ViSTA-SLAM: Visual SLAM with Symmetric Two-view Association

Ganlin Zhang [1,2] Shenhan Qian [1,2] Xi Wang [1,2,3] Daniel Cremers [1,2]
[1] TU Munich [2] MCML [3] ETH Zurich

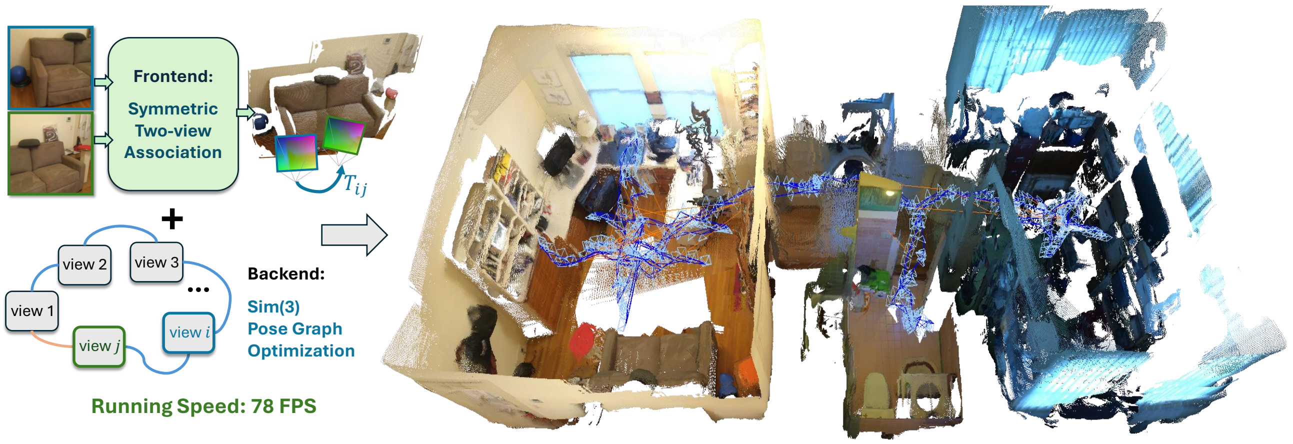


Figure 1. **ViSTA-SLAM Results on a Multi-room Scene [8].** By combining the proposed lightweight frontend Symmetric Two-view Association (STA) model with $\mathrm{Sim}(3)$ pose graph optimization and loop closuring as the backend, ViSTA-SLAM achieves high-quality reconstruction and accurate trajectory estimation on challenging scenes while running in real time.

Abstract

We present ViSTA-SLAM as a real-time monocular visual SLAM system that operates without requiring camera intrinsics, making it broadly applicable across diverse camera setups. At its core, the system employs a lightweight symmetric two-view association (STA) model as the frontend, which simultaneously estimates relative camera poses and regresses local pointmaps from only two RGB images. This design reduces model complexity significantly, the size of our frontend is only 35% that of comparable state-of-the-art methods, while enhancing the quality of two-view constraints used in the pipeline. In the backend, we construct a specially designed $\mathrm{Sim}(3)$ *pose graph that incorporates loop closures to address accumulated drift. Extensive experiments demonstrate that our approach achieves superior performance in both camera tracking and dense 3D reconstruction quality compared to current methods. Github repository:* `https://github.com/zhangganlin/vista-slam`

1. Introduction

1.1. Real-time Monocular Dense SLAM

Simultaneous Localization and Mapping (SLAM) jointly estimates an agent's pose and the surrounding 3D scene from sensor observations. In the monocular dense setting, a single RGB camera is used to reconstruct a continuous 3D map, enabling both geometric accuracy and visual realism. The camera poses and dense reconstruction outputs underpin downstream tasks [2, 19, 31, 32, 42] such as semantic perception, object interaction, and scene editing, and are crucial for applications in VR/AR, robotics, and autonomous driving, where accurate, low-latency 3D perception is essential.

1.2. Related Work

Classical Visual SLAM Classical visual SLAM methods can be broadly categorized into two types. The first, similar to incremental SfM [1, 26, 41], is feature-based SLAM [6, 10, 34, 35], relying on keypoint extraction and descriptor matching to provide constraints for triangulation and PnP [21, 22] pose estimation. The second, known as direct methods, such as LSD-SLAM [15] and DSO [16], optimizes camera poses by minimizing photometric error from pixel intensities while estimating a per-frame depth map. Both categories typically adopt a frontend (feature-based or direct) and a backend for optimization, most often bundle adjustment [49] to jointly refine poses and structure. However, they rely heavily on accurate camera calibration and are generally limited to sparse 3D reconstructions.

Dense Visual SLAM To enable denser reconstructions, recent works have incorporated deep learning into either the frontend or the scene representation. For example, DROID-SLAM [48] uses RAFT [47] for dense optical flow in the frontend and performs dense bundle adjustment on the GPU, while methods such as SuperPrimitive [30] and COMO [12] leverage monocular priors (e.g., surface normals, depth distributions) from pretrained models with direct photometric optimization. BA-Track [7] augments point-based tracking with a scale-grid deformation of monocular depth priors. Neural scene representations have also been adopted, with NICER-SLAM [61] and MonoGS [29] optimizing camera poses and 3D structure using NeRF [33] and 3D Gaussian Splatting [20], respectively. Tracking robustness and geometric accuracy have been further improved by integrating depth priors and auxiliary trackers [40, 57–60]. However, most approaches still require accurate camera intrinsics and many struggle to achieve true real-time performance due to the computational demands of dense optimization and neural rendering.

SLAM with 3D Foundation Models All aforementioned methods require known and accurate camera intrinsics. With the advent of 3D foundation models [23, 51, 53], several intrinsic-free SLAM frameworks have emerged, aiming to produce dense outputs without calibration. Methods such as Spann3R [50] and others [5, 27, 52] extend the two-view DUSt3R [53] model to sequential inputs, directly regressing point clouds in a unified global coordinate system. Reloc3r [13] instead regresses only relative poses and performs offline global optimization using SfM techniques [37, 46, 56]. MASt3R-SLAM [36] extracts dense correspondences from MASt3R [23] and feeds them into a classical optimization pipeline, while submap-based methods [11, 28] employs the multiview model VGGT [51] to regress local submaps before stitching them via pose graph optimization. While these approaches address some classical limitations, they still face notable drawbacks:

1. Current two-view models [23, 53] use asymmetric architectures that regress pointmaps of both views to the first view's coordinates, making it difficult to decouple views for backend optimization (*e.g.* loop closure).
2. Pure regression methods [27, 50, 52] predict incoming frames with previous memory, but suffer from drift and start forgetting once the trajectory gets longer.
3. Methods [27, 36, 50, 52] built on current two-view models inherit the asymmetric architecture with two separate decoders, resulting in large model size. Submap-based methods [11, 28] employ an even larger multiview model [51] to build submaps, which further increases the size of the frontend model.

1.3. Contributions of ViSTA-SLAM

To address these concerns, we propose ViSTA-SLAM, a novel real-time monocular visual SLAM pipeline based on symmetric two-view association. At its core is a lightweight Symmetric Two-view Association (STA) model frontend, which takes two RGB images as input and simultaneously regresses two pointmaps in their respective local coordinate frames, along with the relative camera pose between them. During training, we enforce cycle consistency on relative poses and geometric consistency on pointmaps to improve accuracy and stability. Unlike prior 3D models [23, 51, 53], STA is *fully symmetric* with respect to its inputs: neither view is designated as a reference, and the same encoder–decoder architecture is applied to both. In the backend, we perform $\mathrm{Sim}(3)$ pose graph optimization with loop closures to mitigate drift and ensure global consistency. To further enhance robustness, each view is represented by multiple nodes rather than a single one, which are connected by scale-only edges to handle scale inconsistencies across different forward passes.

This symmetric design makes our frontend substantially more lightweight than existing methods, with STA being only 64% the size of MASt3R [23] and 35% the size of VGGT [51]. Unlike prior approaches [11, 28] that group multiple views into a single submap node, our method assigns each view its own nodes in the pose graph. Leveraging the local pointmaps produced by the STA frontend, each node can be represented independently, connected to others solely through relative transformations. Compared to submap-based methods [28], this design yields a more flexible graph structure and greater robustness. The combination of this flexibility and lightweight architecture underpins our choice of a symmetric two-view model as the frontend.

In summary, our main **contributions** are as follows:

- We design and train a lightweight, symmetric two-view association network as the frontend, which takes only two RGB images as input and regresses their pointmaps in local coordinates along with the relative camera pose.
- We construct a robust $\mathrm{Sim}(3)$ pose graph with loop closures, optimize it using the Levenberg–Marquardt algorithm for fast and stable convergence.
- By integrating these components, we present a real-time monocular dense visual SLAM framework that operates without requiring any camera intrinsics.
- Our method achieves state-of-the-art performance on the real-world 7-Scenes [43] and TUM-RGBD [45] datasets for both camera trajectory estimation and dense 3D reconstruction.

2. ViSTA-SLAM Pipeline

As a monocular dense SLAM pipeline (Fig. 2), our aim is to simultaneously track camera poses and reconstruct the recorded scene online using a dense pointcloud. To achieve this, we propose a lightweight and novel symmetric two-view association model as the frontend of our pipeline, which extracts the relative pose and local point maps of two neighboring input frames (Sec. 2.1 and Sec. 2.2). In

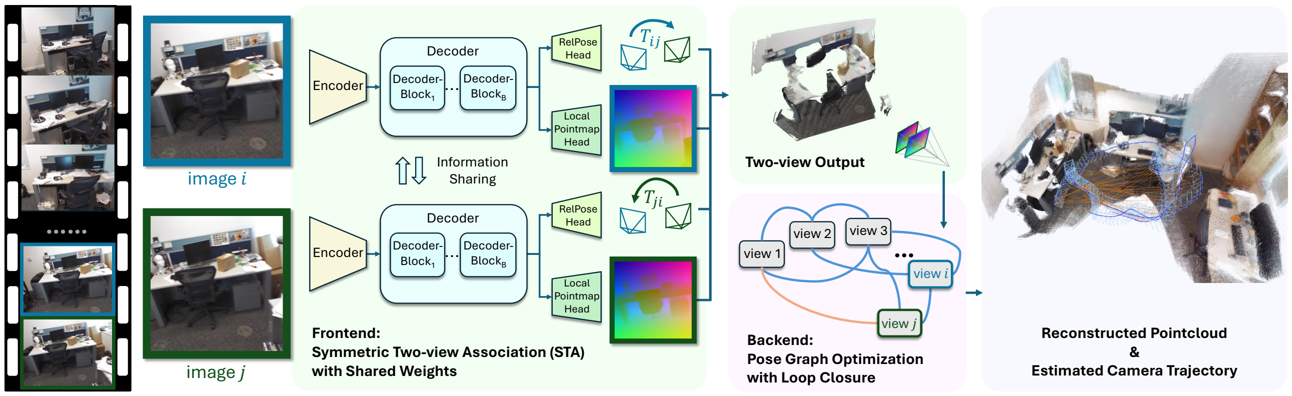


Figure 2. **ViSTA-SLAM Overview.** Given sequential video frames without intrinsics as the input, our frontend model takes in view pairs and predicts local pointmaps and relative poses within each pair. We then use the pair-wise predictions to construct a $\mathrm{Sim}(3)$ pose graph with loop closure and optimize it via Levenberg–Marquardt algorithm. The frontend model employs a fully symmetric design, making the model lightweight and supporting more flexible pose graph optimization. The blue edges in the pose graph and final results represent connections between neighboring nodes (views), while the orange edges correspond to loop closures.

the backend, a sparse and efficient $\mathrm{Sim}(3)$ pose graph optimization with loop closure is performed to mitigate drift accumulation (Sec. 2.3).

2.1. Symmetric Two-view Association Model

In classical monocular SLAM pipelines, the two-view estimation is one of the most critical building block, as it establishes geometric constraints that allow for further optimization. In this work, we follow the same principle; however, instead of relying on traditional methods, we propose a deep learning based Symmetric Two-view Association (STA) model that eliminates the need for camera intrinsics in the SLAM process.

Encoder Our STA model takes in two images $\boldsymbol{I}_i$, $\boldsymbol{I}_j$ as input. It uses a shared ViT encoder [14] to divide input images into patches and encode them into features

$$\boldsymbol{E}_{i/j} = \mathrm{Encoder}\left(\boldsymbol{I}_{i/j}\right) \quad \in \mathbb{R}^{K\times C},$$

where K represents tokens and C denotes the dimensions. Subsequently, we insert a camera pose embedding $\boldsymbol{p}$ to the encoding features of each view, forming

$$\boldsymbol{E}'_{i/j} = \left(\boldsymbol{p}, \boldsymbol{E}_{i/j}\right) \quad \in \mathbb{R}^{(K+1)\times C}.$$

Decoder The decoder further processes and fuses information between the encoding features $\boldsymbol{E}'_i$ and $\boldsymbol{E}'_j$. It contains a sequence of B decoder blocks, each conducting a self-attention operation followed by a cross-attention operation, producing decoding features

$$\boldsymbol{D}^{(b)}_{i/j} = \mathrm{DecoderBlock}^{(b)}\left(\boldsymbol{D}^{(b-1)}_{i/j}, \boldsymbol{D}^{(b-1)}_{j/i}\right) \in \mathbb{R}^{(K+1)\times C'},$$

where $b \in \{1, 2, \ldots, B\}$ is the index of a decoder block, and $\boldsymbol{D}^{(0)}_{i/j} = \boldsymbol{E}'_{i/j}$.

Symmetric Formulation Prior approaches [23, 53] regress both point maps into the coordinate frame of the first view, thus requiring two separate decoders. In contrast, our model predicts only local point maps and relative poses between views. This fully symmetric formulation makes it possible to use only one decoder. As a result, the number of parameters for decoding is effectively reduced by half (shown in Fig. 3), forming a more compact model for real-time applications. Moreover, producing local view outputs in their own coordinate systems is better suited for the subsequent pose graph optimization, see Sec. 2.3 for details.

Local Point Maps Given decoding features $\boldsymbol{D}^{(b)}_{i/j}$, we use a DPT head [38] to regress the local point maps $\boldsymbol{P}$ and corresponding confidence maps $\boldsymbol{W}$:

$$\boldsymbol{P}_{i/j}, \boldsymbol{W}_{i/j} = \mathrm{PointHead}\left(\boldsymbol{D}^{(b)}_{i/j}\right)$$

Relative Poses Given the first embedding of the decoding feature $\boldsymbol{D}^{(B)}_i$, i.e., the camera pose embedding $\boldsymbol{p}^{(B)}_i \in \mathbb{R}^{1\times C'}$, we use an MLP to regress the relative transformation from view i to j . Specifically, the MLP outputs a matrix $\boldsymbol{M}_{ij} \in \mathbb{R}^{3\times 3}$ for rotation, a translation vector $\boldsymbol{t}_{ij} \in \mathbb{R}^{3\times 1}$, and a confidence score $w_{ij} \in [0, 1]$:

$$\boldsymbol{M}_{ij}, \boldsymbol{t}_{ij}, w_{ij} = \mathrm{PoseHead}(\boldsymbol{p}^{(B)}_i)$$

Since $\boldsymbol{M}_{ij}$ is not guaranteed to lie on the $SO(3)$ manifold, we apply SVD orthogonalization [24] to it to obtain a valid rotation matrix $\boldsymbol{R}_{ij}$. Then, the relative transformation is $\boldsymbol{T}_{ij} = [\boldsymbol{R}_{ij}|\boldsymbol{t}_{ij}]$. With our symmetric formulation, we could also input $\boldsymbol{p}_j$ into the pose head to regress $\boldsymbol{T}_{ji}$. But in practice we only regress one of them for building pose graph.

2.2. Training Objective

There are three loss terms to supervise the training of our STA model. *Pointmap Loss* compares the regressed local pointmap with ground-truth points. *Relative Pose Loss* penalizes errors in relative rotation and translation, with a cycle-consistency term ensuring the two predicted poses are mutual inverses. *Geometric Consistency Loss* enforces alignment of the two local point maps after applying the predicted relative transformation, improving local reconstruction consistency.

Local Pointmap Loss Following DUSt3R [53], we apply the confidence-weighted regression loss for all predicted point maps with valid ground truths. Since the reconstruction is up-to-scale, we also normalize the regressed pointmap and the ground-truth pointmap according to their mean Euclidean distance to the origin, n and $\hat{n}$:

$$L_{\text{pmap}} = \sum_{v\in\{i,j\}} \sum_{\boldsymbol{x}\in\boldsymbol{I}_v} \left\| \boldsymbol{W}_v(\boldsymbol{x}) \cdot \left(\frac{\hat{\boldsymbol{P}}_v(\boldsymbol{x})}{\hat{n}} - \frac{\boldsymbol{P}_v(\boldsymbol{x})}{n} \right) \right\| - \alpha^{\text{point}} \log\left(\boldsymbol{W}_v(\boldsymbol{x})\right), \tag{1}$$

where x is the pixel coordinate. Note that all the points are regressed in the local coordinate space of each view.

Relative Pose Loss Relative pose loss consists of three parts: *rotation loss*, *translation loss* and *identity loss*. The rotation loss L_R evaluates the angle between the regressed rotation $\boldsymbol{R}_{ij}$ and the ground-truth rotation $\hat{\boldsymbol{R}}_{ij}$,

$$L_R(\boldsymbol{R}, \hat{\boldsymbol{R}}) = \arccos\left(\frac{\text{tr}(\boldsymbol{R}^{-1}\hat{\boldsymbol{R}}) - 1}{2} \right). \tag{2}$$

The translation loss L_t evaluates the euclidean distance between the predicted translation $\boldsymbol{t}_{ij}$ and the ground truth $\hat{\boldsymbol{t}}_{ij}$, which are normalized by the same factors n and $\hat{n}$ as for the pointmap loss in Eq. (1):

$$L_t(\boldsymbol{t}, \hat{\boldsymbol{t}}) = \left\| \frac{\boldsymbol{t}}{n} - \frac{\hat{\boldsymbol{t}}}{\hat{n}} \right\|^2. \tag{3}$$

The pose identity loss L_{id} minimizes the difference of $\boldsymbol{T}_{ij}\boldsymbol{T}_{ji}$ and the identity transformation $\boldsymbol{I}$, essentially constraining $\boldsymbol{T}_{ij}$ and $\boldsymbol{T}_{ji}$ to be the inverse of each other to improves the consistency of our pose prediction:

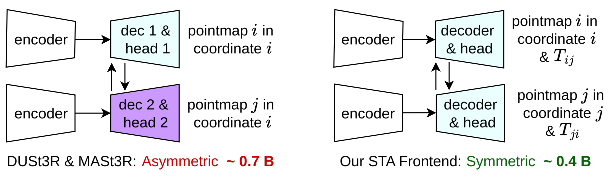
$$L_{id} = L_R(\boldsymbol{R}_{ij}\boldsymbol{R}_{ji}, \boldsymbol{I}) + L_t(\boldsymbol{R}_{ij}\boldsymbol{t}_{ji} + \boldsymbol{t}_{ij}, \boldsymbol{0}). \tag{4}$$

Then the complete relative pose loss is defined as

$$L_{\text{pose}} = w_{ij} \left(L_R(\boldsymbol{R}_{ij}, \hat{\boldsymbol{R}}_{ij}) + L_t(\boldsymbol{t}_{ij}, \hat{\boldsymbol{t}}_{ij}) + L_{id} \right) - \alpha \log(w_{ij}), \tag{5}$$

weighted by a separate confidence score w_{ij} for pose regression.

Geometrical Consistency Loss To ensure that the predicted point maps of each pair are spatially consistent when placed in the same coordinate space, we introduce a *geometric consistency loss*. Given a pair of images with ground-truth intrinsics, depths, and relative poses, each pixel $\boldsymbol{x}$ in



encoder
dec 1 & head 1
pointmap i in coordinate i
encoder
dec 2 & head 2
pointmap j in coordinate i
DUSt3R & MASt3R: Asymmetric ~ 0.7 B
encoder
decoder & head
pointmap i in coordinate i & T_{ij}
encoder
decoder & head
pointmap j in coordinate j & T_{ji}
Our STA Frontend: Symmetric ~ 0.4 B

Figure 3. **Asymmetric vs. Symmetric Architectures.** Asymmetric architectures [23, 53] use two decoders to regress point maps in a shared coordinate space. our symmetric formulation regresses relative pose and local point maps with only a single decoder, reducing over 36% of the parameters ($\sim$ 0.4 vs. 0.7 billion), while achieving higher accuracy and enabling pose graph optimization in the backend.

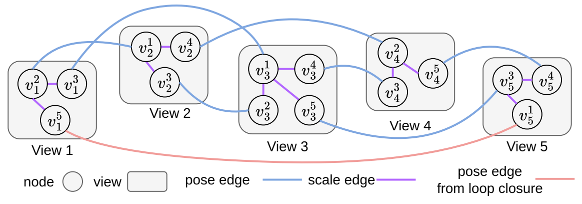


Figure 4. **An Example Pose Graph for 5 views** ($N = 2$)**.** Nodes of the same view are grouped and connected to the first processed node of that view via *scale edges*. Our two-type-edge design enhances optimization robustness, yielding more accurate poses.

view i can be accurately projected into view j, yielding its ground-truth corresponding pixel $\boldsymbol{C}_{ij}(\boldsymbol{x})$ in view j. Then the geometric consistency loss L_{gc} is defined as:

$$L_{\text{gc}} = \sum_{\boldsymbol{x}\in\boldsymbol{I}_i} \left\| \boldsymbol{T}_{ij}\boldsymbol{P}_i(\boldsymbol{x}) - \boldsymbol{P}_j\left(\boldsymbol{C}_{ij}(\boldsymbol{x})\right) \right\| / n, \tag{6}$$

where $\boldsymbol{T}_{ij}$ is the predicted relative transformation and n is the same normalization factor as in Eq. (1).

The total training objective function L is the weighted sum of the above three losses,

$$L = \lambda_{\text{pmap}} L_{\text{pmap}} + \lambda_{\text{pose}} L_{\text{pose}} + \lambda_{\text{gc}} L_{\text{gc}}, \tag{7}$$

where λ_{pmap}, λ_{pose} and λ_{gc} are weighting factors.

2.3. Backend Pose Graph Optimization

Notation In the backend, a $\text{Sim}(3)$ pose graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is maintained and optimized to mitigate accumulated errors introduced by the two-view estimations. The vertex set $\mathcal{V}$ and edge set $\mathcal{E}$ are defined as

$$\mathcal{V} = \{\boldsymbol{v}_i^j \mid i, j \in \mathbb{N}\}, \quad \mathcal{E} = \{\boldsymbol{e}_{ij} \mid i, j \in \mathbb{N}\}, \tag{8}$$

where $\boldsymbol{v}_i^j, \boldsymbol{e}_{ij} \in \text{Sim}(3)$, each vertex $\boldsymbol{v}_i^j$ represents an absolute camera pose with scale, of view i, with view i and view$j$ as the input of STA; each edge $\boldsymbol{e}_{ij}$ encodes the relative transformation between a pair of connected vertices $\boldsymbol{v}_i^j$ and $\boldsymbol{v}_j^i$. Both $\boldsymbol{v}_i^j$ and $\boldsymbol{e}_{ij}$ have a rigid transformation part $\boldsymbol{T} \in \text{SE}(3)$ and a scale part $s \in \mathbb{R}^+$.

Graph Construction To construct the pose graph, for a given view i, the STA model performs $2N$ forward passes

	Method	chess	fire	heads	office	pumpkin	kitchen	stairs	**Avg.**
Calib.	NICER-SLAM [61]	**0.033**	0.069	0.042	0.108	0.200	0.039	0.108	0.086
	DROID-SLAM [48]	0.036	**0.027**	0.025	**0.066**	**0.127**	**0.040**	0.026	**0.049**
	GlORIE-SLAM [57]	0.036	0.029	**0.014**	0.094	0.144	0.053	**0.020**	0.056
Uncalib.	CUT3R [52]	0.743	0.226	0.363	0.664	0.546	0.381	0.413	0.477
	SLAM3R [27]	0.089	0.048	0.036	0.088	0.196	0.102	0.126	0.098
	MASt3R-SLAM [36]	0.063	0.046	0.029	0.103	**0.112**	0.074	0.032	0.066
	VGGT-SLAM [28]	**0.037**	**0.026**	**0.022**	0.103	0.147	0.063	0.095	0.070
	ViSTA-SLAM	0.073	0.035	0.028	**0.055**	0.129	**0.035**	**0.029**	**0.055**

Table 1. **Camera Trajectory Estimation (ATE RMSE) on 7-Scenes [43].** ViSTA-SLAM performs the best on average.

with neighboring views $j \in [i-N, i-1] \cup [i+1, i+N]$. The predicted pointmap for each view in each forward pass corresponds to a node in the graph, resulting in multiple nodes per view since each view is processed multiple times by the frontend model. As shown in Fig. 4, we define two types of edges in the graph. *Pose edges* connect two nodes generated from the same forward pass, using the estimated relative camera pose and an identity relative scale, based on the assumption that the two local point maps from a single forward pass share the same scale. *Scale edges* connect nodes belonging to the same view but obtained in different forward passes (paired with different neighboring views), with the rigid transformation component set to identity, and the scale component solved via weighted least squares between the predicted point maps of different forward passes. Among all nodes of the same view, *scale edges* are constructed only between the first processed node and the others for sparsity and simplicity.

Loop Closure We use Bag of Words [17] to detect loop candidates, which form new pairs. Then we can feed each candidates pair into our STA model to confirm these proximity. If the predicted confidence score of the relative pose is higher than a predefined threshold τ_p, this pair is accepted as a valid loop, and two new nodes connected by a *pose edge* are added. Each new node is also connected to the first processed node of its corresponding view via a *scale edge*.

Optimization We perform pose graph optimization using the Levenberg–Marquardt algorithm in the space of Lie algebra $\mathfrak{sim}(3)$.

$$\min_{\{\boldsymbol{v}_i^j \in \mathcal{V}\}} \sum_{\boldsymbol{e}_{ij} \in \mathcal{E}} \left\| \log_{\mathrm{Sim}(3)} \left(\boldsymbol{e}_{ij} \cdot (\boldsymbol{v}_i^j)^{-1} \cdot \boldsymbol{v}_j^i \right) \right\|^2_{\boldsymbol{\Omega}_{ij}} \tag{9}$$

where $\boldsymbol{\Omega}_{ij}$ represents the covariance matrix, derived from the confidence score predicted by the STA model. The optimization process takes less than 5 iterations to converge in most cases.

Using the optimized camera pose and scale, the reconstructed pointcloud $\tilde{\boldsymbol{P}}_i^j$ in global coordinate is,

$$\tilde{\boldsymbol{P}}_i^j = s_i^j \boldsymbol{R}_i^j \boldsymbol{P}_i^j + \boldsymbol{t}_i^j, \tag{10}$$

where $\boldsymbol{R}_i^j$, $\boldsymbol{t}_i^j$ and s_i^j are the orientation, position, scale of $\boldsymbol{v}_i^j$ respectively. To avoid redundancy, for each view i, we only keep the pointcloud with largest confidence among all $\tilde{\boldsymbol{P}}_i^*$ in the final result.

3. Experiments

Evaluation Datasets and Metrics Following VGGT-SLAM [28], we evaluate our method on standard monocular SLAM benchmarks for camera tracking accuracy and reconstruction quality. We report root mean square error (RMSE) of absolute trajectory error (ATE, in meters) on real-world 7-Scenes [43] and TUM-RGBD [45] datasets using the evo toolkit [18]. Reconstruction quality on 7-Scenes is assessed via RMSE of accuracy, completion, and Chamfer distance (meters), leveraging its ground-truth 3D scenes.

Implementation Details The frontend STA model is initialized from the weights of DUSt3R [53], and trained on ScanNet [8], ScanNet++[54], ARKitScenes[4], CO3D [39], Aria Synthetic Environments [3], and Replica [44] for 7 days using 8 NVIDIA H100 GPUs. We use AdamW optimizer to train our STA model with learning rate $1.5e^{-5}$, weight decay 0.01, $B = 12$, $\alpha^{\text{point}} = 0.2$, $\alpha^{\text{pose}} = 0.05$, $\lambda_{\text{pmap}} = 1$, $\lambda_{\text{pose}} = 1$, $\lambda_{\text{gc}} = 1$, and $\tau_p = 0.75$. We conducted evaluations on a machine with an NVIDIA RTX 4090 GPU and an Intel i9-14900KF CPU, with $N = 2$ for 7-Scenes and $N = 3$ for TUM-RGBD.

Baselines ViSTA-SLAM is primarily compared with state-of-the-art (SOTA) learning-based SLAM methods in uncalibrated scenarios: VGGT-SLAM [28], MASt3R-SLAM [36], SLAM3R [27], and CUT3R [52]. To reduce randomness from VGGT-SLAM's RANSAC, we run it 5 times per scene as suggested in their paper; results for other methods, including ours, are deterministic. We also compare with SOTA methods using known camera intrinsics [6, 25, 48, 57, 61]. Some results are taken from [28, 36]. For MASt3R-SLAM and VGGT-SLAM, we keep their original keyframe selection; for ours, CUT3R, and SLAM3R, we use frame strides of 5 (7-Scenes) and 3 (TUM-RGBD). Calibrated methods are shown in gray. Best results are highlighted as **first**, second, and third.

3.1. Camera Trajectory Evaluation

In Tab. 1 and Tab. 2, we report ATE RMSE. ViSTA-SLAM achieves the best average performance on both datasets, out-

	Method	360	desk	desk2	floor	plant	room	rpy	teddy	xyz	**Avg.**
Calibrated	ORB-SLAM3 [6]	×	0.017	0.210	×	0.034	×	×	×	**0.009**	N/A
	DPV-SLAM [25]	0.112	0.018	0.029	0.057	0.021	0.330	0.030	0.084	0.010	0.076
	DPV-SLAM++ [25]	0.132	0.018	0.029	0.050	0.022	0.096	0.032	0.098	0.010	0.054
	DROID-SLAM [48]	**0.111**	0.018	0.042	**0.021**	**0.016**	0.049	0.026	0.048	0.012	0.038
	GlORIE-SLAM [57]	0.128	**0.016**	**0.028**	**0.021**	0.021	**0.042**	**0.020**	**0.035**	0.010	**0.036**
Uncalibrated	CUT3R [52]	0.174	0.592	0.546	0.662	0.467	0.911	0.051	0.845	0.129	0.486
	SLAM3R [27]	0.211	0.861	0.967	0.790	0.755	1.013	0.063	0.986	0.185	0.648
	MASt3R-SLAM [36]	0.070	0.032	0.055	**0.056**	0.035	0.118	0.041	0.116	0.020	0.060
	VGGT-SLAM [28]	**0.063**	0.031	0.048	0.152	**0.023**	0.133	0.038	**0.039**	0.020	0.061
	ViSTA-SLAM	0.104	**0.030**	**0.030**	0.070	0.052	**0.067**	**0.023**	0.080	**0.015**	**0.052**

Table 2. **Camera Trajectory Estimation (ATE RMSE) on TUM-RGBD [45].** ViSTA-SLAM performs the best on average.

performing current SOTA [36] by 17% (0.055 vs. 0.066) and 13% (0.052 vs. 0.060), and surpasses some calibrated methods [25, 61]. ViSTA-SLAM performs less effectively on TUM-RGBD 360 scene due to predominantly rotational camera motion that leads to frontend ambiguity and degrading performance. Other methods [28, 36] use either heavier multi-view frontend or more intensive optimization to reduce the influence. Pure regression-based methods [27, 52] struggle to maintain consistent registration over long sequences with large camera motion due to forgetting effects.

In Fig. 5, we show the estimated trajectories from different methods on 7-Scenes [43] office and TUM-RGBD [45] room. CUT3R [52] suffers from severe forgetting issues on long sequences; SLAM3R [27] has bad point registration on the challenged scene TUM-RGBD room, thus, does not produce correct camera poses. Compared to pure regression-based methods, MASt3R-SLAM [36] and VGGT-SLAM [28] work well, while ViSTA-SLAM achieves even higher trajectory accuracy.

3.2. Dense Reconstruction Evaluation

In Tab. 3, we evaluate reconstruction quality across methods. Leveraging accurate camera poses and consistent local point clouds, ViSTA-SLAM achieves the best Chamfer distance among all approaches. Despite using a lightweight two-view frontend, ViSTA-SLAM, combined with tailored $\mathrm{Sim}(3)$ pose graph optimization, significantly outperforms multi-view-frontend methods [27, 28] in accuracy (0.45 vs. 0.52) while matching or exceeding completeness. To demonstrate the effectiveness of our lightweight frontend, we add another strong baseline, replacing our STA model with a two-view VGGT [51] as the frontend and conducting the same pose graph optimization. ViSTA-SLAM still achieves better performance in Chamfer distance, completeness, and absolute trajectory error, highlighting the effectiveness of our lightweight symmetric frontend over larger multiview models like VGGT for SLAM tasks

In Fig. 6, we show qualitative reconstruction results on 7-Scenes redkitchen, TUM-RGBD [45] room, and BundleFusion [9] apt1. CUT3R [52] fails to reconstruct

Method	ATE ↓	Acc. ↓	Comp. ↓	Chamfer ↓
DROID-SLAM [48]	0.049	0.141	**0.048**	0.094
MASt3R-SLAM [36]	**0.047**	**0.089**	0.085	**0.087**
Spann3R @20 [50]	N/A	0.069	0.047	0.058
Spann3R @2 [50]	N/A	0.124	**0.043**	0.084
CUT3R [52]	0.477	0.276	0.303	0.290
SLAM3R [27]	0.098	0.053	0.059	0.056
MASt3R-SLAM [36]	0.066	0.059	0.056	0.057
VGGT-SLAM [28]	0.070	0.052	0.060	0.056
2-view VGGT *w/* PGO	0.065	**0.039**	0.077	0.058
ViSTA-SLAM	**0.055**	0.045	0.056	**0.051**

Table 3. **Tracking and Reconstruction Evaluation on 7-Scenes [43].** $@n$ indicates a keyframe every n images. *2-view VGGT w/ PGO* uses the 2-view VGGT frontend with the same pose graph optimization as ours. ViSTA-SLAM achieves the best trajectory estimation and reconstruction performance on 7-Scenes.

	CUT3R [52]	SLAM3R [27]	MASt3R SLAM [36]	VGGT SLAM [28]	2-view VGGT *w/* PGO	ViSTA SLAM
Size ↓	0.79	0.76	0.69	1.26	1.26	**0.44**
FPS ↑	34.2	45.8	30.3	**93.3**	12.6	78.0

Table 4. **Model Size and Running Time Evaluation on 7-Scenes [43] redkitchen.** Model sizes are reported in billions of parameters. FPS indicates the average number of frames processed per second over three runs. ViSTA-SLAM shows highly competitive real-time speed with the smallest model size among baselines.

correctly due to forgetting issues, while SLAM3R [27] struggles in scenes with large camera perspective changes. MASt3R-SLAM [36] and VGGT-SLAM [28] produce artifacts on object boundaries, failing to clearly separate foreground from background, and show misalignment across views. In contrast, ViSTA-SLAM overcomes these challenges through geometric consistency constraints during training. Notably, VGGT-SLAM fails midway through the apt1 scene as backend optimization diverges, which stems from the unstable RANSAC-based 3D homography estimation, which can sample planar regions and cause ambiguity in their proposed $\mathrm{SL}(4)$ pose graph optimization.

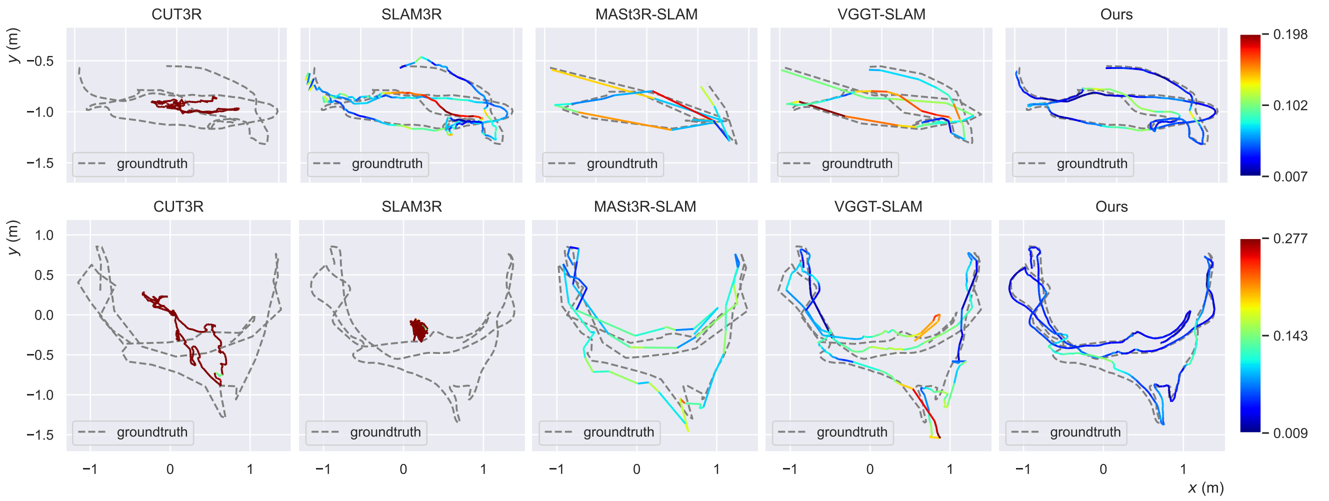


Figure 5. **Trajectory estimation results on 7-Scenes `office` (top) and TUM-RGBD `room` (bottom).** Estimated camera trajectories are projected onto the x–y plane, with ground-truth shown as dashed lines. The trajectory color encodes ATE RMSE: higher errors in red, lower in blue. For MASt3R-SLAM [36] and VGGT-SLAM [28], only the poses of their selected keyframes are estimated.

	load data	encoder	decoder	detect loop	construct graph	optimize graph
time (s)	1.88	0.89	3.38	0.41	2.93	3.31
%	14.7%	7.0%	26.4%	3.18%	22.9%	25.9%

Table 5. **Time Spent on Each Component of ViSTA-SLAM for 7-Scenes [43] `redkitchen` (in seconds and percentage).**

Settings		ATE ↓	Chamfer ↓
STA Model	*w/o* L_{gc}	0.056	0.057
	w/o L_{id}	0.058	0.059
Pose Graph	*w/o* pose graph opt.	0.105	0.070
	w/o loop closure	0.103	0.072
	w/o two edge types	0.057	**0.051**
ViSTA-SLAM with full features		**0.055**	**0.051**

Table 6. **Ablation Study on 7-Scenes [43].** *w/o pose graph opt.* simply accumulates relative poses for absolute poses. *w/o two edge types* uses the classical pose graph in which each view is represented by a single node.

3.3. Model Size and Speed

We compare the frontend model size and processing speed across methods in Tab. 4. Owing to our symmetric design, the decoder and regression heads use only half the parameters of existing feedforward models [23, 27, 50, 53]. Consequently, our model is far more compact: only 64% the size of MASt3R [23] (used in MASt3R-SLAM [36]) and 35% the size of VGGT [51] (used in VGGT-SLAM [28]).

The speed evaluation further confirms that ViSTA-SLAM achieves real-time performance. Benefiting from both the compact frontend and the sparse pose graph, our approach is highly competitive in runtime—faster than the pure regression-based methods CUT3R [52] and SLAM3R [27], and comparable to VGGT-SLAM [28]. It is worth noting that VGGT-SLAM performs inference only once every 32 keyframes, reducing the total number of inference steps. When replacing our STA model with a two-view VGGT that takes two views as input at a time like STA, the running speed is significantly slower, further demonstrating the effectiveness of our lightweight frontend.

Tab. 5 shows the percentage of runtime spent on major pipeline components. Decoding two-view information and pose graph optimization dominate the processing time.

3.4. Ablation Study

In Tab. 6, we present ablations by selectively disabling components of ViSTA-SLAM. Incorporating all proposed features yields the best performance on both camera trajectory estimation and 3D reconstruction.

Both L_{gc} and L_{id} contribute substantially to reconstruction quality. L_{gc} improves consistency between the reconstructed local pointmaps of the two-view input pair, while L_{id} further refines the estimated relative camera poses by enforcing a cycle consistency constraint on the model.

Pose graph optimization with loop closure is also highly effective, bringing 48% improvement in the trajectory accuracy (0.105 → 0.055), as it introduces simple yet powerful constraints through the edges of the pose graph, preventing error accumulation from two-view estimations as the trajectory grows longer. These findings further support the symmetric formulation of our frontend to regress local pointclouds and relative poses, which maximizes the effectiveness of pose graph optimization since the pointmaps of each view are tightly coupled with their corresponding camera poses in the graph. On the contrary, if pointmaps were regressed in a shared coordinate system inside a submap, as in previous works [5, 23, 27, 50, 52, 53], pose updates would not be able to fix misalignments inside submaps.

Our two-edge-type pose graph design improves camera

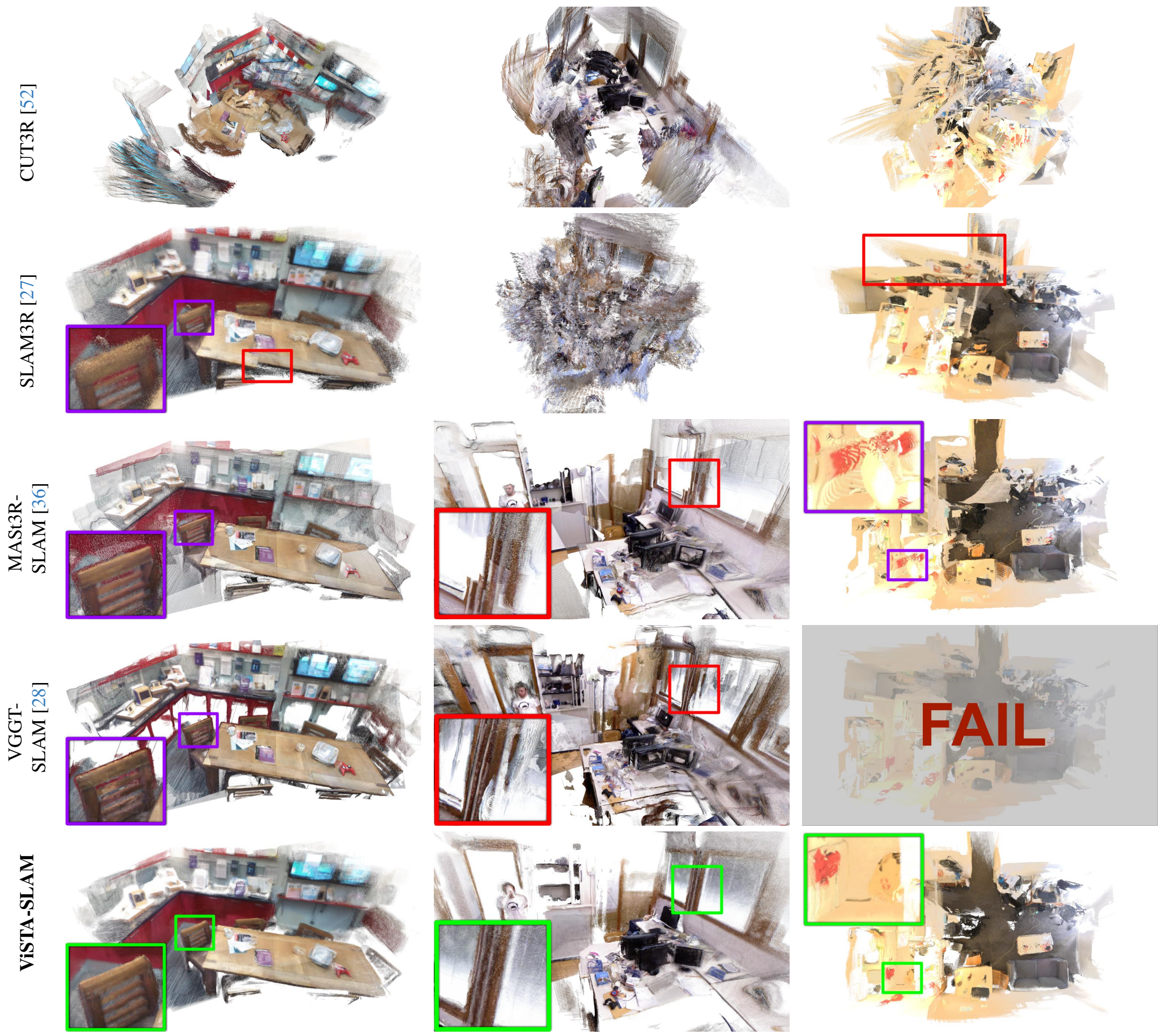


Figure 6. **Reconstruction results on 7-Scenes `redkitchen` (left), TUM-RGBD `room` (middle), and BundleFusion `apt1` (right).** Purple boxes highlight reconstruction artifacts near the edges (background points wrongly mapped to the edge of the foreground). Red boxes indicate misalignments. Green boxes highlights ViSTA-SLAM's competitive results. VGGT-SLAM fails to complete reconstruction on `apt1` due to divergence in pose graph optimization.

trajectory estimation by representing each view with multiple nodes connected by scale and pose edges, rather than a single node with standard edges. This structure better averages out uncertainty, particularly relative scale variations in pointmaps from different forward passes, improving the robustness of pose graph optimization and the accuracy of trajectory estimation.

4. Conclusion

We propose a novel monocular intrinsics-free SLAM pipeline, ViSTA-SLAM, which features a lightweight frontend (Symmetric Two-View Association) and a $\mathrm{Sim}(3)$ pose graph optimization with loop closure as the backend. Experimental results demonstrate the superior camera tracking accuracy and 3D reconstruction quality of ViSTA-SLAM. Meanwhile, it is significantly more lightweight and operates at a faster or comparable speed comparing current state-of-the-art methods.

Limitation and Future Work Our method omits optimization of point clouds in the backend for efficiency consideration. Therefore, it can suffer from misalignments caused by imperfect pointmap prediction by the frontend model. Future work could explore incorporating implicit camera information from previous estimates or aligning latent features across views to enhance local consistency across forward passes.

Acknowledgment This research was funded by the German Federal Ministry of Education and Research through the ExperTeam4KI funding program for UDance (Grant No. 16IS24064).

References

[1] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M Seitz, and Richard Szeliski. Building rome in a day. *Communications of the ACM*, 54 (10):105–112, 2011. 1

[2] Iro Armeni, Zhi-Yang He, JunYoung Gwak, Amir R. Zamir, Martin Fischer, Jitendra Malik, and Silvio Savarese. 3D scene graph: A structure for unified semantics, 3D space, and camera. In *ICCV*, 2019. 1

[3] Armen Avetisyan, Christopher Xie, Henry Howard-Jenkins, Tsun-Yi Yang, Samir Aroudj, Suvam Patra, Fuyang Zhang, Duncan Frost, Luke Holland, Campbell Orme, et al. Scenescript: Reconstructing scenes with an autoregressive structured language model. In *ECCV*, pages 247–263. Springer, 2024. 5

[4] Gilad Baruch, Zhuoyuan Chen, Afshin Dehghan, Tal Dimry, Yuri Feigin, Peter Fu, Thomas Gebauer, Brandon Joffe, Daniel Kurz, Arik Schwartz, and Elad Shulman. ARKitscenes - a diverse real-world dataset for 3D indoor scene understanding using mobile RGB-D data. In *NeurIPS*, 2021. 5

[5] Yohann Cabon, Lucas Stoffl, Leonid Antsfeld, Gabriela Csurka, Boris Chidlovskii, Jerome Revaud, and Vincent Leroy. MUSt3R: Multi-view network for stereo 3D reconstruction. In *CVPR*, pages 1050–1060, 2025. 2, 7

[6] 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 multimap SLAM. *IEEE transactions on robotics*, 37(6): 1874–1890, 2021. 1, 5, 6

[7] Weirong Chen, Ganlin Zhang, Felix Wimbauer, Rui Wang, Nikita Araslanov, Andrea Vedaldi, and Daniel Cremers. Back on track: Bundle adjustment for dynamic scene reconstruction. *arXiv preprint arXiv:2504.14516*, 2025. 2

[8] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In *CVPR*, 2017. 1, 5, 2

[9] Angela Dai, Matthias Nießner, Michael Zollöfer, Shahram Izadi, and Christian Theobalt. BundleFusion: Real-time globally consistent 3D reconstruction using on-the-fly surface re-integration. *ACM TOG*, 2017. 6, 4

[10] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE TPAMI*, 29(6):1052–1067, 2007. 1

[11] Kai Deng, Zexin Ti, Jiawei Xu, Jian Yang, and Jin 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. 2

[12] Eric Dexheimer and Andrew J Davison. COMO: Compact mapping and odometry. In *ECCV*, pages 349–365. Springer, 2024. 2

[13] Siyan Dong, Shuzhe Wang, Shaohui Liu, Lulu Cai, Qingnan Fan, Juho Kannala, and Yanchao Yang. Reloc3r: Large-scale training of relative camera pose regression for generalizable, fast, and accurate visual localization. In *CVPR*, pages 16739–16752, 2025. 2

[14] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 3

[15] Jakob Engel, Thomas Schöps, and Daniel Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *ECCV*, pages 834–849. Springer, 2014. 1

[16] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE TPAMI*, 40(3):611–625, 2017. 1

[17] 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

[18] Michael Grupp. evo: Python package for the evaluation of odometry and SLAM. `https://github.com/MichaelGrupp/evo`, 2017. 5

[19] Anna-Maria Halacheva, Yang Miao, Jan-Nico Zaech, Xi Wang, Luc Van Gool, and Danda Pani Paudel. Holistic understanding of 3D scenes as universal scene description. *arXiv preprint arXiv:2412.01398*, 2024. 1

[20] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023. 2

[21] Laurent Kneip, Davide Scaramuzza, and Roland Siegwart. A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. In *CVPR*, pages 2969–2976. IEEE, 2011. 1

[22] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. EPnP: An accurate O(n) solution to the PnP problem. *IJCV*, 81(2):155–166, 2009. 1

[23] Vincent Leroy, Yohann Cabon, and Jerome Revaud. Grounding image matching in 3D with MASt3R, 2024. 2, 3, 4, 7

[24] Jake Levinson, Carlos Esteves, Kefan Chen, Noah Snavely, Angjoo Kanazawa, Afshin Rostamizadeh, and Ameesh Makadia. An analysis of SVD for deep rotation estimation. *NeurIPS*, 33:22554–22565, 2020. 3

[25] Lahav Lipson, Zachary Teed, and Jia Deng. Deep patch visual SLAM. In *ECCV*, pages 424–440. Springer, 2024. 5, 6

[26] Shaohui Liu, Yidan Gao, Tianyi Zhang, Rémi Pautrat, Johannes L Schönberger, Viktor Larsson, and Marc Pollefeys. Robust incremental structure-from-motion with hybrid features. In *ECCV*, pages 249–269. Springer, 2024. 1

[27] Yuzheng Liu, Siyan Dong, Shuzhe Wang, Yingda Yin, Yanchao Yang, Qingnan Fan, and Baoquan Chen. SLAM3R: Real-time dense scene reconstruction from monocular RGB videos. In *CVPR*, pages 16651–16662, 2025. 2, 5, 6, 7, 8, 1

[28] Dominic Maggio, Hyungtae Lim, and Luca Carlone. VGGT-SLAM: Dense RGB SLAM optimized on the SL(4) manifold. *arXiv preprint arXiv:2505.12549*, 2025. 2, 5, 6, 7, 8

[29] Hidenobu Matsuki, Riku Murai, Paul HJ Kelly, and Andrew J Davison. Gaussian splatting SLAM. In *CVPR*, pages 18039–18048, 2024. 2

[30] Kirill Mazur, Gwangbin Bae, and Andrew J Davison. SuperPrimitive: Scene reconstruction at a primitive level. In *CVPR*, pages 4979–4989, 2024. 2

[31] Yang Miao, Iro Armeni, Marc Pollefeys, and Daniel Barath. Volumetric semantically consistent 3d panoptic mapping. In *IROS*, pages 12924–12931. IEEE, 2024. 1

[32] Yang Miao, Francis Engelmann, Olga Vysotska, Federico Tombari, Marc Pollefeys, and Dániel Béla Baráth. Scenegraphloc: Cross-modal coarse visual localization on 3D scene graphs. In *ECCV*, pages 127–150. Springer, 2024. 1

[33] 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. *Communications of the ACM*, 65(1):99–106, 2021. 2

[34] 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. 1

[35] 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

[36] Riku Murai, Eric Dexheimer, and Andrew J Davison. MASt3R-SLAM: Real-time dense SLAM with 3D reconstruction priors. In *CVPR*, pages 16695–16705, 2025. 2, 5, 6, 7, 8

[37] Linfei Pan, Dániel Baráth, Marc Pollefeys, and Johannes L Schönberger. Global structure-from-motion revisited. In *ECCV*, pages 58–77. Springer, 2024. 2

[38] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *ICCV*, pages 12179–12188, 2021. 3

[39] Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Sbordone, Patrick Labatut, and David Novotny. Common objects in 3D: Large-scale learning and evaluation of real-life 3D category reconstruction. In *ICCV*, 2021. 5

[40] Erik Sandström, Ganlin Zhang, Keisuke Tateno, Michael Oechsle, Michael Niemeyer, Youmin Zhang, Manthan Patel, Luc Van Gool, Martin Oswald, and Federico Tombari. Splat-SLAM: Globally optimized rgb-only SLAM with 3D gaussians. In *CVPRW*, pages 1680–1691, 2025. 2

[41] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, pages 4104–4113, 2016. 1

[42] Sunando Sengupta, Eric Greveson, Ali Shahrokni, and Philip HS Torr. Urban 3D semantic modelling using stereo vision. In *ICRA*, pages 580–585. IEEE, 2013. 1

[43] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. Scene coordinate regression forests for camera relocalization in RGB-D images. In *CVPR*, pages 2930–2937, 2013. 2, 5, 6, 7, 1, 4

[44] 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, 2

[45] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *IROS*, 2012. 2, 5, 6, 4

[46] Christopher Sweeney, Tobias Hollerer, and Matthew Turk. Theia: A fast and scalable structure-from-motion library. In *ACM MM*, pages 693–696, 2015. 2

[47] Zachary Teed and Jia Deng. RAFT: Recurrent all-pairs field transforms for optical flow. In *ECCV*, pages 402–419. Springer, 2020. 2

[48] Zachary Teed and Jia Deng. DROID-SLAM: Deep visual SLAM for monocular, stereo, and RGB-D cameras. *NeurIPS*, 34:16558–16569, 2021. 2, 5, 6

[49] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustment—a modern synthesis. In *International workshop on vision algorithms*, pages 298–372. Springer, 1999. 1

[50] Hengyi Wang and Lourdes Agapito. 3D reconstruction with spatial memory. In *3DV*, 2025. 2, 6, 7

[51] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. VGGT: Visual geometry grounded transformer. In *CVPR*, 2025. 2, 6, 7

[52] Qianqian Wang, Yifei Zhang, Aleksander Holynski, Alexei A Efros, and Angjoo Kanazawa. Continuous 3D perception model with persistent state. In *CVPR*, pages 10510–10522, 2025. 2, 5, 6, 7, 8

[53] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. DUSt3R: Geometric 3D vision made easy. In *CVPR*, pages 20697–20709, 2024. 2, 3, 4, 5, 7

[54] Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. ScanNet++: A high-fidelity dataset of 3D indoor scenes. In *ICCV*, 2023. 5

[55] Jie Yin, Ang Li, Tao Li, Wenxian Yu, and Danping Zou. M2DGR: A multi-sensor and multi-scenario SLAM dataset for ground robots. *IEEE Robotics and Automation Letters*, 7 (2):2266–2273, 2021. 4

[56] Ganlin Zhang, Viktor Larsson, and Daniel Barath. Revisiting rotation averaging: Uncertainties and robust losses. In *CVPR*, pages 17215–17224, 2023. 2

[57] Ganlin Zhang, Erik Sandström, Youmin Zhang, Manthan Patel, Luc Van Gool, and Martin R Oswald. GlORIE-SLAM: Globally optimized RGB-only implicit encoding point cloud SLAM. *arXiv preprint arXiv:2403.19549*, 2024. 2, 5, 6

[58] Wei Zhang, Tiecheng Sun, Sen Wang, Qing Cheng, and Norbert Haala. HI-SLAM: Monocular real-time dense mapping with hybrid implicit fields. *IEEE Robotics and Automation Letters*, 9(2):1548–1555, 2023.

[59] Wei Zhang, Qing Cheng, David Skuddis, Niclas Zeller, Daniel Cremers, and Norbert Haala. HI-SLAM2: Geometry-aware gaussian SLAM for fast monocular scene reconstruction. *arXiv preprint arXiv:2411.17982*, 2024.

[60] Youmin Zhang, Fabio Tosi, Stefano Mattoccia, and Matteo Poggi. GO-SLAM: Global optimization for consistent 3D instant reconstruction. In *ICCV*, pages 3727–3737, 2023. 2

[61] Zihan Zhu, Songyou Peng, Viktor Larsson, Zhaopeng Cui, Martin R Oswald, Andreas Geiger, and Marc Pollefeys. NICER-SLAM: Neural implicit scene encoding for RGB SLAM. In *3DV*, pages 42–52. IEEE, 2024. 2, 5, 6