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

Huajian Huang, Longwei Li, Hui Cheng and Sai-Kit Yeung

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/`**

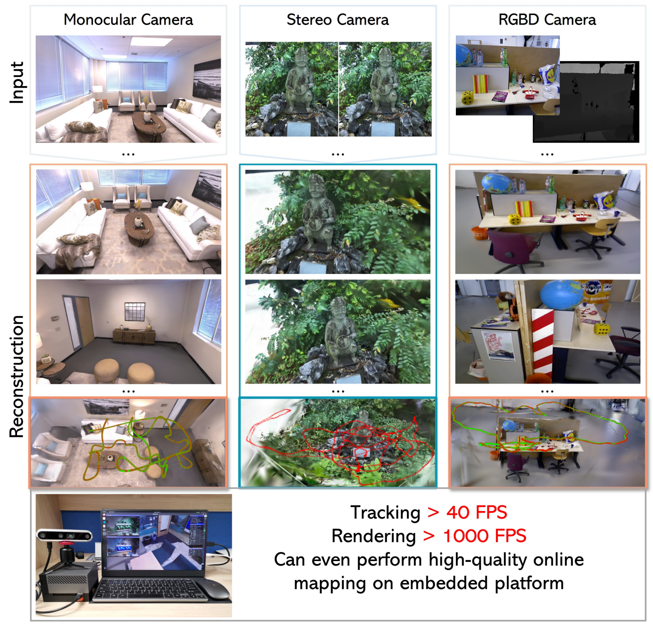


Fig. 1. Rendering and trajectory results. Photo-SLAM is a novel framework for simultaneous localization and photorealistic mapping in real-time supporting monocular, stereo, and RGB-D cameras. The proposed system has real-time performance on embedded platforms, such as Jetson AGX Orin Developer Kit.

I. 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 [1]–[4] primarily focus on geometric mapping, providing accurate but visually simplistic representations of the environment. However, recent developments in neural rendering [5], [6] have demonstrated the potential of integrating photorealistic view reconstruction into the SLAM pipeline, enhancing the perception capabilities of robotic systems.

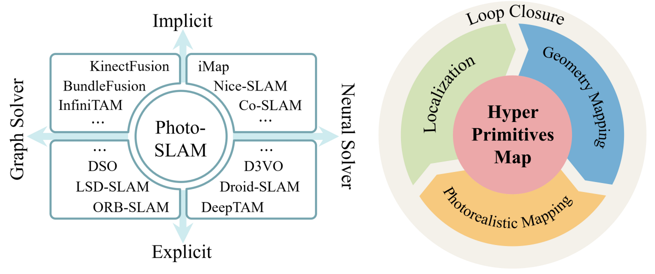
Despite the promising results achieved through the integration of neural rendering and SLAM, existing methods simply and heavily rely on implicit representations, making them computationally intensive and unsuitable for deployment on resource-constrained devices. For example, Nice-SLAM [7] leverages a hierarchical grid [8] to store learnable features representing the environment while ESLAM [9] utilizes multi-scale compact tensor components [10]. They then jointly estimate the camera poses and optimize features by minimizing the reconstruction loss of a batch of ray sampling [11]. 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 [12], or monocular depth estimators [13] to ensure efficient convergence. Additionally, since the implicit features are decoded by the multi-layer perceptrons (MLP), it is typically necessary to carefully define a bounding area to normalize ray sampling for optimal performance [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 [15], rotation, scaling, density, and spherical harmonic (SH) coefficients [16], [17]. The hyper primitives map allows the system to efficiently

Huajian Huang and Sai-Kit Yeung are with the Department of Computer Science and Engineering at the Hong Kong University of Science and Technology. Email: hhuangbg@connect.ust.hk, saikit@ust.hk
Longwei Li and Hui Cheng are with the School of Computer Science and Engineering, Sun Yat-Sen University. Email: lilw23@mail2.sysu.edu.cn, chengh9@mail.sysu.edu.cn

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, as shown in Fig. 1. The schematic overview of Photo-SLAM is demonstrated in Fig. 2.



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

Fig. 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.

II. 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 [1], [19], [20] 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 [3] and DSO [4] 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 [21] predict camera poses by the neural network [22] end-to-end, while the accuracy is limited. To enhance performance, some methods, e.g., D3VO [23] and Droid-SLAM [24], introduce monocular depth estimation [13] or dense optical flow estimation [12] 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 [25], available for training, Droid-SLAM building upon RAFT [12] 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 [26], BundleFusion [27], and InfiniTAM [28] utilize the implicit representation, Truncated Signed Distance Function (TSDF) [29], 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) [11] 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 [7] introduces multi-resolution grids [8] to store features reducing the cost of deep MLP query. Co-SLAM [31] and ESLAM [9] explore Instant-NGP [32] and TensoRF [10] 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. 2(a) 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.

III. Photo-SLAM

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

A. 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 [15] $\mathbf{O} \in \mathbb{R}^{256}$, 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 triangulation, 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.

B. 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 $i \in \mathscr{X}$ be the index of set of matches $\mathscr{X}$, what we are trying to optimize with LM is

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

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 $\mathscr{P}_L$ and keyframes $\mathscr{K}_L$. The keyframes are selected frames from the input camera sequence and provide good visual information. We fix the poses of keyframes $\mathscr{K}_F$ which are also observing $\mathscr{P}_L$ but not in $\mathscr{K}_L$. Let $\mathscr{K} = \mathscr{K}_L \cup \mathscr{K}_F$, and $\mathscr{X}_k$ be the set of matches between 2D keypoints in a keyframe k and 3D points in $\mathscr{P}_L$. The optimization process aims to reduce the geometric inconsistency between $\mathscr{K}$ and $\mathscr{P}_L$, and is defined as

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

with reprojection residual

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

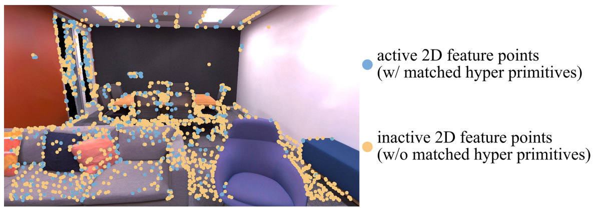

active 2D feature points (w/ matched hyper primitives)
inactive 2D feature points (w/o matched hyper primitives)

Fig. 3. We make use of initial geometric information to densify hyper primitives.

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

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

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 $\sigma_i \cdot \mathscr{G}(\mathbf{R},\mathbf{t},\mathbf{P}_i,\mathbf{r}_i,\mathbf{s}_i)$, $\mathscr{G}$ 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 $\mathscr{L}$ between rendering image I_r and ground truth image I_gt, denoted as

$$\mathscr{L} = (1-\lambda)\left|I_\mathrm{r} - I_\mathrm{gt}\right|_1 + \lambda(1 - \mathrm{SSIM}(I_\mathrm{r}, I_\mathrm{gt})), \qquad (4)$$

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

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. 3. 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 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.

D. *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 [33] and DVGO [34] 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. To enhance performance with multi-resolution features, NGLoD [35] progressively trains multiple MLPs as encoders and decoders, while only retaining the final decoder to decode integrated multi-resolution features. Furthermore, Neuralangelo [36] only maintains a single MLP during training. It progressively activates different levels of hash tables [32] 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 models in these methods is supervised by the same training images. Conversely, BungeeNeRF [37] 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, a new progressive training method. As illustrated in Fig. 4, 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. III-C.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

$$\begin{aligned} & t_0 : \arg\min \mathscr{L}\left(I_r^n, \mathrm{GP}^n(I_{gt})\right), \\ & t_1 : \arg\min \mathscr{L}\left(I_r^{n-1}, \mathrm{GP}^{n-1}(I_{gt})\right), \\ & \quad \ldots \\ & t_n : \arg\min \mathscr{L}\left(I_r^0, \mathrm{GP}^0(I_{gt})\right), \end{aligned} \tag{5}$$

where $\mathscr{L}(I_r, \mathrm{GP}(I_{gt}))$ is Eq. 4, while $\mathrm{GP}^n(I_{gt})$ 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.

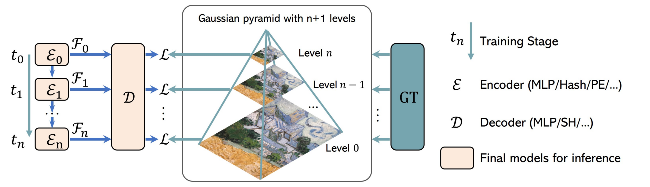


Fig. 4. We proposed a new method based on the Gaussian pyramid to efficiently learn multi-level features.

TABLE I
QUANTITATIVE RESULTS ON THE REPLICA DATASET. WE MARK THE BEST TWO RESULTS WITH **FIRST** AND SECOND.

On Replica Dataset		Localization	Mapping			Resources			
Cam	Method	RMSE (cm)↓	PSNR↑	SSIM↑	LPIPS↓	Operation Time↓	Tracking FPS↑	Rendering FPS↑	GPU Memory↓
Mono	ORB-SLAM3 [20]	3.942	-	-	-	<1 mins	**58.749**	-	0
	DROID-SLAM [24]	0.725	-	-	-	<2 mins	35.473	-	11 GB
	Nice-SLAM* [7]	99.9415	16.311	0.720	0.439	>10 mins	2.384	0.944	12 GB
	Orbeez-SLAM [38]	-	23.246	0.790	0.336	<5 mins	49.200	1.030	6 GB
	Go-SLAM [39]	71.054	21.172	0.703	0.421	<5 mins	25.366	0.821	22 GB
	Ours (Jetson)	1.235	29.284	0.883	0.139	<5 mins	18.315	95.057	**4 GB**
	Ours (Laptop)	**0.713**	33.049	**0.926**	0.086	<5 mins	19.974	353.504	**4 GB**
	Ours	1.091	**33.302**	**0.926**	**0.078**	<2 mins	41.646	**911.262**	6 GB
RGB-D	ORB-SLAM3 [20]	1.833	-	-	-	<1 mins	**52.209**	-	0
	DROID-SLAM [24]	0.634	-	-	-	<2 mins	36.452	-	11 GB
	BundleFusion [27]	1.606	23.839	0.822	0.197	<5 mins	8.630	-	5 GB
	Nice-SLAM [7]	2.350	26.158	0.832	0.232	>10 mins	2.331	0.611	12 GB
	Orbeez-SLAM [38]	0.888	32.516	0.916	0.112	<5 mins	41.333	1.401	6 GB
	ESLAM [9]	**0.568**	30.594	0.866	0.162	<5 mins	6.687	2.626	21 GB
	Co-SLAM [31]	1.158	30.246	0.864	0.175	<5 mins	14.575	3.745	**4 GB**
	Go-SLAM [39]	0.571	24.158	0.766	0.352	<5 mins	19.437	0.444	24 GB
	Point-SLAM [40]	0.596	34.632	0.927	0.083	>2 hrs	0.345	0.510	24 GB
	Ours (Jetson)	0.581	31.978	0.916	0.101	<5 mins	17.926	116.395	**4 GB**
	Ours (Laptop)	0.590	34.853	**0.944**	0.062	<5 mins	20.597	396.082	**4 GB**
	Ours	0.604	**34.958**	0.942	**0.059**	<2 mins	42.485	**1084.017**	5 GB

IV. EXPERIMENT

We implemented Photo-SLAM fully in C++ and CUDA, making use of ORB-SLAM3 [20], 3D Gaussian splatting [18], and the LibTorch framework. We ran Photo-SLAM and all compared methods using their official code on a desktop with an NVIDIA RTX 4090 GPU. We further tested Photo-SLAM on a laptop and a Jetson AGX Orin Developer Kit. As quantitative comparison demonstrated in Table I, 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.

V. 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 to further enhance mapping performance. 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.

REFERENCES

[1] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[2] C. Forster, M. Pizzoli, and D. Scaramuzza, "Svo: Fast semi-direct monocular visual odometry," in *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2014, pp. 15–22.

[3] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *European conference on computer vision*. Springer, 2014, pp. 834–849.

[4] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 3, pp. 611–625, 2017.

[5] A. Tewari, J. Thies, B. Mildenhall, P. Srinivasan, E. Tretschk, W. Yifan, C. Lassner, V. Sitzmann, R. Martin-Brualla, S. Lombardi, *et al.*, "Advances in neural rendering," in *Computer Graphics Forum*, vol. 41, no. 2. Wiley Online Library, 2022, pp. 703–735.

[6] Y. Xie, T. Takikawa, S. Saito, O. Litany, S. Yan, N. Khan, F. Tombari, J. Tompkin, V. Sitzmann, and S. Sridhar, "Neural fields in visual computing and beyond," in *Computer Graphics Forum*, vol. 41, no. 2. Wiley Online Library, 2022, pp. 641–676.

[7] Z. Zhu, S. Peng, V. Larsson, W. Xu, H. Bao, Z. Cui, M. R. Oswald, and M. Pollefeys, "Nice-slam: Neural implicit scalable encoding for slam," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 786–12 796.

[8] A. Yu, R. Li, M. Tancik, H. Li, R. Ng, and A. Kanazawa, "Plenoctrees for real-time rendering of neural radiance fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5752–5761.

[9] M. M. Johari, C. Carta, and F. 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*, 2023, pp. 17 408–17 419.

[10] A. Chen, Z. Xu, A. Geiger, J. Yu, and H. Su, "Tensorf: Tensorial radiance fields," in *European Conference on Computer Vision (ECCV)*, 2022.

[11] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," in *European conference on computer vision*. Springer, 2020, pp. 405–421.

[12] Z. Teed and J. 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*. Springer, 2020, pp. 402–419.

[13] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 270–279.

[14] H. Huang, Y. Chen, T. Zhang, and S.-K. Yeung, "360roam: Real-time indoor roaming using geometry-aware 360° radiance fields," *arXiv preprint arXiv:2208.02705*, 2022.

[15] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," *IEEE International Conference on Computer Vision*, vol. 58, no. 11, pp. 2564–2571, 2011.

[16] S. Wizadwongsa, P. Phongthawee, J. Yenphraphai, and S. Suwajanakorn, "Nex: Real-time view synthesis with neural basis expansion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 8534–8543.

[17] S. Fridovich-Keil, A. Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa, "Plenoxels: Radiance fields without neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5501–5510.

[18] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering," *ACM Transactions on Graphics (ToG)*, vol. 42, no. 4, pp. 1–14, 2023.

[19] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE transactions on robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[20] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[21] H. Zhou, B. Ummenhofer, and T. Brox, "Deeptam: Deep tracking and mapping," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 822–838.

[22] A. Kendall, M. Grimes, and R. Cipolla, "Posenet: A convolutional network for real-time 6-dof camera relocalization," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2938–2946.

[23] N. Yang, L. v. Stumberg, R. Wang, and D. 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*, 2020, pp. 1281–1292.

[24] Z. Teed and J. Deng, "Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras," *Advances in neural information processing systems*, vol. 34, pp. 16 558–16 569, 2021.

[25] W. Wang, D. Zhu, X. Wang, Y. Hu, Y. Qiu, C. Wang, Y. Hu, A. Kapoor, and S. Scherer, "Tartanair: A dataset to push the limits of visual slam," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 4909–4916.

[26] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. 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*, 2011, pp. 559–568.

[27] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt, "Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration," *ACM Transactions on Graphics (ToG)*, vol. 36, no. 4, p. 1, 2017.

[28] 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.

[29] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996, pp. 303–312.

[30] E. Sucar, S. Liu, J. Ortiz, and A. J. Davison, "imap: Implicit mapping and positioning in real-time," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 6229–6238.

[31] H. Wang, J. Wang, and L. 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*, 2023, pp. 13 293–13 302.

[32] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Transactions on Graphics (ToG)*, vol. 41, no. 4, pp. 1–15, 2022.

[33] L. Liu, J. Gu, K. Zaw Lin, T.-S. Chua, and C. Theobalt, "Neural sparse voxel fields," *Advances in Neural Information Processing Systems*, vol. 33, pp. 15 651–15 663, 2020.

[34] C. Sun, M. Sun, and H.-T. 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*, 2022, pp. 5459–5469.

[35] T. Takikawa, J. Litalien, K. Yin, K. Kreis, C. Loop, D. Nowrouzezahrai, A. Jacobson, M. McGuire, and S. 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*, 2021, pp. 11 358–11 367.

[36] Z. Li, T. Müller, A. Evans, R. H. Taylor, M. Unberath, M.-Y. Liu, and C.-H. Lin, "Neuralangelo: High-fidelity neural surface reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 8456–8465.

[37] Y. Xiangli, L. Xu, X. Pan, N. Zhao, A. Rao, C. Theobalt, B. Dai, and D. Lin, "Bungeenerf: Progressive neural radiance field for extreme multi-scale scene rendering," in *European conference on computer vision*. Springer, 2022, pp. 106–122.

[38] C.-M. Chung, Y.-C. Tseng, Y.-C. Hsu, X.-Q. Shi, Y.-H. Hua, J.-F. Yeh, W.-C. Chen, Y.-T. Chen, and W. 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)*. IEEE, 2023, pp. 9400–9406.

[39] Y. Zhang, F. Tosi, S. Mattoccia, and M. Poggi, "Go-slam: Global optimization for consistent 3d instant reconstruction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 3727–3737.

[40] E. Sandström, Y. Li, L. Van Gool, and M. R. Oswald, "Point-slam: Dense neural point cloud-based slam," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.