SLAM-Former: Putting SLAM into One Transformer

Yijun Yuan[1], Zhuoguang Chen[1], Kenan Li[1], Weibang Wang[1], Minghui Qin[2], Zhijian Fang[1], Weicheng Zheng[2], and Hang Zhao[1,2]

[1] IIIS, Tsinghua University
[2] Shanghai Qi Zhi Institute

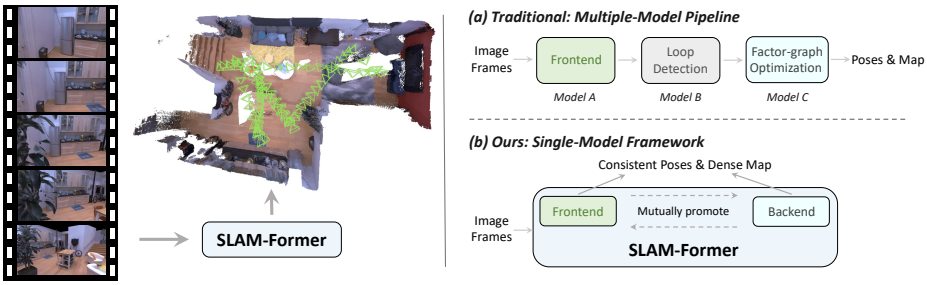


Fig. 1: SLAM-Former is a unified Transformer for SLAM. Traditional SLAM employs a multi-model pipeline for frontend and backend tasks. In contrast, SLAM-Former integrates the full SLAM functionality within one transformer, achieving consistent poses and dense maps.

Abstract. We present SLAM-Former, a neural approach that integrates full SLAM capabilities into a single transformer. Similar to traditional SLAM systems, SLAM-Former comprises both a frontend and a backend that operate in tandem. The frontend processes sequential monocular images in real-time for incremental mapping and tracking, while the backend performs global refinement to ensure a geometrically consistent result. This alternating execution allows the frontend and backend to mutually promote one another, enhancing overall system performance. Comprehensive experimental results demonstrate that SLAM-Former achieves superior or highly competitive performance compared to state-of-the-art dense SLAM methods.

Keywords: Dense SLAM · Transformer

1 Introduction

In the field of robotic perception, Simultaneous Localization and Mapping (SLAM) plays a great significance. It allows robots to construct a map of an unknown environment while simultaneously tracking their own location. This capability is essential for robots to navigate autonomously and carry out tasks in a variety of environments. Early SLAM algorithms primarily focused on localization and

mapping using sparse or semi-dense representations, such as ORB-SLAM [24] and LSD-SLAM [9]. These methods are efficient and robust, but they may not offer detailed information about the surroundings. In contrast, dense mapping techniques aim to create a more detailed and continuous representation of the environment, mainly relying on LiDAR and RGB-D [8, 26].

With the rapid advancement of optical flow and multiview depth estimation techniques, recent research has achieved high-quality dense monocular SLAM using only images as input [22, 25, 32, 43]. These approaches leverage the capability of neural networks and computer vision algorithms to estimate depth and motion from a single camera, thereby creating dense maps without additional sensors. Especially noteworthy is the trend of utilizing geometric foundation models such as DUSt3R [36] and VGGT [34]. These models reveal the high potential of data-driven 3D structure prediction. Their streaming variants, StreamVGGT [47] and Stream3R [16], by carefully leveraging the attention key-value cache (KV cache), enable the model to process incremental visual inputs.

We observe that SLAM methods using geometric foundation models as their reconstruction module, such as MASt3R-SLAM [25] and VGGT-SLAM [22] suffer from global inconsistency, since they rely on the alignment of local submaps. On the other hand, streaming methods such as StreamVGGT and Stream3R process incremental inputs without remapping the past, which can cause a significant mismatch between past and newly incoming data.

In this work, we introduce SLAM-Former, the first fully neural SLAM system with a complete SLAM pipeline, built upon a single unified transformer architecture. SLAM-Former consists of a frontend and a backend that operate cooperatively, as illustrated in Fig. 1 (b). The frontend operates in real-time on the sequential RGB images for keyframe selection and incremental map and pose updates. With the incremental output from the frontend, our backend periodically refines the map and poses globally. The frontend and backend promote each other. The frontend provides initial results and sequential order, assisting the backend in refinement. In return, the backend's KV cache is updated to the frontend periodically for further incremental operation. To empower a single transformer with all SLAM capabilities, we propose three training modes for it.

Compared to traditional SLAM pipelines that require an additional loop detection module to close the pose graph, SLAM-Former's backend accomplishes this with full attention and is equivalent to processing loop detection.

SLAM-Former achieves significantly better reconstruction and state-of-the-art tracking performance on widely used Dense Mono SLAM benchmarks, compared to both calibrated and uncalibrated state-of-the-art methods.

The main contributions of this work are:

- **Novel SLAM Architecture**. SLAM-Former is the first fully neural SLAM system featuring a complete SLAM pipeline. It introduces a novel Frontend-Backend collaboration mechanism with map tokens and KV caches.
- **Novel Function Basis**. We introduce the first unified transformer formulation for SLAM, jointly training three functional modes within a single transformer to enable complete SLAM functionality.

– **Performance**. SLAM-Former achieves outstanding reconstruction and tracking on widely used benchmarks, demonstrating the high potential of transformer-based SLAM.

2 Related Works

2.1 Dense RGB SLAM

In recent years, research on dense SLAM using monocular cameras has achieved significant progress [22, 25, 32, 43, 46], thanks to the application of deep learning techniques. Due to the absence of depth sensors, dense RGB SLAM optimizes the entire sequence of geometry and camera as a whole.

Early works focus on reducing the computational cost of depth estimation. For instance, CodeSLAM [3] and DeepFactors [6] optimize the depth latent as an alternative. Borrowing the strength of MVSNet [40], Tandem relies on an external model, but it breaks the co-optimization structure [15]. Conversely, DROID-SLAM [32] and SceneFactory [43] incorporate deep optical flow model into the pipeline and co-optimize both with a dense bundle adjustment. On the other hand, NeRF [23] and Gaussian Splatting [14] based methods have emerged as a trend to reshape Dense SLAM. NeRF-SLAM methods [28, 46] and GS-SLAM methods [38] optimize the scene as a whole for a highly realistic novel view synthesis objective. Nonetheless, those rendering-based SLAMs are time-consuming, do not provide true 3D geometry, and highly sensitive to blur and noise, which restricts their use in real life.

With the emergence of recent foundational geometry techniques, such as DUSt3R [36] and VGGT [34], researchers have found new inspiration. MASt3R-SLAM [25] utilizes the pairwise model MASt3R [17] for calibration-free matching and geometry construction, demonstrating state-of-the-art performance under a traditional SLAM pipeline. On the other hand, VGGT-SLAM [22] feeds submaps into VGGT and connects them using a novel SL(4) manifold, modeling the geometry distortion from geometric foundational model for the first time.

Existing methods typically rely on pair- or submap-wise geometry optimization, which often leads to structural inconsistencies across frames. MASt3R-SLAM improves cross-frame consistency through global optimization and local TSDF fusion [26], while VGGT-SLAM introduces sparse cross-submap connections. However, neither method explicitly models global geometric consistency across all frames.

This limitation motivates our proposed SLAM framework, which enforces such consistency during SLAM processing.

2.2 Feed-forward 3D Reconstruction

Recently, DUSt3R [36] has led a trend to regress the 3D structure with scalable training data directly. However, with processing image pairs, DUSt3R requires a global optimization for larger scenes, which lowers the inference efficiency. Several works have been proposed to address this limitation. Fast3R [39],

VGGT [34], and Pi3 [37] process multi-view images in a single forward pass, avoiding time-consuming post-processing global optimization. All three are transformer-based models for multi-view pointmap estimation. Fast3R highlights the ability to efficiently handle thousands of images, while VGGT demonstrates that a simple architecture with 3D multi-task learning and scalable training data can achieve state-of-the-art results. Pi3 further introduces a permutation-equivariant design that removes the dependence on a fixed reference view, enhancing robustness to input ordering and scalability.

Besides feed-forward multi-view methods, recent feed-forward streaming approaches tackle online 3D reconstruction. Spann3R [33] extends Dust3R to streaming via an interactive spatial memory, while CUT3R [35] introduces persistent state tokens with recurrent transformer updates. LONG3R [5] adopts a 3D spatio-temporal memory and a coarse-to-fine pipeline for long-sequence streaming. Going further, StreamVGGT [47] and STream3R [16] leverage language-model-inspired causal attention for streaming reconstruction.

However, existing streaming methods focus solely on incremental updates without revisiting past estimates, leading to drift and limited global consistency. To address this, we propose SLAM-Former, a unified neural SLAM pipeline that integrates both frontend and backend for efficient incremental updates and periodic global refinement.

3 SLAM-Former

This section introduces *SLAM-Former*. We first describe the underlying transformer architecture in Sec. 3.1, then detail its roles in the SLAM frontend (Sec. 3.2) and backend (Sec. 3.3). Next, we present a joint training strategy that unifies these tasks within a single model in Sec. 3.4, and a detailed SLAM inference pipeline in Sec. 3.5, followed by a token pruning technique in Sec. 3.6.

3.1 Transformer Architecture

SLAM-Former is built upon a single transformer model, where a **transformer backbone** f aggregates both intra-frame and inter-frame information, and task-specific **heads** h decode scene geometry and camera poses. We assume image features are pre-encoded, and the input to f contains a set of image patch tokens augmented with register tokens. The backbone contains L layers with intra-frame and inter-frame attention to jointly capture local image context and temporal correspondences.

SLAM-Former adopts a unified architecture that supports both a *frontend* for incremental frame processing and a *backend* for global map and pose refinement within the same transformer backbone (see Fig. 2).

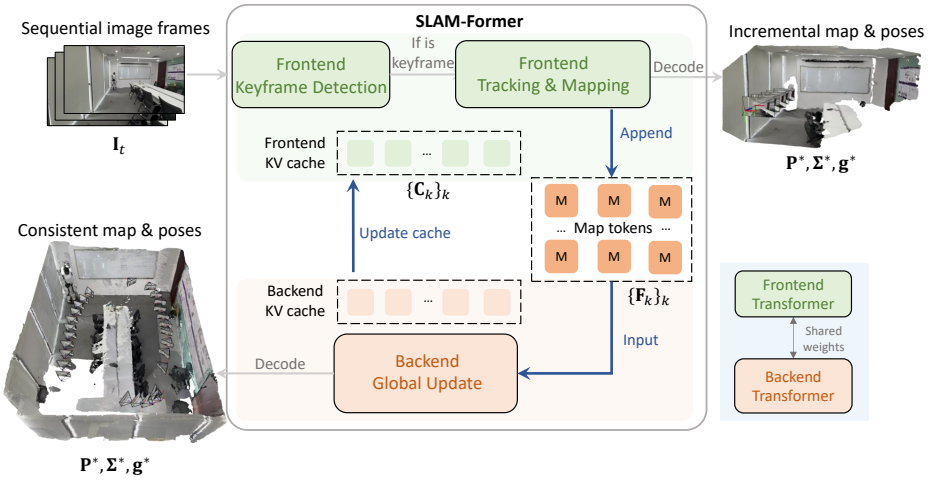


Fig. 2: The working pipeline of SLAM-Former. The frontend detects keyframes and performs incremental pose and map updates, while the backend performs global pose and map updates. The shared map token memory and the KV cache update mechanism ensure that the frontend and the backend promote each other, and this process is marked by blue arrows ⟶.

3.2 Frontend

We illustrate the frontend process in Fig. 2. When a new frame arrives, the frontend first decides whether it should be a new keyframe. If so, the system proceeds with tracking and mapping.

Formally, given an image sequence $\{\mathbf{I}_t \in \mathbb{R}^{3\times H\times W}\}_{t\in\{1,2,\cdots\}}$, the **frontend** f_{fn} maps each frame to a set of *map tokens*:

$$\mathbf{F}_t = f_{\text{fn}}(\mathbf{I}_t)_{\{\mathbf{C}_k\}_{k\in\mathbf{S}}} \tag{1}$$

where $\{\mathbf{C}_k\}_{k\in\mathbf{S}}$ denotes the **KV cache** of previous keyframes, storing the key (K) and value (V) tensors from the inter-frame attention. Here, $\mathbf{S}$ is the set of keyframe indices, and $\mathbf{F}_t$ denotes the resulting set of map tokens for frame t, serving as an implicit neural representation of the scene. The newly generated KV cache in this process, $\mathbf{C}_t = \text{Cache}(f(\mathbf{F}_t))$, is appended to $\{\mathbf{C}_k\}_{k\in\mathbf{S}}$ for subsequent use. Optionally, task-specific heads extract the pointmap $\mathbf{P}_t$, confidence $\mathbf{\Sigma}_t$, and camera pose $\mathbf{g}_t$: $\mathbf{P}_t, \mathbf{\Sigma}_t, \mathbf{g}_t = h(\mathbf{F}_t)$.

Keyframe Detection. After generating map tokens $\mathbf{F}_t$, the frontend estimates the camera pose using the pose head h_{pose}: $\mathbf{g}_t = h_{\text{pose}}(\mathbf{F}_t)$. A frame is marked as a new keyframe if its relative pose to the most recent keyframe k_{prev}, $\mathbf{g}_{k_{\text{prev}},t} = \mathbf{g}_{k_{\text{prev}}}^{-1}\mathbf{g}_t$, exceeds a threshold τ.

In practice, keyframe detection does not rely on the KV cache; instead, we directly apply $f_{\text{fn}}(\mathbf{I}_{k_{\text{prev}}}, \mathbf{I}_t)$ to the frame pair (k_{prev}, t), which improves computational efficiency.

Frontend Tracking and Mapping. Once a new keyframe is confirmed, $\mathbf{F}_t$ is recomputed using the full KV cache as defined in Eq. (1), and the token map $(\mathbf{M}, \mathbf{S})$ is updated:

$$\mathbf{M} \leftarrow \mathbf{M} \cup \{\mathbf{F}_t\}, \quad \mathbf{S} \leftarrow \mathbf{S} \cup \{t\}. \tag{2}$$

The frontend depends only on past frames, making it causal and suitable for online tracking. However, this causal design inevitably leads to error accumulation and local inconsistencies. To mitigate this, we introduce a **backend** module for global refinement.

3.3 Backend

The backend is responsible for refining the map tokens to enforce global consistency. As illustrated in Fig. 1, traditional SLAM pipelines typically rely on loop closure detection and graph optimization to achieve this objective. In contrast, our approach employs a transformer-based **backend** f_{bn} that directly refines all map tokens in a single forward pass:

$$\bar{\mathbf{M}} = f_{\text{bn}}(\mathbf{M}). \tag{3}$$

The effectiveness of this design stems from the full attention mechanism within f_{bn}, which establishes dense interactions among all map tokens. This global receptive field enables the backend to correct accumulated drift and enforce structural coherence throughout the reconstructed scene.

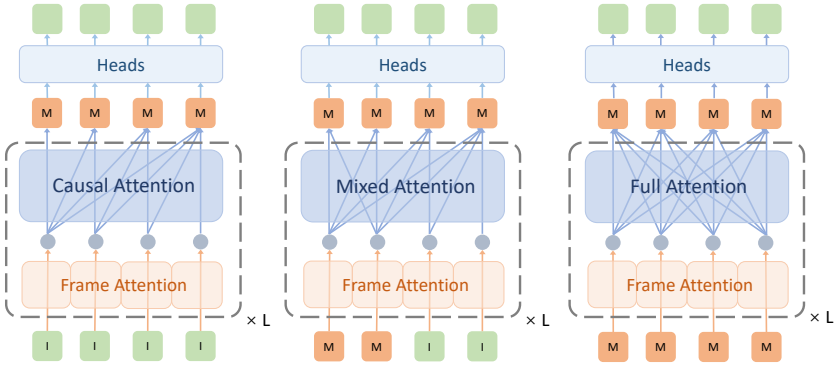
Cache Sharing. To benefit from the backend refinement, the frontend reuses the backend's shared KV cache $\mathbf{C}_{\mathbf{M}}$: $\{\mathbf{C}_k\}_{k\in\mathbf{S}} \leftarrow \mathbf{C}_{\mathbf{M}}$. Consequently, subsequent frames are tracked and mapped with respect to the refined global structure, thereby reducing the risk of error accumulation over long sequences.

3.4 Training Strategy

The training strategy is designed to enable a single transformer to support both frontend and backend SLAM functionalities, as illustrated in Fig. 3. We jointly train SLAM-Former in three modes, each corresponding to a distinct input–output configuration, within a single iteration.

Batched training under causal, mixed, and full attention is mathematically equivalent to the corresponding inference configurations: (1) frontend inference with accumulated KV caches, (2) frontend inference with mixed KV caches from backend and frontend, and (3) backend inference with full attention.

Training Frontend. The frontend is trained with a *causal attention* mask (Mode 1 in Fig. 3 (a)). During inference, it reuses KV caches from previous frames in a mathematically equivariant manner: $\mathbf{F} = f(\mathbf{I})_{\text{KV}}$. This enables efficient, end-to-end learning in a single pass.



(a) Mode 1: Frontend (b) Mode 2: Backend + Frontend (c) Mode 3: Backend

Fig. 3: Three training modes of SLAM-Former. I and M represent the patch-wise image tokens and map tokens of a frame. ● and ■ represent the layer intermediate and final output. In each mode, either I or M tokens, or both, are fed into the transformer backbone f, which contains L layers of frame attention and various inter-frame attentions. Finally, pose and pointmap are regressed by the heads h.

Training Frontend with Backend Cooperation. To bridge frontend and backend operations (Mode 2), we train f with *mixed attention* to process backend and cache sharing functionality. Specifically, the backend refines map tokens with full attention, $\bar{\mathbf{M}} = f_{\text{bn}}(\mathbf{M})$, while the frontend processes new images in the same forward pass as the backend, using causal attention that is equivalent to conditioning on the backend-refined KV cache: $\mathbf{F} = f_{\text{fn}}(\mathbf{I})_{\mathbf{C}_\mathbf{M}}$.

Training Backend. The backend (Mode 3) refines map tokens originating from different runs or KV cache states. *Full attention* is applied throughout, allowing the model to correct drift and enforce global consistency: $\bar{\mathbf{M}} = f_{\text{bn}}(\mathbf{M})$.

Joint Training. In all modes, the resulting tokens serve as implicit representations of geometry and camera poses. Task-specific heads predict the pointmap $\mathbf{P}^*$, confidence $\mathbf{\Sigma}^*$, and camera pose $\mathbf{g}^*$: $\mathbf{P}^*, \mathbf{\Sigma}^*, \mathbf{g}^* = h(\mathbf{F})$. Unlike VGGT, which predicts global geometry, SLAM-Former produces local pointmaps for each frame to avoid the need to define a specific world coordinate.

The overall loss combines depth, pointmap, and camera supervision: $L = L_{\text{depth}} + L_{\text{pmap}} + \lambda L_{\text{cam}}$. For depth loss, the predicted depth $\mathbf{D}^* = \mathbf{P}^*_z$ is supervised against ground-truth depth $\mathbf{D}$, weighted by confidence Σ^*:

$$L_{\text{depth}} = \sum_t \Big(\|\Sigma^*_t \odot (s^*\mathbf{D}^*_t - \mathbf{D}_t)\| + \|\Sigma^*_t \odot (\nabla s^*\mathbf{D}^*_t - \nabla \mathbf{D}_t)\| - \alpha \log \Sigma^*_t \Big), \quad (4)$$

where $\odot$ is element-wise multiplication, ∇ the spatial gradient, and s^* a scale factor estimated following Pi3: $s^* = \text{argmin}_s \sum_t \|(s\mathbf{P}^*_t - \mathbf{P}_t)/\mathbf{D}_t\|_1$. For pointmap loss, similar to depth loss but defined on transformed local pointmaps aligned

to the first frame: $\mathbf{P}^*_{t,1} = {\mathbf{g}^*_1}^{-1}\mathbf{g}^*_t\mathbf{P}^*_t$, the loss is designed as

$$L_{\text{pmap}} = \sum_t \Big(\|\Sigma^*_t \odot (s^*\mathbf{P}^*_{t,1} - \mathbf{P}_t)\| + \|\Sigma^*_t \odot (\nabla s^*\mathbf{P}^*_{t,1} - \nabla\mathbf{P}_t)\| - \alpha \log \Sigma^*_t \Big). \quad (5)$$

Camera loss employs a scaled Huber loss for relative pose supervision:

$$L_{\text{cam}} = \sum_{i,j} \| s^* \otimes ({\mathbf{g}^*_i}^{-1}\mathbf{g}^*_j) - (\mathbf{g}_i^{-1}\mathbf{g}_j) \|_\epsilon, \quad (6)$$

where $\otimes$ scales translations and $\|\cdot\|_\epsilon$ denotes the Huber norm. All three modes are executed sequentially in a single iteration with shared weights. The final training objective is $L_{\text{all}} = L_1 + L_2 + \beta L_3$.

3.5 Execution Pipeline

The execution pipeline integrates frontend and backend for online incremental SLAM inference, both powered by the same SLAM-Former transformer.

Frontend. Each incoming frame is first passed to the keyframe detector. Specifically, the current frame and the most recent keyframe are jointly processed by the transformer using full attention to estimate their poses. If the relative spatial distance exceeds a predefined threshold τ, the incoming frame is designated as a new keyframe.

The first two keyframes are jointly processed for initialization, producing map tokens $\{\mathbf{F}_1, \mathbf{F}_2\}$ and KV caches $\{\mathbf{C}_1, \mathbf{C}_2\}$, which are stored. For subsequent keyframes ($k > 2$), the frontend utilizes the KV caches $\{\mathbf{C}_i\}_{i<k}$ to track the k-th keyframe, generating the corresponding map tokens $\mathbf{F}_k$ and KV cache $\mathbf{C}_k$.

Backend. Periodically, after every T keyframes (at k-th), the accumulated map tokens are refined by the backend, and the resulting KV caches are used to update the frontend KV caches $\{\mathbf{C}_i\}_{i\le k}$.

3.6 KV Pruning for Runtime Efficiency

To alleviate the scalability challenge (latency and VRAM) in long-sequence processing, we introduce a KV pruning mechanism to **speed up backend**. Then reuse the pruned KV as cache to **further accelerate the frontend**.

Inspired by DivPrune [1], we formulate the token retention as a Max-Min Diversity Problem (MMDP). For each keyframe, let $\mathcal{T} = \{\mathbf{t}_1, \ldots, \mathbf{t}_P\}$ denote the P non-register patch tokens (and their associated KV pairs). We select a subset $\mathcal{S} \subset \mathcal{T}$ of size $k = \gamma \cdot P$ that maximizes the minimum pairwise diversity:

$$\mathcal{S}^* = \arg \max_{\mathcal{S}\subset\mathcal{T}, |\mathcal{S}|=k} \min_{\mathbf{t}_i,\mathbf{t}_j\in\mathcal{S}, i\neq j} d(\mathbf{t}_i, \mathbf{t}_j), \quad (7)$$

with pairwise diversity measured by cosine distance $d(\mathbf{t}_i, \mathbf{t}_j) = 1 - \mathbf{t}_i^\top \mathbf{t}_j / (\|\mathbf{t}_i\|\|\mathbf{t}_j\|)$. We adopt a greedy forward selection algorithm following [1] to iteratively select the tokens, achieving near-optimal diversity with $\mathcal{O}(P^2)$ precomputation cost.

Naively discarding tokens loses the corresponding point cloud in reconstruction. **Instead, we prune only KV while preserving all queries**. For each keyframe, $\mathbf{K}' = \mathbf{K}_{[\mathcal{S}^*]}, \quad \mathbf{V}' = \mathbf{V}_{[\mathcal{S}^*]}, \quad \mathbf{Q}' = \mathbf{Q}$, where $[\mathcal{S}^*]$ indexes the retained tokens. By sparsifying the keys and values while retaining all queries, the dominant attention computation in the frontend is reduced from $\mathcal{O}(n)$ to $\mathcal{O}(\gamma n)$ under a constant retention ratio, and to **sublinear** $o(n)$ **under a dynamic retention ratio that decays with sequence length**. Likewise, the backend complexity is reduced from $\mathcal{O}(n^2)$ to $\mathcal{O}(\gamma n^2)$ with a constant retention ratio, and to $\mathcal{O}(n) \cdot o(n)$ with the same dynamic retention ratio. This yields 2×, 4×, 8×, or greater speedups (see Sec. 4.5) while maintaining comparable performance.

4 Experiments

We evaluate SLAM-Former on multiple tasks, including camera tracking (Sec. 4.2) and dense 3D reconstruction (Sec. 4.3). Subsequently, we analyze the impact of our frontend-backend design (Sec. 4.4), and speedup with KV Pruning (Sec. 4.5).

4.1 Experimental Setup

Implementation Details. SLAM-Former has 36 transformer layers of both frame- and global-attention in total. We initialize SLAM-Former with Pi3 pretrained weights and train 10 epochs with a batch size of 32 (excluding the frozen image encoder and camera head). During training, each batch only contains 12 frames, while the test is applied on much longer sequences. For training, we utilize the AdamW optimizer with a learning rate of $1e-5$, and a cosine learning rate scheduler. In the loss function, the hyperparameters are set as $\lambda = 100$ and $\beta = 10$. We set $\tau = 0.1$ for translation, backend run every $T = 10$ keyframes. Regarding datasets, SLAM-Former is trained on ARKitScenes [2], ScanNet [7], ScanNet++ [42], HyperSim [27], BlendedMVS [41], MegaDepth [19], and MVS-Synth [13]. During each iteration, all three modes of a single SLAM-Former are trained. The entire training process takes 11 hours on 32 A100 GPUs. Evaluation is conducted on a single RTX 4090 GPU during testing.

Baselines. The baselines utilized in our experiments are categorized as Calibrated and Uncalibrated:

- Calibrated: ORB-SLAM3 [4], DeepV2D [31], DeepFactors [6], DPV-SLAM [20], DPV-SLAM++ [20], GO-SLAM [45], DROID-SLAM [32], MASt3R-SLAM [25] and NICER-SLAM [46].
- Uncalibrated: DROID-SLAM, MASt3R-SLAM, VGGT-SLAM, EC3R-SLAM [12], ViSTA-SLAM [44] and SLAM3R [21], as well as our method. We also test related methods: CUT3R [35] and StreamVGGT [47] using our keyframes.

Unless otherwise specified, evaluation is performed using all KVs.

Table 1: Root mean square error (RMSE) of absolute trajectory error (ATE) on TUM RGB-D [30] (unit: m). The * symbol indicates that the baseline is evaluated in the uncalibrated mode from the VGGT-SLAM paper [22], the + symbol indicates that the baseline is tested on our machine.

	Method	Sequence									Avg
		360	desk	desk2	floor	plant	room	rpy	teddy	xyz	
Calib.	ORB-SLAM3 [24]	×	0.017	0.210	×	0.034	×	×	×	**0.009**	N/A
	DeepV2D [31]	0.243	0.166	0.379	1.653	0.203	0.246	0.105	0.316	0.064	0.375
	DeepFactors [6]	0.159	0.170	0.253	0.169	0.305	0.364	0.043	0.601	0.035	0.233
	DPV-SLAM [20]	0.112	0.018	0.029	0.057	0.021	0.330	0.030	0.084	0.010	0.076
	DPV-SLAM++ [20]	0.132	0.018	0.029	0.050	0.022	0.096	0.032	0.098	0.010	0.054
	GO-SLAM [45]	0.089	**0.016**	0.028	0.025	0.026	0.052	**0.019**	0.048	0.010	0.035
	DROID-SLAM [32]	0.111	0.018	0.042	**0.021**	**0.016**	**0.049**	0.026	0.048	0.012	0.038
	MASt3R-SLAM [25]	**0.049**	**0.016**	**0.024**	0.025	0.020	0.061	0.027	**0.041**	**0.009**	**0.030**
Uncalib.	CUT3R$^+$ [35]	0.102	0.054	0.118	0.211	0.083	0.264	0.044	0.120	0.020	0.113
	StreamVGGT$^+$ [47]	0.088	0.063	0.105	0.604	0.070	0.633	0.025	0.081	0.015	0.187
	DROID-SLAM* [32]	0.202	0.032	0.091	0.064	0.045	0.918	0.056	0.045	0.012	0.158
	MASt3R-SLAM* [25]	0.070	0.035	0.055	0.056	0.035	0.118	0.041	0.114	0.020	0.060
	VGGT-SLAM [22]	0.071	0.025	0.040	0.141	0.023	0.102	0.030	0.034	0.014	0.053
	EC3R-SLAM [12]	0.101	0.038	0.050	**0.055**	0.097	0.101	0.045	0.125	0.018	0.070
	ViSTA-SLAM [44]	0.104	0.030	0.030	0.070	0.052	**0.067**	0.023	0.080	0.015	0.052
	Ours	**0.067**	**0.018**	**0.026**	0.079	**0.021**	0.082	**0.017**	**0.030**	**0.011**	**0.039**

4.2 3D Tracking Evaluation

We first evaluate the tracking performance of SLAM-Former on the TUM RGB-D, 7-Scenes, and Replica. We compute the Root Mean Square Error (RMSE) of Absolute Trajectory Error (ATE) for various methods under both calibrated and uncalibrated settings.

TUM RGB-D Tracking. In the TUM test, evaluations are conducted on a widely used subset of scenes. The results are summarized in Tab. 1. As shown, our model consistently outperforms most baselines in the uncalibrated setting. The superior performance in these more complex sequences, such as the room and floor that involve significant camera rotation and potential for loop closure, suggests that our backend's global refinement is particularly effective at mitigating accumulated drift. More importantly, it significantly reduced the error relative to calibrated baselines, achieving a highly competitive level.

7-Scenes Tracking. We evaluate the tracking performance on 7-Scenes in Tab. 2, where our method outperforms most baselines under uncalibrated and calibrated settings. In more complex scenes, such as the office, pumpkin and kitchen, our model achieves a notably higher performance gap compared to other methods. On average, our method outperforms all baselines.

Replica Tracking. The previous tracking experiments were conducted with real captured data, while the replica dataset is synthetic. Our approach shows substantial improvements under the uncalibrated setting, achieving an approximate 50% reduction in ATE compared to SLAM3R and outperforming all baselines, as shown in Tab. 3. However, our method performs on par with NICER-SLAM but still lags behind the traditional SLAM method, DROID-SLAM. This

Table 2: RMSE of ATE on 7-Scenes [10] (unit: m). The * symbol indicates that the baseline is evaluated in the uncalibrated mode from the VGGT-SLAM paper [22], the + symbol indicates that the baseline is tested on our machine.

	Method	Sequence							Avg
		chess	fire	heads	office	pumpkin	kitchen	stairs	
Calib.	NICER-SLAM [46]	**0.033**	0.069	0.042	0.108	0.200	**0.039**	0.108	0.086
	DROID-SLAM [32]	0.036	0.027	0.025	**0.066**	0.127	0.040	0.026	0.049
	MASt3R-SLAM [25]	0.053	**0.025**	**0.015**	0.097	**0.088**	0.041	**0.011**	**0.047**
Uncalib.	CUT3R^{+} [35]	0.046	0.043	0.055	0.120	0.096	0.061	0.086	0.073
	StreamVGGT^{+} [47]	0.048	0.036	0.030	0.117	0.094	0.063	0.179	0.081
	DROID-SLAM* [32]	0.047	0.038	0.034	0.136	0.166	0.080	0.044	0.078
	MASt3R-SLAM* [25]	0.063	0.046	0.029	0.103	0.114	0.074	0.032	0.066
	VGGT-SLAM [34]	**0.036**	**0.028**	**0.018**	0.103	0.133	0.058	0.093	0.067
	EC3R-SLAM [12]	0.050	0.042	0.059	0.113	0.143	0.065	0.050	0.075
	ViSTA-SLAM [44]	0.073	0.035	0.028	0.055	0.129	0.035	0.029	0.055
	Ours	0.039	0.033	**0.018**	**0.070**	**0.065**	**0.035**	**0.033**	**0.042**

Table 3: RMSE of ATE on Replica [29] (unit: m). The + symbol indicates that the baseline is tested on our machine.

	Method	Sequence								Avg
		Rm0	Rm1	Rm2	Of0	Of1	Of2	Of3	Of4	
Calib.	NICER-SLAM [46]	0.013	0.016	0.011	0.021	0.032	0.021	0.014	0.020	0.019
	DROID-SLAM [32]	**0.003**	**0.001**	**0.003**	**0.003**	**0.004**	**0.003**	**0.005**	**0.004**	**0.003**
Uncalib.	SLAM3R [21]	0.046	0.059	0.057	0.112	0.063	0.062	0.050	0.081	0.066
	CUT3R $^{+}$ [35]	0.145	0.243	0.127	0.159	0.230	0.162	0.088	0.204	0.170
	StreamVGGT^{+} [47]	0.113	0.163	0.077	0.076	0.070	0.180	0.153	0.168	0.125
	VGGT-SLAM $^{+}$ [22]	0.030	0.167	0.086	0.042	0.064	0.095	0.039	0.043	0.071
	DROID-SLAM [36]	0.313	0.111	0.125	0.029	0.421	0.045	0.253	0.249	0.193
	MASt3R-SLAM [25]	0.023	0.035	0.103	0.038	0.033	0.047	0.035	0.045	0.045
	EC3R-SLAM [12]	0.038	0.049	0.027	0.043	0.034	0.059	0.025	0.053	0.041
	ViSTA-SLAM [44]	0.069	0.093	0.136	0.074	0.193	0.118	0.049	0.130	0.108
	Ours	**0.030**	**0.026**	**0.027**	**0.028**	**0.029**	**0.038**	**0.028**	**0.031**	**0.030**

is because synthetic data lacks noise and blur, making the matchings sufficiently accurate for pose solving in bundle adjustment. In contrast, in the previous real-life data tests, DROID-SLAM performed on a similar level to our method.

4.3 Reconstruction Evaluation

We evaluate the reconstruction performance of our SLAM-Former on the 7-Scenes dataset following the protocol of VGGT-SLAM and on the Replica dataset following the protocol of SLAM3R.

7-Scenes Reconstruction. The reconstruction results on 7-Scenes are shown in Tab. 4. Our method exhibits a significant gap compared to other dense SLAM state-of-the-art methods. In terms of reconstruction quality, our method achieves the highest accuracy of 0.017m, while the other methods have an error that is about 47% higher than ours. In terms of completeness and chamfer distance, our method achieves 0.037m and 0.027m, respectively, still outperforming all baselines by around 30%.

Table 4: Reconstruction evaluation on 7-Scenes [10] (unit: m). @n indicates a keyframe every n images.

	Method	7-Scenes			
		ATE↓	Acc.↓	Complet.↓	Chamfer↓
Calib.	DROID-SLAM [32]	0.049	0.141	0.048	0.094
	MASt3R-SLAM [25]	0.047	0.089	0.085	0.087
	Spann3R @20 [33]	N/A	0.069	0.047	0.058
	Spann3R @2 [33]	N/A	0.124	0.043	0.084
Uncalib.	CUT3R^{+} [35]	0.073	0.032	0.047	0.040
	StreamVGGT^{+} [47]	0.081	0.058	0.057	0.057
	MASt3R-SLAM* [25]	0.066	0.068	0.045	0.056
	VGGT-SLAM [22]	0.067	0.052	0.058	0.055
	EC3R-SLAM [12]	0.075	0.025	0.054	0.040
	ViSTA-SLAM [44]	0.056	0.051	0.055	0.045
	Ours	**0.042**	**0.017**	**0.037**	**0.027**

Table 5: Reconstruction results on the Replica [29] dataset. * denotes the results reported in NICER-SLAM [46]. $^{-}$ shows the results from SLAM3R [21]. $^{+}$ represented the results from our run.

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 [36]	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 [17]	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
NICER-SLAM* [46]	2.53 / 3.04	3.93 / 4.10	3.40 / 3.42	5.49 / 6.09	3.45 / 4.42	4.02 / 4.29	3.34 / 4.03	3.03 / 3.87	3.65 / 4.16
DROID-SLAM* [32]	12.18 / 8.96	8.35 / 6.07	3.26 / 16.01	3.01 / 16.19	2.39 / 16.20	5.66 / 15.56	4.49 / 9.73	4.65 / 9.63	5.50 / 12.29
DIM-SLAM* [18]	14.19 / 6.24	9.56 / 6.45	8.41 / 12.17	10.16 / 5.95	7.86 / 8.33	16.50 / 8.28	13.01 / 6.77	13.08 / 8.62	11.60 / 7.85
GO-SLAM^{-} [45]	-	-	-	-	-	-	-	-	3.81 / 4.79
Spann3R^{-} [33]	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
SLAM3R^{-} [21]	3.19 / 2.40	3.12 / 2.34	2.72 / 2.00	4.28 / 2.60	3.17 / 2.34	3.84 / 2.78	3.90 / 3.16	4.32 / 3.36	3.57 / 2.62
CUT3R^{+} [35]	6.09 / 3.09	9.89 / 4.55	5.77 / 2.66	5.23 / 2.46	11.60 / 6.94	8.00 / 3.16	6.12 / 3.09	7.46 / 3.05	7.52 /3.62
StreamVGGT^{+} [47]	9.01 / 4.37	12.22 / 4.66	5.61 / 2.61	6.64 / 3.45	5.14 / 2.55	12.84 / 7.23	12.09 / 6.59	15.48 / 6.35	9.88 4.73
ViSTA-SLAM [44]	7.76 /2.09	11.05 /5.52	11.94/ 2.23	5.95 /2.26	26.14 /18.33	9.14/3.16	6.23/ 2.21	13.38/ 6.88	11.45/ 5.34
VGGT-SLAM^{+} [22]	3.23 / 2.66	18.92 / 15.41	15.93 / 12.37	4.01 / 2.49	3.01 / 2.32	6.93 / 5.31	2.97 / 2.53	5.19 / 3.83	7.52 / 5.86
MASt3R-SLAM [25]	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
EC3R-SLAM [12]	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
SLAM-Former (Ours)	**2.40** / **1.86**	**1.79** / **1.35**	**1.83** / **1.39**	**1.84** / **1.37**	**1.70** / **1.28**	**2.44** / **1.67**	**2.36** / **1.89**	**2.38** / **1.70**	**2.09** / **1.56**

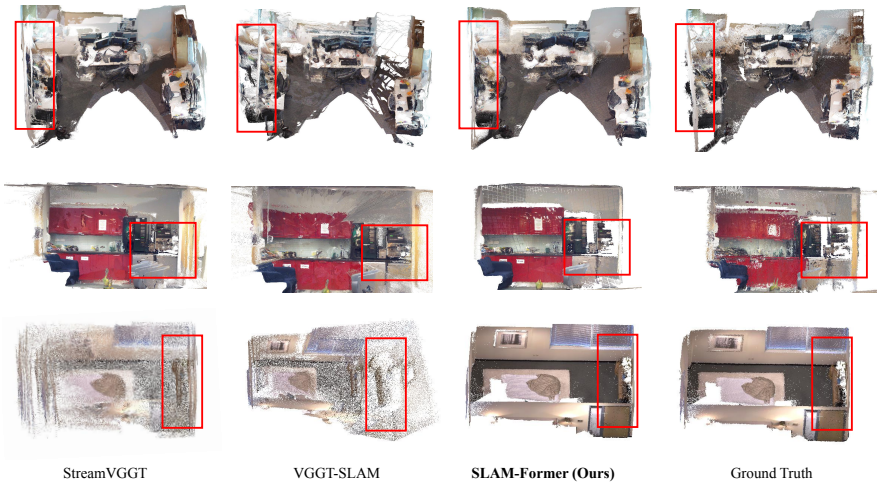


Fig. 4: Qualitative reconstruction comparison. Note the significant structural errors, such as misalignments, from baseline methods (red boxes), which are corrected by SLAM-Former's globally consistent refinement.

This consistently superior performance across all major reconstruction metrics is also demonstrated in our reconstruction demo Fig. 4. As shown in the first two rows (office and pumpkin), the baselines exhibit mismatched surfaces be-

tween frames within the red-windowed regions. In contrast, our SLAM-Former's reconstruction consistently shows coherent and accurate structure.

Replica Reconstruction. The dense reconstruction results on the Replica dataset are listed in Tab. 5. Our method also achieves the best in terms of both accuracy and completion across all baselines. Specifically, our 2.09/1.56 Acc./Comp. show large gap to the second best on both metrics.

We also demonstrate the reconstruction in the third row of Fig. 4. Here, StreamVGGT shows multiple layers of surface in the room, as highlighted in the red-windowed region. More severely, VGGT-SLAM exhibits layers with substantial scale discrepancies. While SLAM-Former closely matches the ground truth. The less dense point clouds of the baselines are caused by the mismatch of layers, as the test samples a constant number of points for evaluation. We also add qualitative comparison with MASt3R-SLAM in Supplementary Fig. C1.

4.4 Frontend and backend co-operation

To investigate how the backend design of our SLAM-Former contributes to the overall system performance in a SLAM-like manner, we conducted a series of ablation experiments. The results are summarized in Tab. 6. All evaluations are conducted on the TUM RGB-D using the RMSE of ATE as the metric.

Table 6: Evaluation of module cooperation on TUM RGB-D [30] (unit: m).

Method	Sequence									Avg
	360	desk	desk2	floor	plant	room	rpy	teddy	xyz	
Frontend-only	0.137	0.041	0.045	0.264	0.053	0.547	0.036	0.073	0.013	0.134
Frontend+Backend	**0.067**	**0.018**	**0.026**	**0.079**	**0.021**	**0.082**	**0.017**	**0.030**	**0.011**	**0.039**

The results demonstrate that the inclusion of backend modules yields significantly improved accuracy over using the frontend alone, confirming the effectiveness of our proposed frontend-backend design.

In addition to the above quantitative experiments, we also analyze how the frontend and backend mutually promote each other.

How does the backend assist the frontend? We demonstrate intermediate results on some of the most challenging sequences in datasets, including Replica-room1, ICLNUIM [11]-ofkt1, and TUM-room, all of which are inside-out captures of an indoor environment as illustrated in Fig. 5. Initially, the errors in the frontend-only results are relatively small, as highlighted by the red windows. However, as time progresses, the frontend-only reconstruction becomes significantly distorted. This distortion arises because the frontend-only **accumulates errors over time**, leading to substantial inaccuracy in the later stages. In contrast, our model, which incorporates a backend, **maintains consistency throughout the entire process**, effectively mitigating these issues.

How does the backend benefit from the frontend? To answer this question, we conduct a demonstration using the ICL-NUIM scene, ofkt0 sequence.

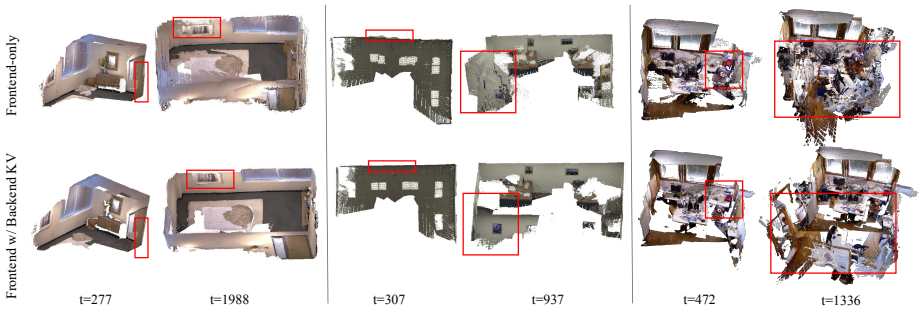
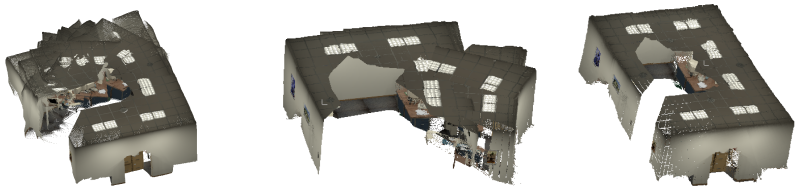


Fig. 5: Qualitative reconstruction comparison with and without backend assistance. The first row displays the frontend-only results at the corresponding timestamps, while the second row shows the results with the assistance of backend KV caches.



(a) VGGT **(b)** Pi3 **(c)** SLAM-Former

Fig. 6: Qualitative reconstruction comparison on ICL-NUIM ofkt1. The results from VGGT, Pi3, and ours are displayed from left to right, respectively. Both VGGT and Pi3 suffer from pose drift, leading to geometric inaccuracies, while our method demonstrates consistent and accurate reconstruction.

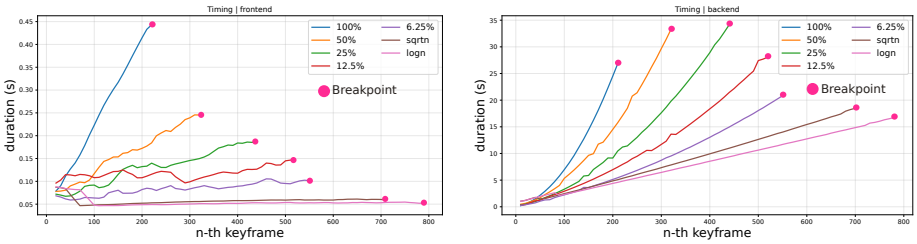
As shown in Fig. 6, the left two images display the results of VGGT and Pi3 when all keyframe images are used as input, without any sequential information. The right image shows our result. It is evident that without the sequential information provided by our frontend, VGGT and Pi3 produce disordered reconstructions. In contrast, our backend leverages the implicit order provided by the frontend to achieve a more coherent and accurate reconstruction.

4.5 Speeding up with KV Pruning

We evaluate the KV pruning on both runtime efficiency and reconstruction quality. As shown in Figure 7 and Table 7, the method delivers substantial gains in frontend and backend runtime while incurring only negligible degradation in re-

Table 7: Reconstruction evaluation on 7-Scenes [10] (unit: m). Retained ratios indicate the fraction of the original KV kept after pruning.

Retained Ratio	7-Scenes			
	ATE↓	Acc.↓	Complet.↓	Chamfer↓
100%	0.042	0.017	0.036	0.026
50%	0.041	0.017	0.036	0.027
25%	0.042	0.018	0.037	0.027
12.5%	0.042	0.020	0.038	0.029
6.25%	0.055	0.026	0.042	0.034
log KV pool	0.042	0.017	0.037	0.027
sqrt KV pool	0.041	0.017	0.037	0.027



(a) Frontend runtime **(b)** Backend runtime

Fig. 7: Frontend and backend runtime on n-th keyframes.

construction accuracy down to a 12.5% retention ratio. Noticeable degradation only appears when the retained information becomes overly sparse. Only then does the accuracy-efficiency trade-off begin to emerge.

This redundancy also suggests that information growth is sub-linear rather than linear. So our model has the potential to further reduce time complexity to for scalability. That KV-retention schedule can change hyperparameter from const-ratio to dynamic-ratio, maintaining the performance (dynamic log- or sqrt-size KV pool in Tab. 7) with $\mathcal{O}(nlog(n))$ or $\mathcal{O}(n^{1.5})$ time complexity. The time curves in Fig. 7 demonstrate the effectiveness of KV Pruning. For each curve, the final point indicates the breakpoint before reaching the memory overhead.

5 Conclusion

In this work, we introduce SLAM-Former, putting full SLAM capability into a single transformer. With alternating incremental frontend processing and global backend processing, SLAM-Former enables the frontend and backend to cooperate and enhance each other, resulting in an overall improvement. Experiments demonstrate that SLAM-Former significantly outperforms traditional geometry foundation-based SLAM methods in both tracking and reconstruction. Furthermore, it achieves highly competitive tracking performance and vastly superior reconstruction compared to traditional methods on real-world data.

Acknowledgements

This work is supported by the Beijing Municipal Science and Technology Program (Z251100008125011).

References

1. Alvar, S.R., Singh, G., Akbari, M., Zhang, Y.: Divprune: Diversity-based visual token pruning for large multimodal models. In: Proceedings of the Computer Vision and Pattern Recognition Conference (2025)
2. Baruch, G., Chen, Z., Dehghan, A., Dimry, T., Feigin, Y., Fu, P., Gebauer, T., Joffe, B., Kurz, D., Schwartz, A., et al.: Arkitscenes: A diverse real-world dataset for 3d indoor scene understanding using mobile rgb-d data. arXiv preprint arXiv:2111.08897 (2021)

3. Bloesch, M., Czarnowski, J., Clark, R., Leutenegger, S., Davison, A.J.: Codeslam-learning a compact, optimisable representation for dense visual slam. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2560–2568 (2018)
4. Campos, C., Elvira, R., Rodríguez, J.J.G., Montiel, J.M., Tardós, J.D.: Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. IEEE transactions on robotics **37**(6), 1874–1890 (2021)
5. Chen, Z., Qin, M., Yuan, T., Liu, Z., Zhao, H.: Long3r: Long sequence streaming 3d reconstruction. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5273–5284 (2025)
6. Czarnowski, J., Laidlow, T., Clark, R., Davison, A.J.: Deepfactors: Real-time probabilistic dense monocular slam. IEEE Robotics and Automation Letters **5**(2), 721–728 (2020)
7. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: Scannet: Richly-annotated 3d reconstructions of indoor scenes. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 5828–5839 (2017)
8. Elseberg, J., Borrmann, D., Nüchter, A.: One billion points in the cloud–an octree for efficient processing of 3d laser scans. ISPRS Journal of Photogrammetry and Remote Sensing **76**, 76–88 (2013)
9. Engel, J., Schöps, T., Cremers, D.: Lsd-slam: Large-scale direct monocular slam. In: European conference on computer vision. pp. 834–849. Springer (2014)
10. Glocker, B., Izadi, S., Shotton, J., Criminisi, A.: Real-time rgb-d camera relocalization. In: 2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR). pp. 173–179. IEEE (2013)
11. Handa, A., Whelan, T., McDonald, J., Davison, A.J.: A benchmark for rgb-d visual odometry, 3d reconstruction and slam. In: 2014 IEEE international conference on Robotics and automation (ICRA). pp. 1524–1531. IEEE (2014)
12. Hu, L., Oufroukh, N.A., Bonardi, F., Ghandour, R.: Ec3r-slam: Efficient and consistent monocular dense slam with feed-forward 3d reconstruction. arXiv preprint arXiv:2510.02080 (2025)
13. Huang, P.H., Matzen, K., Kopf, J., Ahuja, N., Huang, J.B.: Deepmvs: Learning multi-view stereopsis. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2821–2830 (2018)
14. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Trans. Graph. **42**(4), 139–1 (2023)
15. Koestler, L., Yang, N., Zeller, N., Cremers, D.: Tandem: Tracking and dense mapping in real-time using deep multi-view stereo. In: Conference on Robot Learning. pp. 34–45. PMLR (2022)
16. Lan, Y., Luo, Y., Hong, F., Zhou, S., Chen, H., Lyu, Z., Yang, S., Dai, B., Loy, C.C., Pan, X.: Stream3r: Scalable sequential 3d reconstruction with causal transformer. arXiv preprint arXiv:2508.10893 (2025)
17. Leroy, V., Cabon, Y., Revaud, J.: Grounding image matching in 3d with mast3r. In: European Conference on Computer Vision. pp. 71–91. Springer (2024)
18. Li, H., Gu, X., Yuan, W., Yang, L., Dong, Z., Tan, P.: Dense rgb slam with neural implicit maps. arXiv preprint arXiv:2301.08930 (2023)
19. Li, Z., Snavely, N.: Megadepth: Learning single-view depth prediction from internet photos. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2041–2050 (2018)
20. Lipson, L., Teed, Z., Deng, J.: Deep patch visual slam. In: European Conference on Computer Vision. pp. 424–440. Springer (2024)

21. Liu, Y., Dong, S., Wang, S., Yin, Y., Yang, Y., Fan, Q., Chen, B.: Slam3r: Real-time dense scene reconstruction from monocular rgb videos. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 16651–16662 (2025)
22. Maggio, D., Lim, H., Carlone, L.: Vggt-slam: Dense rgb slam optimized on the sl (4) manifold. arXiv preprint arXiv:2505.12549 (2025)
23. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM **65**(1), 99–106 (2021)
24. Mur-Artal, R., Montiel, J.M.M., Tardos, J.D.: Orb-slam: A versatile and accurate monocular slam system. IEEE transactions on robotics **31**(5), 1147–1163 (2015)
25. Murai, R., Dexheimer, E., Davison, A.J.: Mast3r-slam: Real-time dense slam with 3d reconstruction priors. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 16695–16705 (2025)
26. Newcombe, R.A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A.J., Kohi, P., Shotton, J., Hodges, S., Fitzgibbon, A.: Kinectfusion: Real-time dense surface mapping and tracking. In: 2011 10th IEEE international symposium on mixed and augmented reality. pp. 127–136. Ieee (2011)
27. Roberts, M., Ramapuram, J., Ranjan, A., Kumar, A., Bautista, M.A., Paczan, N., Webb, R., Susskind, J.M.: Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 10912–10922 (2021)
28. Rosinol, A., Leonard, J.J., Carlone, L.: Nerf-slam: Real-time dense monocular slam with neural radiance fields. In: 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 3437–3444. IEEE (2023)
29. Straub, J., Whelan, T., Ma, L., Chen, Y., Wijmans, E., Green, S., Engel, J.J., Mur-Artal, R., Ren, C., Verma, S., et al.: The replica dataset: A digital replica of indoor spaces. arXiv preprint arXiv:1906.05797 (2019)
30. Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A benchmark for the evaluation of rgb-d slam systems. In: 2012 IEEE/RSJ international conference on intelligent robots and systems. pp. 573–580. IEEE (2012)
31. Teed, Z., Deng, J.: Deepv2d: Video to depth with differentiable structure from motion. arXiv preprint arXiv:1812.04605 (2018)
32. Teed, Z., Deng, J.: Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras. Advances in neural information processing systems **34**, 16558–16569 (2021)
33. Wang, H., Agapito, L.: 3d reconstruction with spatial memory. In: 2025 International Conference on 3D Vision (3DV). pp. 78–89. IEEE (2025)
34. Wang, J., Chen, M., Karaev, N., Vedaldi, A., Rupprecht, C., Novotny, D.: Vggt: Visual geometry grounded transformer. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 5294–5306 (2025)
35. Wang, Q., Zhang, Y., Holynski, A., Efros, A.A., Kanazawa, A.: Continuous 3d perception model with persistent state. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 10510–10522 (2025)
36. Wang, S., Leroy, V., Cabon, Y., Chidlovskii, B., Revaud, J.: Dust3r: Geometric 3d vision made easy. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 20697–20709 (2024)
37. Wang, Y., Zhou, J., Zhu, H., Chang, W., Zhou, Y., Li, Z., Chen, J., Pang, J., Shen, C., He, T.: π^3: Scalable Permutation-Equivariant Visual Geometry Learning. arXiv e-prints pp. arXiv–2507 (2025)

38. Yan, C., Qu, D., Xu, D., Zhao, B., Wang, Z., Wang, D., Li, X.: Gs-slam: Dense visual slam with 3d gaussian splatting. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 19595–19604 (2024)
39. Yang, J., Sax, A., Liang, K.J., Henaff, M., Tang, H., Cao, A., Chai, J., Meier, F., Feiszli, M.: Fast3r: Towards 3d reconstruction of 1000+ images in one forward pass. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 21924–21935 (2025)
40. Yao, Y., Luo, Z., Li, S., Fang, T., Quan, L.: Mvsnet: Depth inference for unstructured multi-view stereo. In: Proceedings of the European conference on computer vision (ECCV). pp. 767–783 (2018)
41. Yao, Y., Luo, Z., Li, S., Zhang, J., Ren, Y., Zhou, L., Fang, T., Quan, L.: Blendedmvs: A large-scale dataset for generalized multi-view stereo networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 1790–1799 (2020)
42. Yeshwanth, C., Liu, Y.C., Nießner, M., Dai, A.: Scannet++: A high-fidelity dataset of 3d indoor scenes. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 12–22 (2023)
43. Yuan, Y., Bleier, M., Nüchter, A.: Scenefactory: A workflow-centric and unified framework for incremental scene modeling. IEEE Transactions on Robotics **41**, 3183–3201 (2025)
44. Zhang, G., Qian, S., Wang, X., Cremers, D.: Vista-slam: Visual slam with symmetric two-view association. In: 2026 International Conference on 3D Vision (3DV). pp. 396–406. IEEE (2026)
45. Zhang, Y., Tosi, F., Mattoccia, S., Poggi, M.: Go-slam: Global optimization for consistent 3d instant reconstruction. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 3727–3737 (2023)
46. Zhu, Z., Peng, S., Larsson, V., Cui, Z., Oswald, M.R., Geiger, A., Pollefeys, M.: Nicer-slam: Neural implicit scene encoding for rgb slam. In: 2024 International Conference on 3D Vision (3DV). pp. 42–52. IEEE (2024)
47. Zhuo, D., Zheng, W., Guo, J., Wu, Y., Zhou, J., Lu, J.: Streaming 4d visual geometry transformer. arXiv preprint arXiv:2507.11539 (2025)