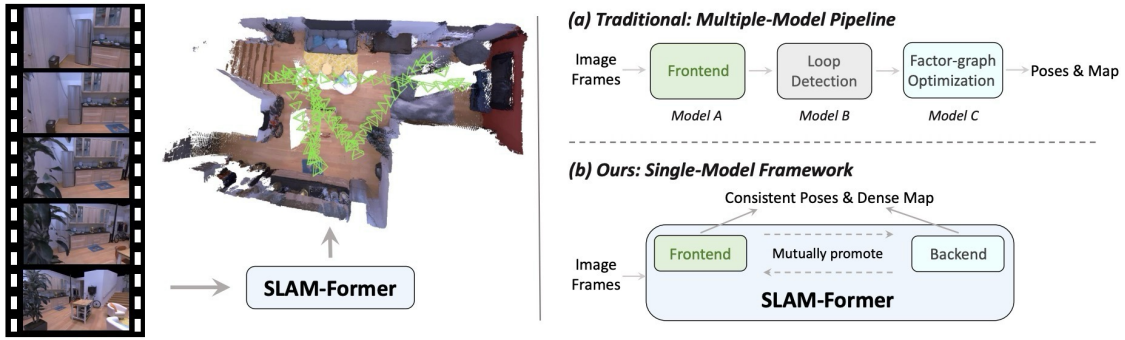
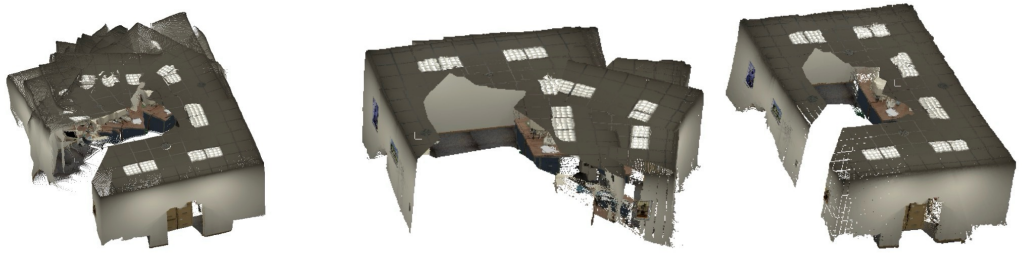




(a) VGGT (b) Pi3 (c) SLAM-Former

SLAM-Former: Putting SLAM into One Transformer

Yijun Yuan Zhuoguang Chen Kenan Li Weibang Wang Hang Zhao

IIIS, Tsinghua University

ECCV 2026

Presenter: 송우석

- **site:** https://tsinghua-mars-lab.github.io/SLAM-Former
- **github:** https://github.com/Tsinghua-MARS-Lab/SLAM-Former
- **paper:** https://arxiv.org/abs/2509.16909

RAISE LAB

Index

- Problem & Motivation
- Method
- Experiments
- Limitation

SLAM

- plays great significance in robotics perception
- simultaneuous mapping & localization → necessary for robot navigation

Dense SLAM

- sparse SLAM is efficient and robust, but has lack of details about surroundings
- significant progress thanks to Deep Learning → absence of depth sensor
  - there were some reasearches about reducing computational cost of depth estimation
- NeRF, GS SLAM
  - highly realistic novel view synthesis
  - but it's time consuming, highly sensitive to blur and noise

  ⇒ limited in real-life usage
- Foundational Geometry Techniques
  - MASt3R-SLAM: MASt3R + traditional SLAM pipeline
  - VGGT-SLAM: submap + SL(4) manifold

  ⇒ rely on pair-wise, submap-wise optimization respectively

Feed-forward 3D Reconstruction

- DUSt3R 🔥
  - regress 3D structure with scalable training data (led a trend)

  ⇒ limited in pair-wise input, so it requires global optimization (lowers the inference efficiency)

- Feed-forward Multi-view

  - **Fast3R:** efficiently handling hundreds of images
  - **VGGT:** 3D multi-task learning with scalable training data
  - **Pi3:** permutation-equivariant desgin that removes dependence on a fixed reference view

  - process multi-view images wth single forward pass, avoiding cost of post-processing global optimization
  - transformer based methods for multi-view pointmap estimation

- Real-time 3D Reconstruction

  - **Span3R:** extends DUST3R to stream while maintaining & interacting with spatial memory
  - **CUT3R:** persistent state token, transformer based recurrent update
  - **LONG3R:** 3D spatio-temporal memory, coarse-to-fine pipeline to handle long sequence

- Stream VGGT, STream3R
  - inspired by modern language model
  - incorporate causal attention to enable real-time streaming reconstruction

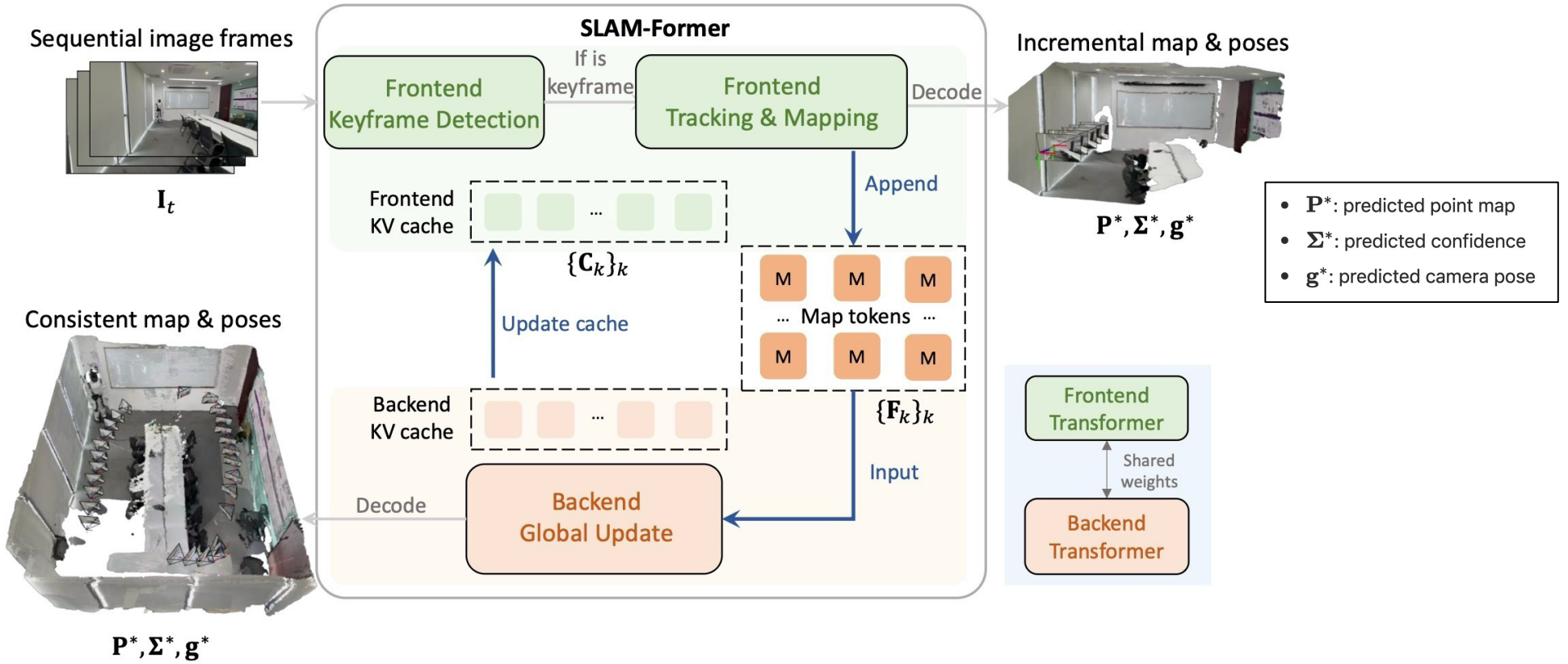
  ⇒ but, exsisting streaming methods don’t cosider revisiting past estimates, so it leads to drift and limited global consistency

SLAM-Former

- Architecture: Frontend + Backend
  - efficient incremental updates
  - periodic global refinement

Frontend
- keyframe detection
- Tracking & Mapping

Backend
- Global Update



SLAM-Former

- Transformer Architecture (f)
  - transformer backbone that aggregates intra/inter frame information
  - consists of L layers, and each layer has intra/inter frame attention
  - **input:** image patch tokens + register tokens

- Task-specific Heads (h)
  - decode scene geometry, camera poses

Frontend

- map token

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

  - mapping keyframe into map token using previous KV cache
    (save KV Cache before passing to inter-frame attention layer)
  - mapping keyframe into map-token using previous KV cache

$$\mathbf{C}_t = \text{Cache}(f(\mathbf{F}_t))$$

- keyframe detection

$$\mathbf{g}_t = h_{\text{pose}}(\mathbf{F}_t) \quad (2)$$

  - decode map token into camera pose

$$\mathbf{g}_{k_{\text{prev}},t} = \mathbf{g}_{k_S}^{-1}\mathbf{g}_t > \tau$$

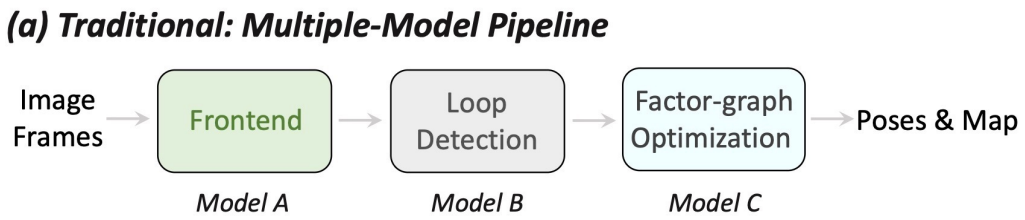
  - select frame as new keyframe if relative pose of previous keyframe exceeds threshold τ
  - if a new keyframe is confirmed, $\mathbf{F}_t$ is recomputed with full KV cache, and token map $(\mathbf{M}, \mathbf{S})$ is updated

$$\mathbf{M} \leftarrow \mathbf{M} \cup \{\mathbf{F}_t\}, \quad \mathbf{S} \leftarrow \mathbf{S} \cup \{t\}, \quad S \leftarrow S + 1. \quad (3)$$

To mitigate a causality(relies on past frames) of frontend, we use backend

Backend

- Traditional SLAM pipeline



  - traditional SLAM pipeline relies on loop closure detection & graph optimization for global consistency

- SLAM-Former
  - but, SLAM-Former refines whole map token in a single pass using backend f_{bn}

$$\bar{\mathbf{M}} = f_{\text{bn}}(\mathbf{M}) \tag{4}$$

  - effectiveness lies in the full attention mechanism within f_{bn}

  - Cache Sharing

$$\{\mathbf{C}_k\}_{k\in\mathbf{S}} \leftarrow \mathbf{C}_{\mathbf{M}}. \tag{5}$$

  - frontend and backend shares KV cache $\mathbf{C}_{\mathbf{M}}$ (so that frontend get benefits of backend refinement)

Training Strategy

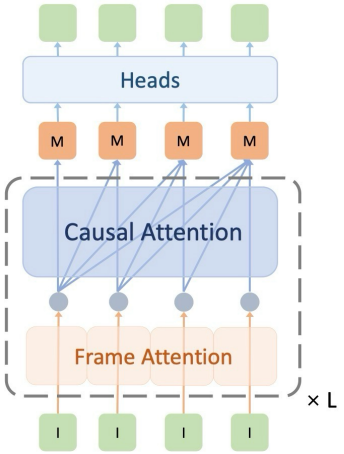
- Training Modes
  - jointly train SLAM-Former across three modes with a single iteration

- Mode 1 (Training Frontend)
  - **Training:** causal attention mask
  - **Inference:** single pass with previous KV caches

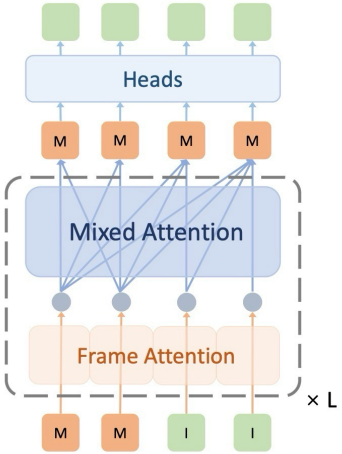
    $$\mathbf{F} = f(\mathbf{I})_{\mathrm{KV}}$$
  - first two frames are applied full attention (removing dependence on a fixed reference view)
  - subsequent frames are applied causal attention

- Mode 2 (Training Frontend with Backend Cooperation)
  - train with mixed attention to simultaneously process backend and cache-sharing
  - **backend:** full attention $\bar{\mathbf{M}} = f_{\mathrm{bn}}(\mathbf{M})$
  - **frontend:** causal attention $\mathbf{F} = f_{\mathrm{fn}}(\mathbf{I})_{\mathbf{C}_{\mathbf{M}}}$

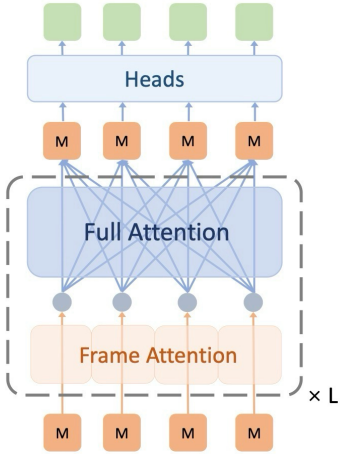
- Mode 3 (Training Backend)
  - refines map token $\bar{\mathbf{M}} = f_{\mathrm{bn}}(\mathbf{M})$
  - **backend:** full attention



(a) Mode 1: Frontend



(b) Mode 2: Backend + Frontend



(c) Mode 3: Backend

Training Strategy

- Joint Training
  - resulting tokens in all modes (=implicit representations of geometry and camera poses)

$$\mathbf{P}^*, \mathbf{\Sigma}^*, \mathbf{g}^* = h(\mathbf{F}) \tag{6}$$

  - produces pointmaps for each frame to avoid the need to define specific world coordinate

- Loss Function

$$L = L_{\text{depth}} + L_{\text{pmap}} + \lambda L_{\text{cam}} \tag{7}$$

  - depth loss

scale factor (by Pi3)

predicted depth

$$L_{\text{depth}} = \sum_t \left(\|\Sigma_t^* \odot (s^* \mathbf{D}_t^* - \mathbf{D}_t)\| + \|\Sigma_t^* \odot (\nabla s^* \mathbf{D}_t^* - \nabla \mathbf{D}_t)\| - \alpha \log \Sigma_t^* \right)$$

confidence

spatial gradient

  - pointmap loss

$$L_{\text{pmap}} = \sum_t \left(\|\Sigma_t^* \odot (s^* \mathbf{P}_{t,1}^* - \mathbf{P}_t)\| + \|\Sigma_t^* \odot (\nabla s^* \mathbf{P}_{t,1}^* - \nabla \mathbf{P}_t)\| - \alpha \log \Sigma_t^* \right)$$

  - camera loss

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

Huber norm

$$L_{\text{all}} = L_1 + L_2 + \beta L_3$$

final training objective

$$s^* = \text{argmin}_s \sum_t \|s(\mathbf{P}_t^* - \mathbf{P}_t)/\mathbf{D}_t\|_1$$

scale factor (by Pi3)

$$\mathbf{P}_{t,1}^* = \mathbf{g}_1^{*-1} \mathbf{g}_t^* \bar{\mathbf{P}}_t^*$$

transformed local pointmaps aligned to the first frame

Experiments

- Camera pose estimation

[TUM RGB-D dataset]

	Method	Sequence									Avg
		360	desk	desk2	floor	plant	room	rpy	teddy	xyz	
Calib.	ORB-SLAM3 [1]	×	0.017	0.210	×	0.034	×	×	×	**0.009**	N/A
	DeepV2D [28]	0.243	0.166	0.379	1.653	0.203	0.246	0.105	0.316	0.064	0.375
	DeepFactors [15]	0.159	0.170	0.253	0.169	0.305	0.364	0.043	0.601	0.035	0.233
	DPV-SLAM [29]	0.112	0.018	0.029	0.057	0.021	0.330	0.030	0.084	0.010	0.076
	DPV-SLAM++ [29]	0.132	0.018	0.029	0.050	0.022	0.096	0.032	0.098	0.010	0.054
	GO-SLAM [30]	0.089	**0.016**	0.028	0.025	0.026	0.052	**0.019**	0.048	0.010	0.035
	DROID-SLAM [5]	0.111	0.018	0.042	**0.021**	**0.016**	**0.049**	0.026	0.048	0.012	0.038
	MASt3R-SLAM [7]	**0.049**	**0.016**	**0.024**	0.025	0.020	0.061	0.027	**0.041**	**0.009**	**0.030**
Uncalib.	DROID-SLAM* [5]	0.202	0.032	0.091	0.064	0.045	0.918	0.056	0.045	0.012	0.158
	MASt3R-SLAM* [7]	0.070	0.035	0.055	**0.056**	0.035	0.118	0.041	0.114	0.020	0.060
	VGGT-SLAM [8]	0.071	0.025	0.040	0.141	0.023	0.102	0.030	0.034	0.014	0.053
	CUT3R^{+} [26]	0.102	0.054	0.118	0.211	0.083	0.264	0.044	0.120	0.020	0.113
	StreamVGGT^{+} [11]	0.088	0.063	0.105	0.604	0.070	0.633	0.025	0.081	0.015	0.187
	VGGT-SLAM^{+} [10]	0.053	0.024	0.040	0.335	0.022	0.166	0.040	0.037	0.041	0.084
	Ours	**0.067**	**0.018**	**0.026**	0.079	**0.021**	**0.082**	**0.017**	**0.030**	**0.011**	**0.039**

[Replica dataset]

	Method	Sequence								Avg
		Rm0	Rm1	Rm2	Of0	Of1	Of2	Of3	Of4	
Calib.	NICER-SLAM [13]	0.013	0.016	0.011	0.021	0.032	0.021	0.014	0.020	0.019
	DROID-SLAM [5]	**0.003**	**0.001**	**0.003**	**0.003**	**0.004**	**0.003**	**0.005**	**0.004**	**0.003**
Uncalib.	SLAM3R [33]	0.046	0.059	0.057	0.112	0.063	0.062	0.050	0.081	0.066
	CUT3R^{+} [26]	0.145	0.243	0.127	0.159	0.230	0.162	0.088	0.204	0.170
	StreamVGGT^{+} [11]	0.113	0.163	0.077	0.076	0.070	0.180	0.153	0.168	0.125
	VGGT-SLAM^{+} [8]	0.030	0.167	0.086	0.042	0.064	0.095	0.039	0.043	0.071
	Ours	**0.030**	**0.026**	**0.027**	**0.028**	**0.029**	**0.038**	**0.028**	**0.031**	**0.030**

- SLAM-Former outperforms overall baseline methods in TUM RGB-D dataset
- but, lags behind the traditional SLAM method in Replica dataset (because synthetic data lacks noise and blur)

Experiments

- Reconstruction

[Replic dataset]

Method	Room 0 Acc. / Comp.	Room 1 Acc. / Comp.	Room 2 Acc. / Comp.	Office 0 Acc. / Comp.	Office 1 Acc. / Comp.	Office 2 Acc. / Comp.	Office 3 Acc. / Comp.	Office 4 Acc. / Comp.	Average Acc. / Comp.
DUSt3R [9]	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 [22]	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* [13]	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* [5]	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* [43]	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^{-} [30]	-	-	-	-	-	-	-	-	3.81 / 4.79
Spann3R^{-} [25]	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^{-} [33]	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^{+} [26]	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^{+} [11]	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
VGGT-SLAM^{+} [8]	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
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**

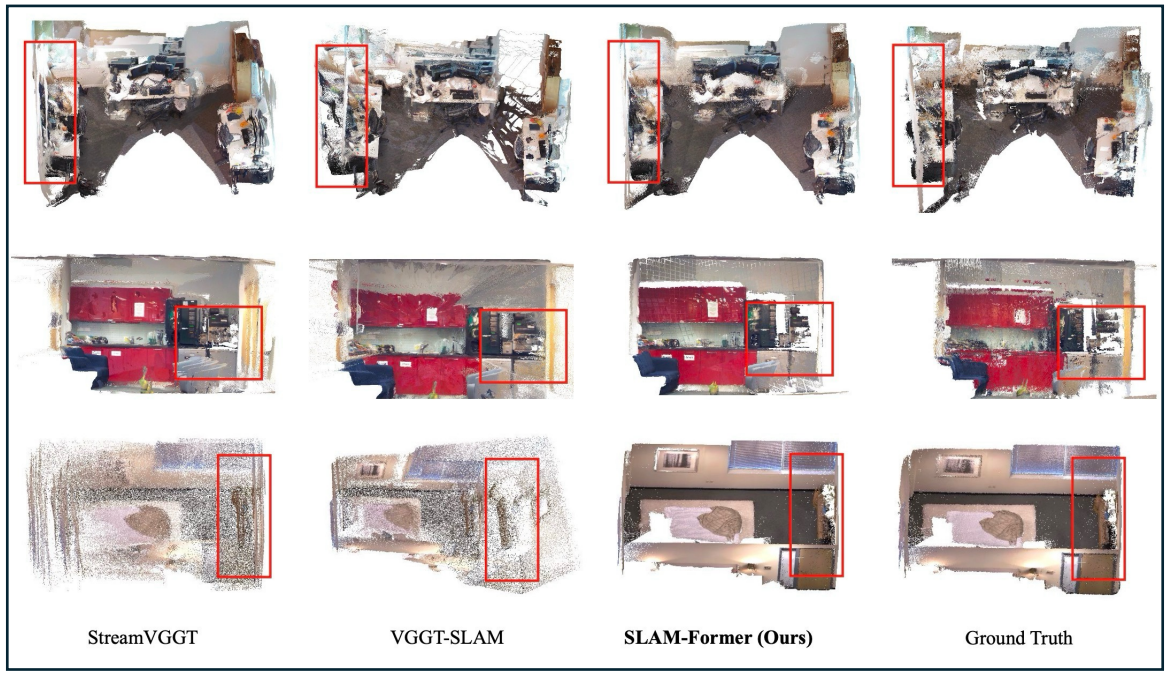
[7-Scenes dataset]

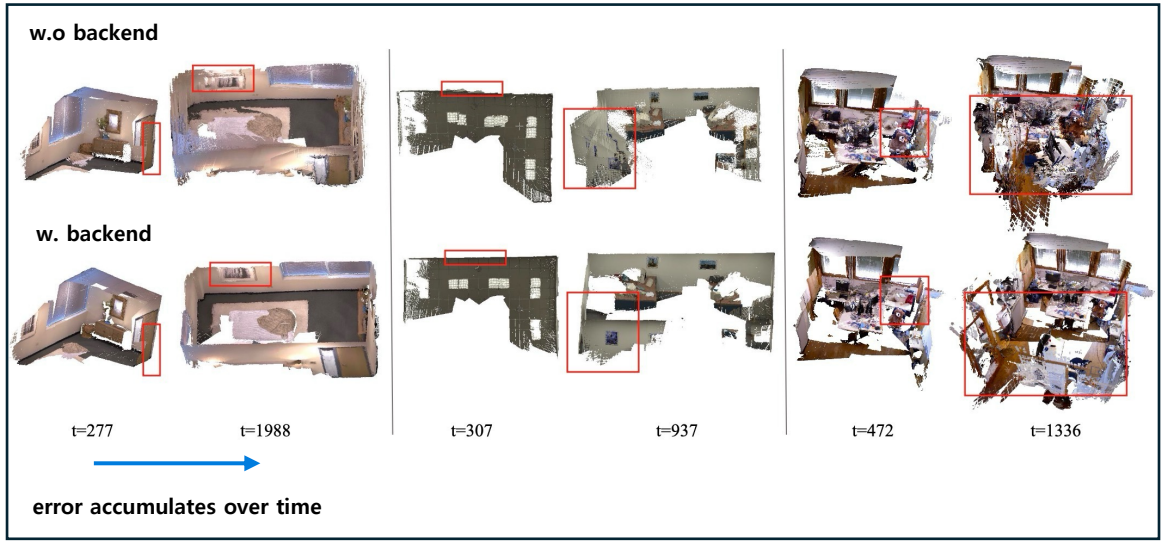
	Method	7-Scenes ATE↓	Acc.↓	Complet.↓	Chamfer↓
Calib.	DROID-SLAM [5]	0.049	0.141	0.048	0.094
	MASt3R-SLAM [7]	0.047	0.089	0.085	0.087
	Spann3R @20 [25]	N/A	0.069	0.047	0.058
	Spann3R @2 [25]	N/A	0.124	0.043	0.084
Uncalib.	MASt3R-SLAM* [7]	0.066	0.068	0.045	0.056
	VGGT-SLAM [8]	0.067	0.052	0.058	0.055
	CUT3R^{+} [26]	0.073	0.032	0.047	0.040
	StreamVGGT^{+} [11]	0.081	0.058	0.057	0.057
	VGGT-SLAM^{+} [8]	0.068	0.054	0.060	0.057
	Ours	**0.042**	**0.017**	**0.037**	**0.027**

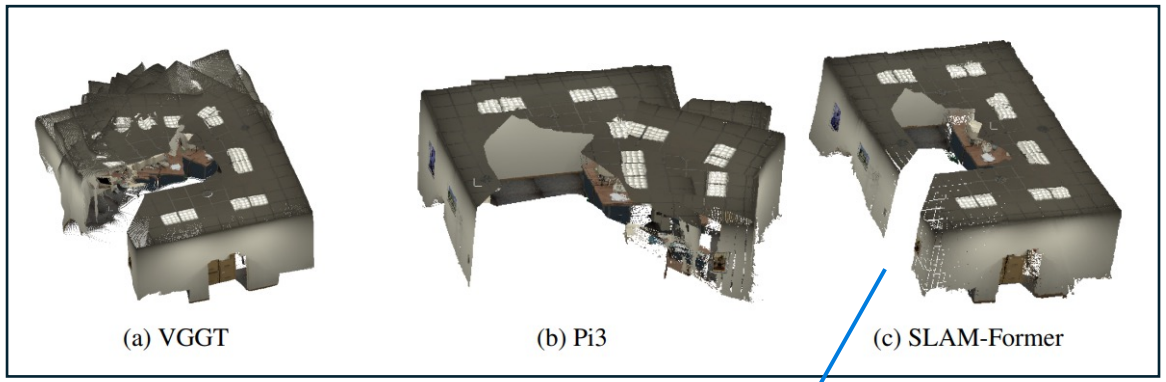
- SLAM-Former outperforms overall baseline methods in Replica, 7-Scenes dataset

Experiments

- Reconstruction







backend use implicit order provided by frontend

Experiments

- Ablation

Method	Sequence									Avg
	360	desk	desk2	floor	plant	room	rpy	teddy	xyz	
F.-only	0.137	0.041	0.045	0.264	0.053	0.547	0.036	0.073	0.013	0.134
F. + EB.	0.073	0.021	0.026	0.078	0.022	0.104	0.018	0.027	0.011	0.042
F. + MB.	0.067	0.018	0.025	0.081	0.023	0.082	0.017	0.031	0.011	**0.039**
F. + MB. + EB.	**0.067**	**0.018**	**0.026**	**0.079**	**0.021**	**0.082**	**0.017**	**0.030**	**0.011**	**0.039**

- **EB. :** End Backend
- **MB :** Middle Backend

- evaluate camera pose on TUM RGB-D
- backend significantly improves performance
- MB. shows great contribution

- Execution Speed

	KF-detection TPE(ms)	Frontend TPE(ms)	Backend TPE(ms)	Summary FPS
TUM: room	89	97	187	10.8
7Scene: chess	89	77	83	11.0
Replica: room1	87	76	113	11.3

- overall speed is over 10Hz, so SLAM-Former can operate in real-time

Limitation

1. SLAM-Former use full-attention in the backend ($= O(n^2)$)
   - could be solved in SLAM ways(using sparse graph) or transformer techniques (using sparse attention & token merge)

2. SLAM-Former doesn't support frontend mode locally
   - all previous KV caches should be fed into the model during inference