.160	21.224
		Ours	0.315	0.00383	37.592	0.950	0.062	42.296	857.324	26.982
	RGB-D	Ours (Jetson)	0.402	0.00517	37.510	0.953	0.065	19.194	118.584	20.656
		Ours (Laptop)	0.440	0.00543	39.109	0.962	0.049	22.349	496.644	19.548
		Ours	0.436	0.00477	39.089	0.961	0.047	47.333	1263.343	21.193
office2	Mono	Ours (Jetson)	1.402	0.01452	28.083	0.900	0.131	17.502	90.181	19.704
		Ours (Laptop)	2.087	0.02154	31.202	0.927	0.098	18.975	343.662	27.332
		Ours	5.031	0.04696	31.794	0.929	0.091	39.604	930.777	31.558
	RGB-D	Ours (Jetson)	1.209	0.00964	29.755	0.919	0.110	17.860	124.420	27.560
		Ours (Laptop)	1.188	0.00972	32.720	0.940	0.080	21.507	425.452	31.711
		Ours	1.276	0.01094	33.034	0.938	0.077	44.062	904.249	34.065
office3	Mono	Ours (Jetson)	0.429	0.00232	28.058	0.886	0.132	17.881	96.872	15.505
		Ours (Laptop)	0.409	0.00239	32.012	0.924	0.090	19.518	368.530	20.475
		Ours	0.472	0.00227	31.622	0.920	0.086	40.870	1131.957	26.653
	RGB-D	Ours (Jetson)	0.718	0.00222	30.954	0.917	0.103	17.889	120.118	20.270
		Ours (Laptop)	0.747	0.00233	33.594	0.939	0.072	20.051	388.624	23.617
		Ours	0.782	0.00233	33.789	0.938	0.066	40.603	1125.175	25.226
office4	Mono	Ours (Jetson)	0.579	0.00305	30.399	0.921	0.109	18.755	102.949	15.201
		Ours (Laptop)	0.616	0.00279	33.656	0.940	0.078	20.311	375.033	21.444
		Ours	0.583	0.00272	34.168	0.941	0.072	42.262	849.305	26.154
	RGB-D	Ours (Jetson)	0.661	0.00367	32.219	0.931	0.091	17.107	92.237	32.405
		Ours (Laptop)	0.629	0.00446	35.534	0.951	0.059	19.361	333.874	32.270
		Ours	0.582	0.00423	36.020	0.952	0.054	39.870	1061.749	35.421
room0	Mono	Ours (Jetson)	0.369	0.00321	26.423	0.787	0.221	17.987	87.246	17.121
		Ours (Laptop)	0.349	0.00294	29.899	0.868	0.125	19.521	332.127	33.151
		Ours	0.345	0.00299	29.772	0.871	0.106	41.020	754.729	44.333
	RGB-D	Ours (Jetson)	0.514	0.00265	27.867	0.833	0.165	17.424	104.248	31.196
		Ours (Laptop)	0.521	0.00257	31.288	0.914	0.075	19.119	322.585	52.266
		Ours	0.541	0.00270	30.716	0.899	0.075	39.825	897.870	55.397
room1	Mono	Ours (Jetson)	0.803	0.00670	27.076	0.841	0.177	19.834	99.038	19.699
		Ours (Laptop)	1.046	0.00868	30.459	0.902	0.092	21.580	333.430	32.959
		Ours	1.183	0.00772	31.302	0.910	0.083	44.316	782.326	43.865
	RGB-D	Ours (Jetson)	0.381	0.00299	30.191	0.895	0.108	18.881	121.986	29.503
		Ours (Laptop)	0.399	0.00277	33.071	0.931	0.062	21.782	367.455	43.568
		Ours	0.394	0.00320	33.511	0.934	0.057	43.352	1018.111	49.617
room2	Mono	Ours (Jetson)	0.241	0.00280	27.432	0.889	0.138	17.918	80.568	16.303
		Ours (Laptop)	0.235	0.00263	32.970	0.935	0.075	19.499	339.442	22.790
		Ours	0.225	0.00258	33.181	0.934	0.067	40.313	1053.078	26.834
	RGB-D	Ours (Jetson)	0.260	0.00257	31.883	0.928	0.078	15.982	95.480	33.592
		Ours (Laptop)	0.275	0.00250	35.295	0.954	0.045	18.158	336.103	38.307
		Ours	0.305	0.00257	35.028	0.951	0.043	36.244	953.755	41.032

Table 8. Detailed results of Photo-SLAM with different platforms on the Replica dataset.

Dataset.



Figure 14. Qualitative results of Photo-SLAM on TUM using monocular cameras.

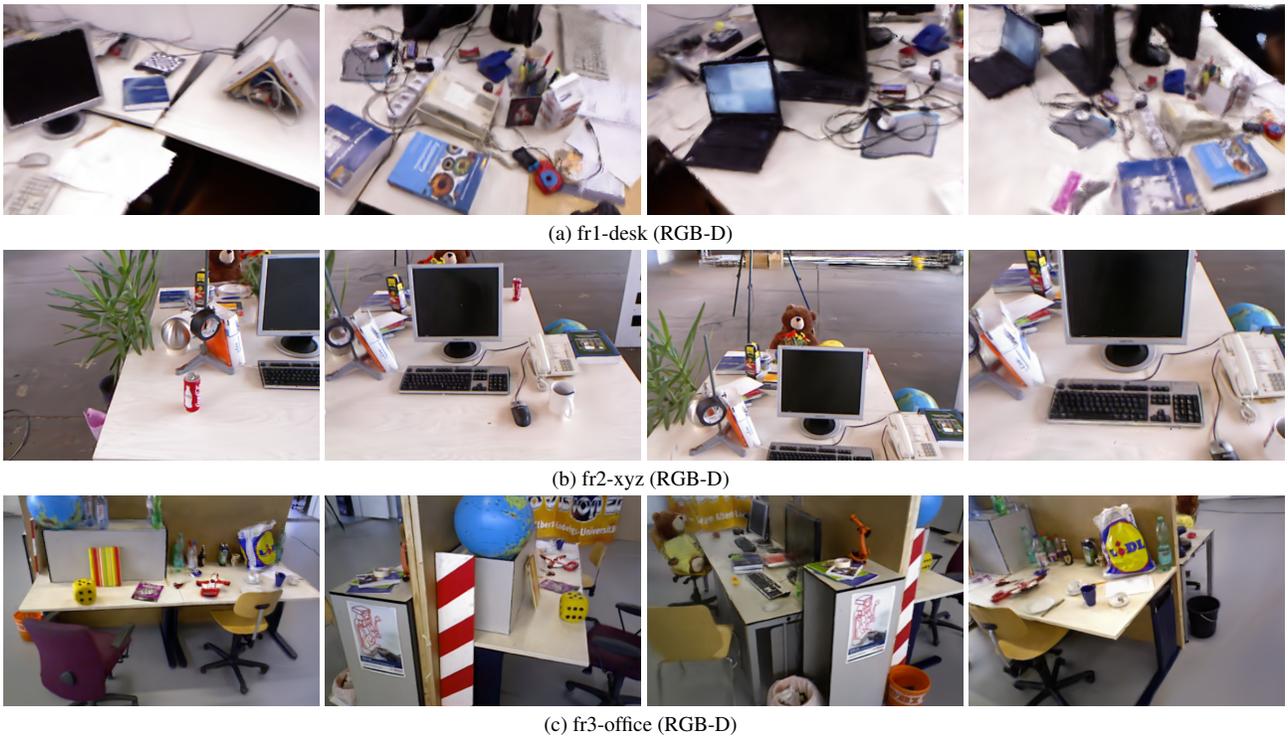
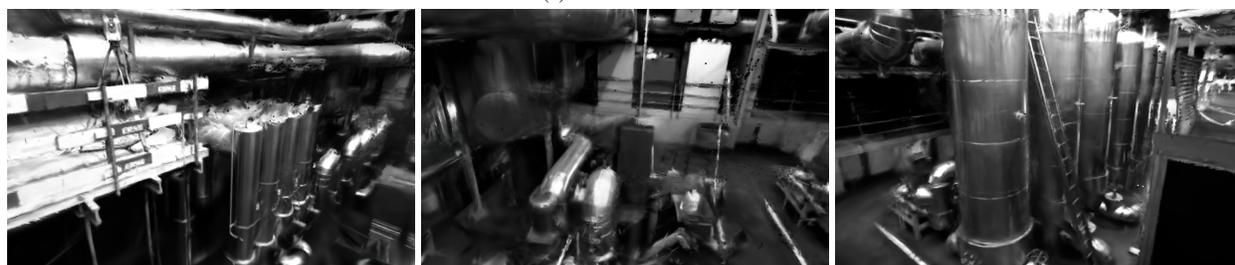


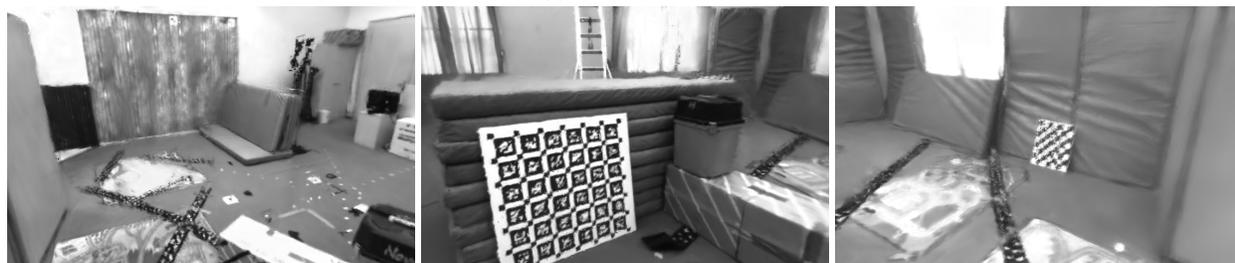
Figure 15. Qualitative results of Photo-SLAM on TUM using RGB-D cameras.



(a) EuRoC MH-01



(b) EuRoC MH-02



(c) EuRoC V1-01



(d) EuRoC V2-01

Figure 16. Qualitative results of Photo-SLAM on stereo EuRoC.