2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

EC3R-SLAM: Efficient and Consistent Monocular Dense SLAM with Feed-Forward 3D Reconstruction

Lingxiang Hu[1], Naima Ait Oufroukh[1], Fabien Bonardi[1], and Raymond Ghandour[2]

***Abstract*— The application of monocular dense Simultaneous Localization and Mapping (SLAM) is often hindered by high latency, large GPU memory consumption, and reliance on camera calibration. To relax this constraint, we propose EC3R-SLAM, a novel calibration-free monocular dense SLAM framework that jointly achieves high localization and mapping accuracy, low latency, and low GPU memory consumption. This enables the framework to achieve efficiency through the coupling of a tracking module, which maintains a sparse map of feature points, and a mapping module based on a feed-forward 3D reconstruction model that simultaneously estimates camera intrinsics. In addition, both local and global loop closures are incorporated to ensure mid-term and long-term data association, enforcing multi-view consistency and thereby enhancing the overall accuracy and robustness of the system. Experiments across multiple benchmarks show that EC3R-SLAM achieves competitive performance compared to state-of-the-art methods, while being faster and more memory-efficient. Moreover, it runs effectively even on resource-constrained platforms such as laptops and Jetson Orin NX, highlighting its potential for real-world robotics applications. Project page: `https://h0xg.github.io/ec3r/`**

I. Introduction

Monocular dense SLAM has found wide applications in autonomous driving, robotic navigation, and AR/VR [1], [2]. Existing approaches exploit neural priors [3]–[7] to recover dense geometry from monocular input. Although effective, these methods often suffer from high GPU memory usage [3], [4], [8], significant latency [1], [6], [8], [9], and complex optimization pipelines [1], [5], which limit their suitability for real-time deployment. More recently, a new class of 3D reconstruction models [10]–[14] has demonstrated the ability to recover dense geometry directly from uncalibrated RGB frames. Among them, VGGT [13] and Fast3R [14] are able to reconstruct thousands of images within seconds, but their high GPU memory requirements render them unsuitable for use on consumer-grade hardware [4], [15].

To overcome these limitations, we propose EC3R-SLAM, an **E**fficient and **C**onsistent **3**D **R**econstruction framework for calibration-free monocular dense SLAM. Our framework employs lightweight feature-based tracking to select keyframes, while only a small subset of frames (five in our implementation) is forwarded to VGGT for feed-forward reconstruction, generating local submaps that are subsequently fused into a global map. This strategy preserves the efficiency of feed-forward models while markedly reducing computational and memory demands: our pipeline operates with less than 10 GB of GPU memory and runs at more than 30 FPS. By comparison, the concurrent work VGGT–SLAM [4] feeds 32 frames at once, whereas VGGT-Long [15] feeds more than 60 frames at once into VGGT, usually pushing memory usage beyond 20GB, which constrains practical deployment on resource-constrained robotic platforms. However, fusing multiple small submaps can introduce severe inconsistencies, a limitation also observed in VGGT–SLAM and MASt3R–SLAM [3] (see Fig. 1a). To mitigate this, we incorporate both local and global loop closure modules that establish mid-term and long-term associations, thereby enforcing multi-view consistency and alleviating misalignment. As a result, EC3R-SLAM achieves accurate, real-time, and resource-efficient dense mapping and camera pose estimation, as illustrated in Fig. 1b.

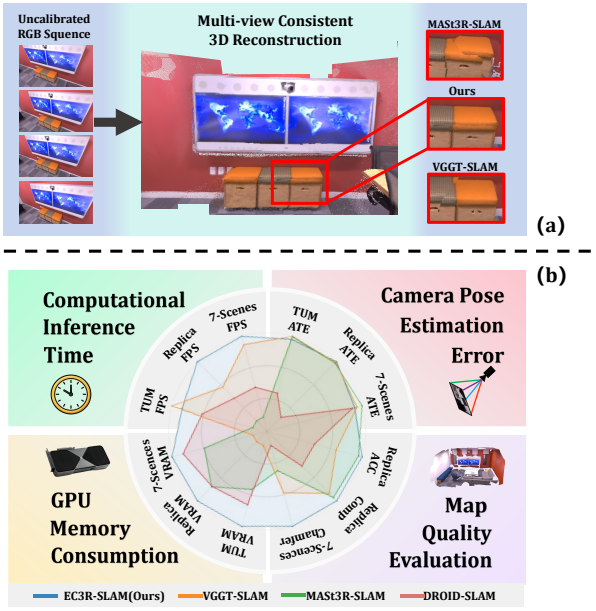

Fig. 1: (a) Our method achieves real-time multi-view consistent 3D reconstruction from uncalibrated RGB sequence. (b) Benchmark results show fast inference and low GPU memory use with competitive accuracy, highlighting its efficiency.

In summary, our main contributions are:

1) We present, for the first time, an innovative method that combines lightweight feature point-based tracking with feed-forward 3D reconstruction model-based

[1]Lingxiang Hu, Naima Ait Oufroukh, and Fabien Bonardi are with IBISC, Université Évry Paris-Saclay, 91000 Évry, France. (Corresponding author: Lingxiang Hu.) E-mail: hulxhlx@gmail.com, naima.aitoufroukh@univ-evry.fr, fabien.bonardi@univ-evry.fr

[2]Raymond Ghandour is with the College of Engineering and Technology, American University of the Middle East, Kuwait. E-mail: Raymond.Ghandour@aum.edu.kw

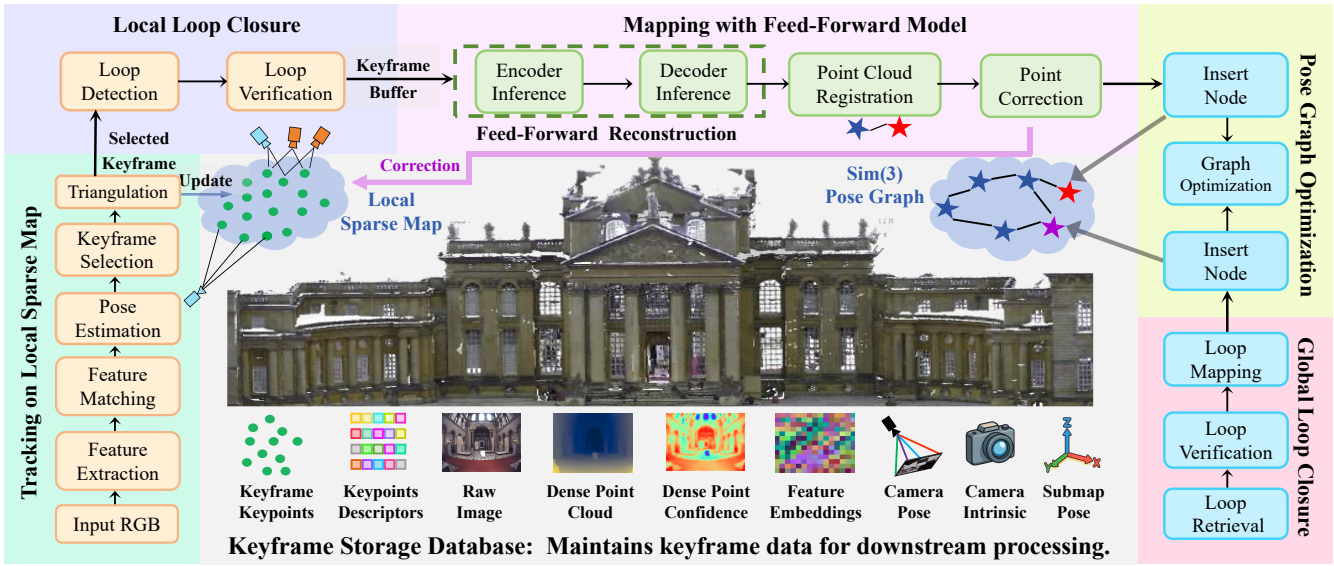

Fig. 2: System overview. The RGB images are first processed in the tracking module, where keyframes are selected and used for local loop closure to identify similar frames. The verified keyframes are stored in the keyframe buffer, and once a sufficient number of keyframes are accumulated, they are passed to the mapping module to generate reconstruction information, which is stored in the database. At the same time, the global loop closure module retrieves features from the database for loop detection and performs pose graph optimization.

mapping, enabling an efficient pipeline for localization and mapping.

2) We propose a novel data association strategy that integrates local loop and global loop closure, and injects loop information into pose graph optimization, thereby enhancing multi-view consistency and significantly improving the overall accuracy of the system.
3) By unifying these components, we propose a new uncalibrated monocular dense SLAM framework that delivers competitive results across multiple benchmarks, while maintaining low GPU memory usage and real-time performance.

II. Related Works

We first review recent works on how decoupled tracking and mapping can improve the efficiency of SLAM. We then introduce approaches that leverage data association to ensure map consistency. Finally, we review recent developments in 3D reconstruction models.

A. Decoupled Tracking and Mapping

To improve efficiency, many SLAM systems adopt a decoupled design in which tracking and mapping are handled separately. The frontend typically employs lightweight methods for camera tracking and keyframe selection [5], [16], [17], while the backend focuses on constructing dense maps. For instance, Orbeez-SLAM [18] leverages ORB-SLAM2 [16] for tracking to construct an implicit neural map, while other methods [6]–[9] employ DROID-SLAM [5] to enable dense mapping. However, the loose coupling between tracking and mapping in these systems often leads to suboptimal information usage and increased computational overhead. In contrast, our framework tightly couples an ultra-lightweight tracking module with mapping, substantially improving both efficiency and accuracy.

B. Mid-Term and Long-Term Data Association

Mid-term data association links frames captured from nearby camera positions. For example, ORB-SLAM achieves this by projecting map points into the estimated camera pose. Long-term data association, conversely, relies on place recognition techniques, which can be implemented using either traditional bag-of-words models [19] or learning-based approaches such as NetVLAD [20] and SALAD [21]. While many existing frameworks employ only one type of data association [1], [3]–[5], our approach integrates both. Moreover, long-term associations are directly obtained from the embeddings of our feed-forward reconstruction model, eliminating the need for extra place recognition networks. This design further improves accuracy while preserving real-time efficiency.

C. Uncalibrated reconstruction

DUSt3R [11] and MASt3R [12] pioneered this direction, but multi-view reconstruction still required time-consuming post-processing. Subsequent methods such as Spann3R [10], SLAM3R [22], CUT3R [23], and MASt3R-SLAM [3] extended these to sequence-based reconstruction. While these approaches substantially improve real-time capability, they rely on computationally expensive dense neural inference, as the network must process every frame. VGGT-SLAM mitigates this by selecting keyframes via optical flow and applying a feed-forward model for reconstruction, yet it still incurs high GPU memory consumption. In our method, the tight coupling of tracking and mapping ensures that dense inference is avoided and memory consumption is minimized, thereby enabling efficient and scalable reconstruction.

III. Method

Figure 2 provides an overview of our system architecture. In the following sections, we detail each component of our

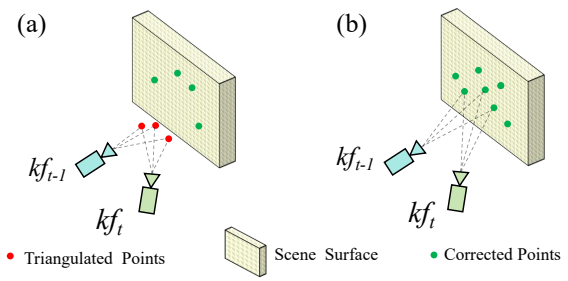

Fig. 3: Illustration of point correction. (a) Before correction. (b) After point correction .

framework, including tracking (III-A), local loop closure (III-B), mapping (III-C), global loop closure (III-D), and pose graph optimization (III-E).

A. Tracking on the Local Sparse Map

In this stage, we extract and match visual features across frames, perform pose estimation, and select the keyframes. Unlike traditional approaches, these operations are carried out on the local sparse map, which serves as the basis for maintaining robust and efficient tracking.

Local Sparse Map. Tracking in our system relies on a local sparse map, which stores the 3D points corresponding to the keypoints of the current keyframe. These points are updated whenever a new keyframe is selected and are continuously corrected during the mapping process.

Initialization. Initially, a set of selected N frames is forwarded to the mapping module to estimate the camera intrinsics and perform an initial 3D reconstruction, resulting in the creation of an initial local sparse map.

Per-Frame Tracking. For each incoming frame, we employ XFeat [24], an efficient learning-based feature matching network, to extract keypoints and their descriptors. The descriptors of the current frame are matched with 3D points in the local sparse map $\{\mathbf{X}_i\}_{i=1}^{N}$, yielding 2D–3D correspondences $\{(\mathbf{x}_i, \mathbf{X}_i)\}_{i=1}^{N}$. Given these correspondences, the camera pose $(\mathbf{R}, \mathbf{t})$ of the current frame is estimated by solving a Perspective-n-Point (PnP) problem that minimizes the reprojection error:

$$\min_{\mathbf{R},\mathbf{t}} \sum_{i=1}^{N} \|\pi(\mathbf{R}\mathbf{X}_i + \mathbf{t}) - \mathbf{x}_i\|^2, \tag{1}$$

where $\pi(\cdot)$ denotes the perspective projection function. To ensure robustness to outliers in the correspondences, we solve the problem using a RANSAC-based PnP algorithm [25], [26], from which the inlier ratio τ_{inlier} is computed.

Keyframe Selection and Triangulation. If the inlier ratio τ_{inlier} falls below a threshold τ_1, or if pose estimation fails, indicating degraded tracking quality, the current frame is promoted to a new keyframe. Once selected as a keyframe, triangulation is performed to add new points into the local sparse map.

B. Local Loop Closure for Mid-Term Data Association

To enforce mid-term data association, we integrate a local loop closure mechanism immediately after the tracking stage, whose main purpose is to associate the current keyframe with spatially nearby frames.

Loop Detection. Potential local loop frames are identified by projecting the current local sparse map onto a set of temporally adjacent keyframes, as illustrated in Fig. 2. This set is defined as

$$\{\text{KF}_k\}_{k=i-r_{\text{local}}}^{i+r_{\text{local}}}, \quad \text{where } i + r_{\text{local}} = n - 2N_{\text{map}},$$

with n denoting the index of the current keyframe. A keyframe within this set is regarded as a loop closure candidate if the number of 3D points from the local sparse map projected onto its image plane exceeds a predefined threshold τ_p.

Loop Verification. To verify loop-closure candidates, we compute the inlier ratio r_{inlier} using RANSAC homography estimation. If $r_{\text{inlier}} \leq \tau_2$, the candidate is considered dissimilar and discarded. If $r_{\text{inlier}} > \tau_2$, the candidate is appended to the keyframe buffer for further processing. In particular, when $r_{\text{inlier}} > \tau_1$, the candidate is regarded as highly similar, and the current keyframe is replaced by the loop candidate.

Keyframe Buffer. The keyframe buffer consists of two components: (1) new keyframes obtained from the tracking module, and (2) old frames, including loop closure candidates and the most recent keyframe from the previous buffer. Once the number of buffered frames exceeds the predefined threshold N, the buffer is flushed, and all accumulated keyframes are forwarded to the mapping module for dense reconstruction. After flushing, the buffer is re-initialized with the latest keyframe to ensure continuity for subsequent processing.

C. Feed-Forward Model Reconstruction

This module runs in a separate process parallel to tracking and enables the system to construct a dense point cloud of the scene. We adopt VGGT [13] to infer a local submap, which is subsequently aligned with the global map.

Feed-Forward Inference. To improve efficiency, encoder inference is only performed on new keyframes from the keyframe buffer. Given the RGB image $\mathbf{I}_1$ of a new keyframe, the image encoder $\mathscr{E}(\cdot)$ produces image feature embeddings:

$$\mathbf{E}_1 = \mathscr{E}(\mathbf{I}_1), \tag{2}$$

where $\mathbf{E}_1 = \{\mathbf{e}_1, \cdots, \mathbf{e}_k\}$ denotes the keyframe feature embeddings. For old keyframes, their embeddings $\mathbf{E}_2 = \{\mathbf{e}_{k+1}, \cdots, \mathbf{e}_n\}$ are directly retrieved from the keyframe storage database.

The embeddings $\mathbf{E}_1$ and $\mathbf{E}_2$ are concatenated and fed into the prediction decoder $\mathscr{D}(\cdot)$, which in the VGGT architecture refers to the remaining part of the network after the image encoder:

$$\mathbf{D}, \mathbf{C}, \mathbf{g} = \mathscr{D}\left(\text{Concat}\left(\mathbf{E}_1, \mathbf{E}_2\right)\right), \tag{3}$$

where $\mathbf{D}$ denotes the predicted depth maps, $\mathbf{C}$ the corresponding confidence maps that quantify the reliability of

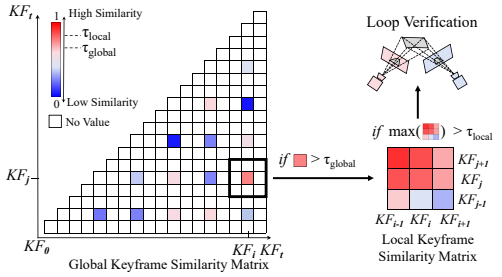

Fig. 4: We perform loop detection by computing a similarity matrix and filtering it with global and local thresholds, followed by homography-based verification.

each depth estimate, and $\mathbf{g}$ the camera parameters (intrinsics and extrinsics). Following [4], [13], the current submap is obtained by inverse-projecting the estimated depths $\mathbf{D}$ using the projection matrices derived from $\mathbf{g}$. This submap consists of a dense point cloud, where each pixel in the keyframe corresponds to a 3D point with an associated confidence score. The submap is defined with respect to the coordinate frame of the first camera.

Sim3 Point Cloud Registration. Current submap contains a subset of 3D points $\mathbf{P} = \{\mathbf{p}_i\}_{i=1}^{N}$ originating from old keyframes. These points can be associated with their counterparts $\mathbf{Q} = \{\mathbf{q}_i\}_{i=1}^{N}$ in other submaps that belong to the same old keyframes, thereby forming reliable 3D–3D correspondences. Their corresponding confidence scores are normalized into weights $\mathbf{w} = \{w_i\}_{i=1}^{N}$. We then estimate the optimal Sim(3) transformation $\mathbf{S} = \{s, \mathbf{R}, \mathbf{t}\} \in \mathrm{Sim}(3)$ by minimizing the weighted alignment error:

$$\min_{s,\mathbf{R},\mathbf{t}} \sum_{i=1}^{N} w_i \left\| s\mathbf{R}\,\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i \right\|^2, \tag{4}$$

where $s \in \mathbb{R}^+$ is a scale factor, $\mathbf{R} \in \mathrm{SO}(3)$ a rotation, and $\mathbf{t} \in \mathbb{R}^3$ a translation. We solve (4) using a weighted extension of Umeyama's closed-form Sim(3) algorithm [27]. This step yields the transformation between the current submap and other connected submaps, which is then incorporated into the pose graph as a relative Sim(3) constraint.

Point Correction. Through the estimated transformations between submaps, we obtain the global coordinates of all points in the current submap, which are regarded as accurate. A subset of these points has correspondences with those in the local sparse map. For such points, we replace their coordinates with the accurate global ones, as illustrated in Fig. 3, thereby reducing the accumulated drift of tracking. Moreover, to further prevent the frontend from being lost, whenever the number of frames in the keyframe buffer reaches 2, we trigger an additional mapping operation. This operation is solely used to obtain the current keyframe's points relative to the global frame, which are then applied for correction.

Keyframe Storage Database. As illustrated in Fig. 2, the keyframe storage database maintains all keyframe information obtained during the tracking and mapping stages. This information is later exploited for loop detection and loop retrieval. To save computational resources, the database is stored on the CPU.

D. Global Loop Closure for Long-Term Data Association

We employ a novel approach for global loop closure. This module runs in a separate thread and leverages the keyframe features extracted in Section III-C to detect global loop closures, thereby establishing long-term associations in the global map. An overview of the pipeline is presented in Fig. 4.

Loop Retrieval and Verification. We first construct a sparse similarity matrix by measuring feature similarity between every N-th keyframe stored in the database. If two keyframes are sufficiently similar, their neighboring frames are also considered and incorporated into the similarity matrix. When the maximum similarity score exceeds the threshold τ_{local}, we further verify the candidate pair by estimating a homography and counting the inliers ratio. If this verification succeeds, a loop closure is confirmed.

Loop Mapping. We select the embeddings of the top-N most similar keyframes from the local similarity matrix. These embeddings are passed through the decoder inference of the mapping module (Section III-C) to generate a submap, which is subsequently registered and inserted into the pose graph.

E. Pose Graph Optimization

The relative Sim(3) constraints T_{ij} obtained in Section III-C.2 are incorporated as edges into a submap-level pose graph, where each node represents the global pose of a submap $T_t \in \mathrm{Sim}(3)$ in the world coordinate frame. The goal of pose graph optimization is to jointly refine all submap poses by enforcing consistency across the 3D–3D correspondences.

During optimization, each transformation T is mapped to a minimal representation in $\mathbb{R}^7$ of the associated Lie algebra using the logarithmic mapping function $\log_{\mathrm{Sim}(3)}(\cdot)$. Given the constructed pose graph, the error on an edge (i, j) is defined as

$$e_{i,j} = \log_{\mathrm{Sim}(3)}\left(T_{ij}^{-1} T_i^{-1} T_j\right). \tag{5}$$

The overall objective is to minimize the total energy

$$\chi^2(T_1, \ldots, T_m) = \sum_{(i,j)\in\mathcal{E}} e_{i,j}^{\top} \Omega_{i,j}\, e_{i,j}, \tag{6}$$

where $\Omega_{i,j}$ denotes the information matrix associated with the measurement. The absolute submap poses $\{T_t\}$ are initialized by chaining the relative Sim(3) transformations, and the optimization is carried out using the Levenberg–Marquardt algorithm implemented in *PyPose* [34], which enables efficient Lie-group optimization on Sim(3). This process yields globally consistent submap poses and effectively reduces accumulated drift.

IV. Experiment

We first introduce the experimental setup, followed by a detailed analysis of localization and mapping results, runtime, GPU memory consumption, and ablation studies.

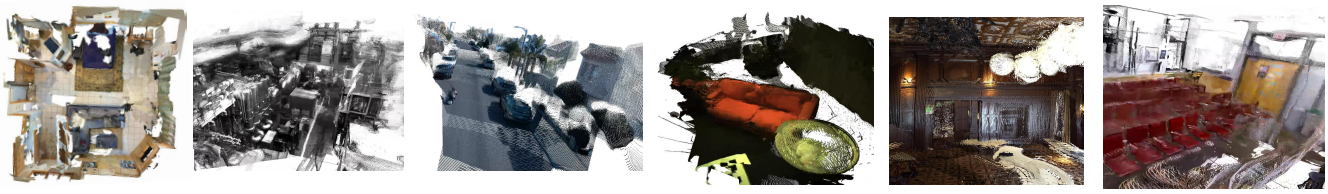

Fig. 5: EC3R-SLAM can generalize to new datasets. From left to right, we show results from ScanNet [28], EuRoC [29], Waymo open [30], ETH3D [31], Tanks and Temples [32], and SUN3D [33].

TABLE I: Evaluation of uncalibrated dense monocular SLAM methods on the TUM-RGBD dataset.

Map type	Method	Sequence									Avg↓	VRAM↓	FPS↑
		360	desk	desk2	floor	plant	room	rpy	teddy	xyz			
NeRF	GlORIE-SLAM	0.194	0.028	0.105	0.062	0.036	0.871	0.051	0.042	0.014	0.155	18.9	<1
	GO-SLAM	0.195	0.031	0.198	0.077	0.049	0.865	0.058	0.049	0.014	0.171	20.8	7
3DGS	Photo-SLAM	0.165	0.028	0.880	**0.055**	0.705	0.924	**0.027**	0.933	**0.013**	0.414	7.6	23
	Mono-GS	0.159	0.043	0.689	0.603	0.616	0.726	0.041	0.100	0.022	0.333	13.0	2
	Splat-SLAM	0.205	0.026	0.101	0.062	0.038	0.879	0.051	0.042	0.014	0.158	18.8	1
	Hi-SLAM2	0.223	0.068	12.62	0.305	0.059	2.204	0.035	0.213	0.033	1.750	22.1	3
Point Cloud	DROID-SLAM	0.194	0.034	0.822	0.168	0.038	0.975	0.056	0.059	0.019	0.263	13.1	23
	MASt3R-SLAM	0.070	0.035	0.055	0.056	0.035	0.119	0.041	0.115	0.019	0.061	15.8	13
	VGGT-SLAM	**0.064**	**0.024**	**0.036**	0.126	**0.023**	0.163	0.032	**0.037**	0.017	**0.059**	23.5	**34**
	Ours(w.Fast3R)	0.129	0.056	0.091	0.067	0.113	0.145	0.052	0.143	0.022	0.091	**7.1**	27
	Ours	0.101	0.038	0.050	**0.055**	0.097	**0.101**	0.045	0.125	0.018	0.070	9.3	31

TABLE II: Evaluation of uncalibrated dense monocular SLAM methods on the Replica dataset.

Map type	Method	Sequence								Avg↓	VRAM↓	FPS↑
		Room0	Room1	Room2	Office0	Office1	Office2	Office3	Office4			
NeRF	GlORIE-SLAM	0.374	0.119	0.117	0.337	0.419	0.050	0.472	0.282	0.234	17.9	<1
	GO-SLAM	0.378	0.118	0.098	0.033	0.416	0.059	0.543	0.323	0.246	16.2	12
3DGS	Photo-SLAM	0.363	0.139	0.056	0.030	0.423	0.094	0.313	0.382	0.222	9.4	2
	Mono-GS	0.716	0.518	0.167	0.231	0.439	0.296	0.269	0.679	0.456	12.3	2
	Splat-SLAM	0.374	0.119	0.107	0.034	0.418	0.052	0.472	0.287	0.233	22.6	4
	Hi-SLAM2	0.314	0.113	0.116	0.031	0.418	0.040	0.253	0.249	0.192	15.2	16
Point Cloud	DROID-SLAM	0.313	0.111	0.125	**0.029**	0.421	0.045	0.253	0.249	0.193	13.1	28
	MASt3R-SLAM	**0.023**	0.035	0.103	0.038	0.033	0.047	0.035	0.045	0.045	13.0	20
	VGGT-SLAM	0.030	0.082	0.057	0.061	**0.028**	**0.024**	0.028	**0.032**	0.043	25.4	36
	Ours(w.Fast3R)	0.042	**0.030**	0.077	0.129	0.042	0.051	0.035	0.074	0.060	**6.8**	35
	Ours	0.038	0.049	**0.027**	0.043	0.034	0.059	**0.025**	0.053	**0.041**	9.1	**45**

TABLE III: Comparison of reconstruction quality and camera pose accuracy on the 7-Scenes dataset.

Method	Acc ↓	Comp ↓	Cham ↓	ATE ↓	VRAM ↓	FPS ↑
Spann3R@20*	0.069	0.047	0.058	–	–	–
Spann3R@2*	0.124	**0.043**	0.084	–	–	–
SLAM3R	0.038	0.070	0.054	0.084	–	–
DROID-SLAM	0.099	0.057	0.078	0.078	11.2	24
MASt3R-SLAM	0.065	0.067	0.056	**0.065**	15.2	17
VGGT-SLAM	0.052	0.059	0.056	0.072	23.4	34
Ours (w. Fast3R)	0.044	0.076	0.060	0.090	**7.0**	32
Ours	**0.025**	0.054	**0.040**	0.075	9.1	**36**

@n indicates a keyframe every n images.
* indicates results reported in MASt3R-SLAM.

A. Experimental Setup

Implementation Details. We implement EC3R-SLAM in Python and adopt all neural network models directly from their official implementations. Unless otherwise specified, all experiments were conducted on a desktop equipped with an NVIDIA GeForce RTX 5090 GPU and an Intel Core i9-12900KF CPU. In addition, we replace VGGT with Fast3R [14] across the entire system and conduct the same set of experiments, reported in the tables as *Ours (w. Fast3R)*.

Hyperparameters. We set the thresholds as follows: $\tau_1 = 0.4$ and $\tau_2 = 0.3$ for loop verification and keyframe selection, $\tau_p = 0.7$ for local loop detection, $\tau_{\text{global}} = 0.93$ and $\tau_{\text{local}} = 0.96$ for global loop retrival, and $N = 5$ for the keyframe buffer.

Datasets. To ensure comprehensive evaluation, we use sequences from three standard datasets: the fr1 series of TUM-RGBD [35], seq-01 from 7-Scenes [36], and 8 scenes from Replica [37]. Replica and 7-Scenes provide reliable ground-truth reconstructions, making them suitable for evaluating accuracy and completeness, while TUM-RGBD is more challenging due to handheld motion, rolling shutter, and motion blur. We do not perform subsampling on the datasets; instead, all images are resized while preserving aspect ratio, with the longer side scaled to 518 pixels.

B. Camera Pose Estimation Evaluation

Baselines. Our evaluation focuses on state-of-the-art monocular dense SLAM approaches, including point cloud-based methods such as MASt3R-SLAM [3], DROID-SLAM [5], and VGGT-SLAM [4], as well as NeRF [38]-based methods (GlORIE-SLAM [6], GO-SLAM [7]) and 3D Gaussian Splatting(3DGS) [39]-based methods (MonoGS [1], Splat-SLAM [9], Hi-SLAM2 [8]). All methods, except for MASt3R-SLAM and VGGT-SLAM, rely on known camera intrinsic parameters. To ensure a comparison

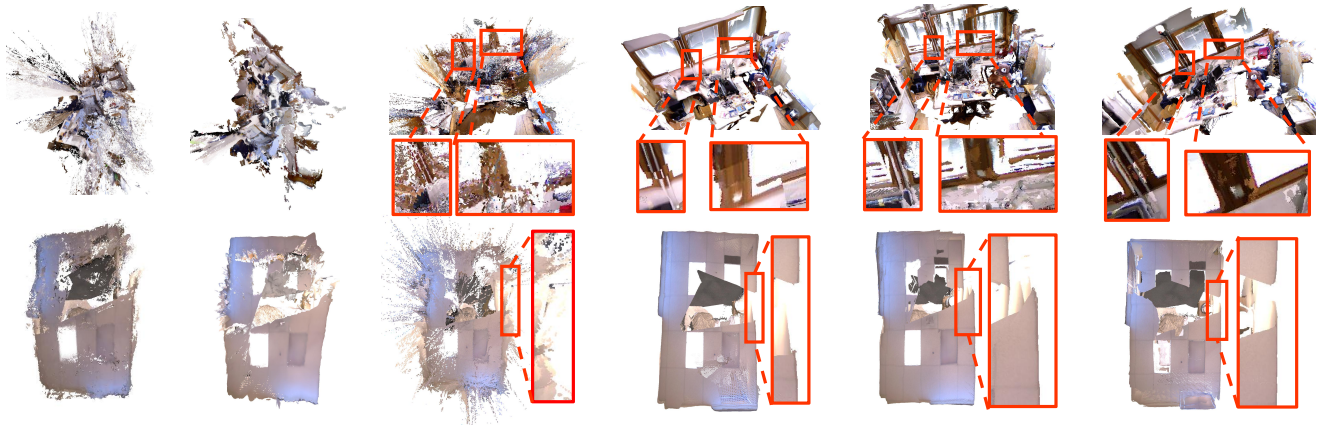

Fig. 6: Qualitative comparison on TUM-RGBD fr1/room (top) and Replica Room-1 (bottom). Our method achieves high-quality reconstruction with both local detail preservation and global structural consistency.

TABLE IV: Comparison of 3D reconstruction accuracy and completeness (in cm) on Replica dataset.

Method	Room 0		Room 1		Room 2		Office 0		Office 1		Office 2		Office 3		Office 4		Average	
	Acc.	Comp.	Acc.	Comp.	Acc.	Comp.	Acc.	Comp.	Acc.	Comp.	Acc.	Comp.	Acc.	Comp.	Acc.	Comp.	Acc.	Comp.
DUSt3R*	3.47	2.50	**2.53**	1.86	2.95	1.76	4.92	3.51	3.09	2.21	4.01	3.10	3.27	2.25	3.66	2.61	3.49	2.48
MASt3R*	4.01	4.10	3.61	3.25	3.13	2.15	**2.57**	**1.63**	12.85	8.13	3.13	1.99	4.67	3.15	3.69	2.47	4.71	3.36
SLAM3R	4.31	3.51	2.72	1.91	3.76	2.73	4.07	2.37	3.57	2.44	3.62	2.66	4.44	3.08	**3.01**	2.29	3.69	2.62
Spann3R*	9.75	12.94	15.51	12.94	7.28	8.50	5.46	18.75	5.24	16.64	9.33	11.80	16.00	9.03	13.97	16.02	10.32	13.33
Hi-SLAM2	77.78	34.73	21.83	30.33	64.64	26.39	5.74	6.87	40.82	96.74	5.47	7.09	39.36	22.59	15.37	10.69	41.58	31.97
Splat-SLAM	61.67	25.30	23.56	31.82	59.33	34.32	4.52	4.29	60.71	80.21	8.12	6.80	52.40	22.55	18.80	10.23	36.99	29.95
GlORIE-SLAM	68.92	15.22	29.16	26.69	60.27	21.66	4.61	3.69	21.55	116.11	11.71	6.10	49.78	15.03	17.39	7.74	39.49	24.39
DROID-SLAM	33.89	14.85	23.64	9.37	30.25	8.01	5.52	1.96	38.12	39.61	5.15	2.65	24.86	7.59	17.97	5.23	22.43	11.16
MASt3R-SLAM	2.55	2.01	2.66	**1.83**	2.14	1.58	2.88	1.87	3.45	2.57	2.80	2.06	3.75	2.76	3.13	**2.21**	2.92	2.25
VGGT-SLAM	3.39	2.56	7.59	3.37	2.31	1.71	5.50	2.43	22.89	15.39	**2.61**	**1.79**	2.86	2.56	3.44	2.25	6.32	4.01
Ours(w.Fast3R)	4.34	2.83	6.43	4.41	4.08	2.44	16.20	6.85	5.30	4.46	3.90	2.52	3.84	2.61	4.58	2.89	6.08	3.63
Ours	**2.07**	**1.63**	2.90	2.03	**1.93**	**1.57**	4.06	2.48	**2.17**	**1.69**	3.41	2.43	**2.51**	**2.08**	3.52	2.66	**2.82**	**2.07**

* indicates results reported in SLAM3R

under the uncalibrated setting, we follow the self-calibration protocol proposed in [3], [4]. Specifically, for methods that assume calibrated cameras, we estimate the intrinsic parameters from the first frame of each sequence using GeoCalib [40], a learning-based single-image calibration approach.

The results reported in Tables tables I to IV are all obtained under this evaluation protocol. All baselines are run with their default or recommended settings as specified in the original publications.

Metrics. We evaluate the proposed method on accuracy and efficiency. Accuracy is measured by RMSE-ATE (root-mean-square error of absolute trajectory error) [m], where the camera poses are estimated from corrected and optimized keypoints. Efficiency is evaluated in terms of average frame rate (FPS) across sequences and GPU memory (VRAM, video random-access memory) usage during sequence execution. We highlight the top-2 rankings, with bold indicating the best result and underline indicating the second best in each column.

Quantitative Analysis. We evaluate the camera pose estimation performance of our method on the TUM-RGBD, 7-Scenes, and Replica datasets, as reported in Tables I, II, and III. Without relying on ground-truth intrinsics, our method significantly outperforms both 3DGS-based and NeRF-based approaches across all three benchmarks. On the TUM-RGBD dataset, our method achieves performance on par with the state-of-the-art (SOTA) VGGT-SLAM in terms of accuracy and runtime, while requiring only half of its GPU memory (Table I). On the 7-Scenes dataset, our accuracy is slightly lower than that of MASt3R-SLAM; however, our runtime is nearly twice as fast (Table III). On the Replica dataset, our method attains SOTA accuracy while running at an impressive 45 FPS, substantially outperforming all other methods in speed (Table II). Finally, we note that replacing VGGT with Fast3R degrades both accuracy and runtime, mainly due to the poorer quality of the reconstructed point clouds. This degradation is mainly due to the poor quality of its reconstructed point clouds, which in turn hampers the efficiency of tracking. Most importantly, across all benchmarks, our approach consistently delivers low pose estimation errors with high computational efficiency, operating at real-time frame rates while consuming substantially less GPU memory than existing methods.

C. Dense Reconstruction Evaluation

Following the protocol of [10], [13], [22], we construct a ground-truth point cloud for each test sequence by back-projecting RGB pixels into 3D space using corresponding ground-truth depth maps and camera poses. The resulting reconstructed point clouds are then aligned to the ground-truth models using the Umeyama algorithm [27] for closed-form similarity transformation estimation, followed by fine registration via Iterative Closest Point (ICP) [41].

Benchmark on 7-Scenes. Following [3], [4], we evaluate reconstruction in terms of the RMSE of accuracy,

TABLE V: Average runtime of the key modules in our system (ms).

Dataset	Sequence	Prepare	Per-frame tracking				Per-keyframe			Dense mapping			Summary				
		Load Frame	Feature extraction	Feature matching	Pose estimation	Total	Triangu-lation	Local Loop	Total	Feed-forward Reconstruction	Point Cloud registration	Total	Total frame	Keyframe number	Submap number	Total time (s)	FPS
7-Scenes	chess	10.98	4.28	1.06	4.02	9.36	1.33	9.91	11.24	250.65	55.60	306.25	1000	33	12	27.6	36.13
Replica	room0	12.40	3.60	0.70	3.20	7.50	1.13	14.50	15.63	236.40	67.88	304.28	2000	37	11	44.23	45.21
TUM	fr1/room	9.98	6.81	1.86	4.22	12.89	1.99	12.68	14.67	293.33	61.76	355.09	1362	163	52	43.64	31.20

TABLE VI: Runtime Analysis on Multiple Devices

Dataset	Sequence	Method	PC	Laptop	Jetson
TUM-RGBD	fr1/room	DROID-SLAM	23	Failed	1
		MASt3R-SLAM	13	2	0.1
		Ours(w.F3R)	27	15	7
		VGGT-SLAM	34	Failed	Failed
7-Scenes	seq-01/chess	DROID-SLAM	24	Failed	1
		MASt3R-SLAM	17	2	0.1
		Ours(w.F3R)	32	18	9
		VGGT-SLAM	34	Failed	Failed

completeness, and symmetric Chamfer Distance. As shown in Table III, our method achieves highly competitive reconstruction performance, particularly excelling in accuracy. Compared to VGGT-SLAM, which is also based on the VGGT framework, our approach reduces the accuracy error by half (0.025 vs. 0.052).

Benchmark on Replica. Following the evaluation protocols in [8]–[10], [22], we evaluate 3D reconstruction quality using mean accuracy and completeness, measured in centimeters (cm). In addition to the previously introduced state-of-the-art SLAM baselines, we further compare with recent 3D reconstruction methods: SLAM3R [22], DUSt3R [11], MASt3R [12], and Spann3R [10]. As shown in Table IV, our method achieves state-of-the-art performance, surpassing all existing approaches in both accuracy and completeness. By comparison, VGGT-SLAM shows much weaker results, with errors roughly twice as large (Acc: 6.32 vs. 2.82, Comp: 4.01 vs. 2.07).

D. Time Analysis

As shown in Table V, our per-frame tracking runs at an exceptionally high speed thanks to the use of a lightweight neural network. This design choice also explains why our system is substantially more efficient than methods such as MASt3R-SLAM, Spann3R, and SLAM3R. Although the mapping module remains relatively time-consuming, it is executed in a separate thread and therefore does not affect tracking performance. In addition, the significantly higher runtime speed observed on the Replica dataset is mainly attributed to the smoother camera motion, which results in fewer detected keyframes.

E. Qualitative Results

From Figure 5, we can observe that our method demonstrates strong generalization capability across diverse datasets. Moreover, Figure 6 shows that our method achieves better multi-view consistency in reconstruction compared to other approaches.

F. Run on Multiple Devices

We include Ours (w. Fast3R) in the comparison, since it can run under the 8 GB VRAM constraint of the laptop. We then compare our method with state-of-the-art monocular dense SLAM approaches across diverse hardware platforms on real-world datasets. As shown in Table VI, evaluations are conducted on a laptop, an NVIDIA Jetson Orin NX, and a desktop PC. The laptop is equipped with an NVIDIA RTX 4060 GPU (8 GB VRAM) and an Intel i5-13500H CPU.

TABLE VII: Ablation Study on Loop Closure Configurations

Local	Global	TUM	7-Scenes			Replica		
		ATE↓	ATE↓	Acc↓	Comp↓	ATE↓	Acc↓	Comp↓
×	×	0.225	0.185	0.048	0.069	0.084	0.042	0.033
×	✓	0.122	0.153	0.033	0.056	0.066	0.031	0.023
✓	×	0.206	0.096	0.028	0.058	0.043	**0.025**	0.025
✓	✓	**0.070**	**0.075**	**0.025**	**0.054**	**0.041**	0.028	**0.021**

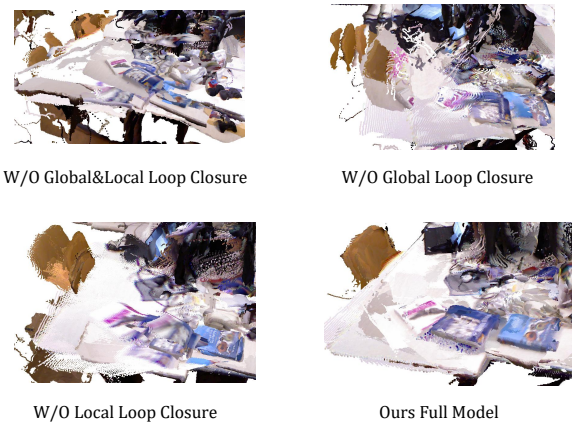

Fig. 7: Ablation study on the TUM-RGBD desk2 sequence.

G. Ablation Studies

From Table VII, we observe that both local and global loop closures play a critical role in our system, leading to substantial improvements in both localization and mapping accuracy, especially on the TUM-RGBD dataset. As shown in Figure 7, the visualization of the reconstruction on the TUM-RGBD desk2 sequence demonstrates that incorporating both local and global loop closures enables our method to achieve multi-view consistent reconstruction results, significantly reducing misalignments.

V. Conclusion

In this work, we proposed EC3R-SLAM, an efficient and consistent monocular dense SLAM system that integrates feed-forward 3D reconstruction with both local and global loop closure. Comprehensive evaluations on TUM-RGBD, 7-Scenes, and Replica datasets demonstrate that our method achieves superior accuracy compared to state-of-the-art uncalibrated dense SLAM approaches, while maintaining lower memory consumption and competitive real-time performance.

REFERENCES

[1] H. Matsuki, R. Murai, P. H. Kelly, and A. J. Davison, "Gaussian splatting slam," in *CVPR*, 2024, pp. 18 039–18 048.

[2] L. Hu, Z. Li, X. Zhu, D. Li, and R. Song, "Dpr-splat: Depth and pose refinement with sparse-view 3d gaussian splatting for novel view synthesis," in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2025, pp. 12 900–12 907.

[3] R. Murai, E. Dexheimer, and A. J. Davison, "Mast3r-slam: Real-time dense slam with 3d reconstruction priors," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 16 695–16 705.

[4] D. Maggio, H. Lim, and L. Carlone, "Vggt-slam: Dense rgb slam optimized on the sl (4) manifold," *Advances in Neural Information Processing Systems*, vol. 39, 2025.

[5] 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.

[6] G. Zhang, E. Sandström, Y. Zhang, M. Patel, L. Van Gool, and M. R. Oswald, "Glorie-slam: Globally optimized rgb-only implicit encoding point cloud slam," *arXiv preprint arXiv:2403.19549*, 2024.

[7] 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.

[8] W. Zhang, Q. Cheng, D. Skuddis, N. Zeller, D. Cremers, and N. Haala, "Hi-slam2: Geometry-aware gaussian slam for fast monocular scene reconstruction," *arXiv preprint arXiv:2411.17982*, 2024.

[9] E. Sandström, G. Zhang, K. Tateno, M. Oechsle, M. Niemeyer, Y. Zhang, M. Patel, L. Van Gool, M. Oswald, and F. Tombari, "Splat-slam: Globally optimized rgb-only slam with 3d gaussians," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 1680–1691.

[10] H. Wang and L. Agapito, "3d reconstruction with spatial memory," in *2025 International Conference on 3D Vision (3DV)*. IEEE, 2025, pp. 78–89.

[11] S. Wang, V. Leroy, Y. Cabon, B. Chidlovskii, and J. Revaud, "Dust3r: Geometric 3d vision made easy," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 20 697–20 709.

[12] V. Leroy, Y. Cabon, and J. Revaud, "Grounding image matching in 3d with mast3r," in *ECCV*. Springer, 2024, pp. 71–91.

[13] J. Wang, M. Chen, N. Karaev, A. Vedaldi, C. Rupprecht, and D. Novotny, "Vggt: Visual geometry grounded transformer," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 5294–5306.

[14] J. Yang, A. Sax, K. J. Liang, M. Henaff, H. Tang, A. Cao, J. Chai, F. Meier, and M. Feiszli, "Fast3r: Towards 3d reconstruction of 1000+ images in one forward pass," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 21 924–21 935.

[15] K. Deng, Z. Ti, J. Xu, J. Yang, and J. Xie, "Vggt-long: Chunk it, loop it, align it–pushing vggt's limits on kilometer-scale long rgb sequences," *arXiv preprint arXiv:2507.16443*, 2025.

[16] 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.

[17] C. Campos, R. Elvira, J. J. Gomez, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[18] 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.

[19] R. Salinas-Monteagudo, "Dbow3: An improved c++ library for bag-of-words image retrieval," https://github.com/rmsalinas/DBow3, 2017, accessed: 2024-06-28.

[20] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "Netvlad: Cnn architecture for weakly supervised place recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 5297–5307.

[21] S. Izquierdo and J. Civera, "Optimal transport aggregation for visual place recognition," in *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, 2024, pp. 17 658–17 668.

[22] Y. Liu, S. Dong, S. Wang, and et al., "Slam3r: Real-time dense scene reconstruction from monocular rgb videos," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025, pp. 16 651–16 662.

[23] Q. Wang, Y. Zhang, A. Holynski, A. A. Efros, and A. Kanazawa, "Continuous 3d perception model with persistent state," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 10 510–10 522.

[24] G. Potje, F. Cadar, A. Araujo, R. Martins, and E. R. Nascimento, "Xfeat: Accelerated features for lightweight image matching," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 2682–2691.

[25] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[26] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Ep n p: An accurate o (n) solution to the p n p problem," *International journal of computer vision*, vol. 81, no. 2, pp. 155–166, 2009.

[27] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 13, no. 04, pp. 376–380, 1991.

[28] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5828–5839.

[29] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1157–1163, 2016.

[30] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine *et al.*, "Scalability in perception for autonomous driving: Waymo open dataset," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 2446–2454.

[31] T. Schops, T. Sattler, and M. Pollefeys, "Bad slam: Bundle adjusted direct rgb-d slam," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 134–144.

[32] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun, "Tanks and temples: Benchmarking large-scale scene reconstruction," *ACM Transactions on Graphics (ToG)*, vol. 36, no. 4, pp. 1–13, 2017.

[33] J. Xiao, A. Owens, and A. Torralba, "Sun3d: A database of big spaces reconstructed using sfm and object labels," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 1625–1632.

[34] C. Wang, D. Gao, K. Xu, and et al., "Pypose: A library for robot learning with physics-based optimization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 22 024–22 034.

[35] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2012, pp. 573–580.

[36] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon, "Scene coordinate regression forests for camera relocalization in rgb-d images," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 2930–2937.

[37] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma *et al.*, "The replica dataset: A digital replica of indoor spaces," *arXiv preprint arXiv:1906.05797*, 2019.

[38] 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," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[39] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering." *ACM Trans. Graph.*, vol. 42, no. 4, pp. 139–1, 2023.

[40] A. Veicht, P.-E. Sarlin, P. Lindenberger, and M. Pollefeys, "Geocalib: Learning single-image calibration with geometric optimization," in *European Conference on Computer Vision*. Springer, 2024, pp. 1–20.

[41] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Sensor fusion IV: control paradigms and data structures*, vol. 1611. Spie, 1992, pp. 586–606.