TopoMA: Topology-Guided Multi-Agent Dense RGB 3D Reconstruction via Distributed Inference

Supplementary Material

1. Experimental Results: Runtime, Memory, and Reconstruction Metrics

In the main paper, we primarily evaluate TopoMA in terms of trajectory accuracy (ATE) and reconstruction quality (Depth L1 and Accuracy). However, a distributed multi-agent system is also judged by its runtime efficiency and resource footprint, as well as its ability to maintain reconstruction fidelity under stronger geometric metrics such as Completeness and Chamfer distance. To provide a more comprehensive picture and to justify the claim that TopoMA is both accurate and practical for large-scale deployments, we report additional experiments in this supplementary section.

First, we compare runtime (FPS) and memory usage (GPU/CPU) against single-agent and multi-agent baselines on KITTI [4], ScanNet [2], and Replica [9], demonstrating that our topology-aware design and residual transport enable efficient, server-free multi-agent reconstruction with competitive or better resource usage. Second, we extend the reconstruction evaluation by reporting Completeness and Chamfer distance on ScanNet [2] and Replica [9], confirming that the improvements reported in the main paper translate to consistent gains under stronger geometric criteria. Finally, we explicitly compare TopoMA with multi-agent baselines MAGiC-SLAM [10] and CP-SLAM [5] on KITTI [4] and ScanNet [2], highlighting that our method achieves slightly better accuracy while maintaining favorable runtime and memory profiles.

1.1. Runtime and Memory Efficiency

KITTI multi-agent efficiency. Table 1 reports average runtime and memory usage for all methods under the same multi-agent splitting protocol on KITTI [4]. TopoMA attains higher FPS than most baselines, including VGGT-Long, while consuming less GPU and CPU memory than other multi-agent methods such as MAGiC-SLAM [10] and CP-SLAM [5], illustrating that topology-guided residual transport does not incur prohibitive overhead.

ScanNet indoor efficiency. Table 2 shows that TopoMA maintains high frame rates on ScanNet [2] while using less memory than other multi-agent systems. This confirms that the residual transport and topology-skeleton sparsification effectively constrain the global optimization budget even in cluttered indoor scenes.

Table 1. Runtime and resource usage on KITTI [4] multi-agent experiments. We report average FPS, GPU memory, and CPU memory across all evaluated KITTI sequences. Higher FPS and lower memory usage are better.

Method	FPS [Hz] ↑	GPU [GB] ↓	CPU [GB] ↓
Single-Agent Methods			
VGGT-Long[3]	5.83	6.27	10.64
TTT3R[1]	5.41	6.52	10.91
SLAM3R[6]	5.07	6.81	11.13
MASt3R-SLAM[8]	4.92	7.08	11.32
VGGT-SLAM[7]	4.79	7.29	11.47
Multi-Agent Methods			
MAGiC-SLAM[10]	4.63	7.56	11.82
CP-SLAM[5]	4.71	7.43	11.68
Ours (TopoMA)	6.12	5.97	10.09

Table 2. Runtime and resource usage on ScanNet [2] indoor scenes. We report average FPS and memory usage over all evaluated ScanNet sequences.

Method	FPS [Hz] ↑	GPU [GB] ↓	CPU [GB] ↓
Single-Agent Methods			
VGGT-Long[3]	5.18	6.83	11.23
TTT3R[1]	4.87	7.02	11.49
SLAM3R[6]	4.59	7.19	11.81
MASt3R-SLAM[8]	4.31	7.41	12.04
VGGT-SLAM[7]	4.08	7.63	12.19
Multi-Agent Methods			
MAGiC-SLAM[10]	3.93	7.88	12.53
CP-SLAM[5]	4.02	7.77	12.31
Ours (TopoMA)	5.53	6.71	10.86

Replica efficiency. Fig. 2 is the reconstruction results on the Replica [9] and Table 3 summarizes runtime and memory usage on Replica. Here TopoMA achieves a favorable trade-off: FPS close to the fastest baselines and clearly lower GPU/CPU usage than other multi-agent methods, consistent with the ablations in the main paper that show the benefit of residual transport and decentralized loop closure.

1.2. Extended Reconstruction Metrics

While the main paper reports Depth L1 and surface Accuracy, these metrics do not fully capture the geometric completeness of reconstructed scenes. To this end, we further evaluate Completeness (Comp) on ScanNet [2] and

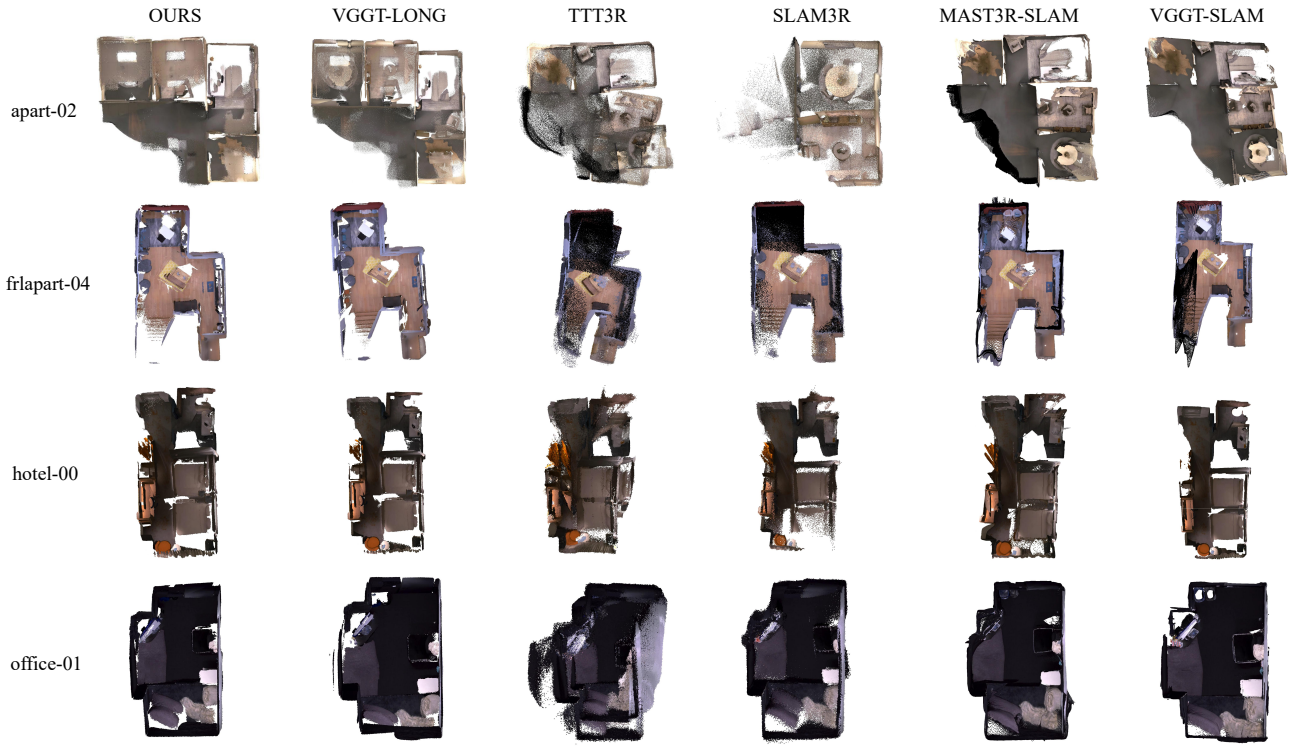

Figure 2. Indoor reconstruction results on the Replica [9]. The first column shows reconstructions produced by TopoMA, while the remaining columns display outputs from VGGT-Long[3], TTT3R[1], SLAM3R[6], MASt3R-SLAM[8], and VGGT-SLAM[7].

Table 3. Runtime and resource usage on Replica [9] scenes. All values are averaged over all evaluated Replica sequences under the same multi-agent protocol.

Method	FPS [Hz] ↑	GPU [GB] ↓	CPU [GB] ↓
Single-Agent Methods			
VGGT-Long[3]	6.03	6.73	10.57
TTT3R[1]	6.42	6.13	10.12
SLAM3R[6]	6.05	6.39	10.56
MASt3R-SLAM[8]	5.71	6.82	11.03
VGGT-SLAM[7]	5.54	7.04	11.29
Multi-Agent Methods			
MAGiC-SLAM[10]	5.27	7.38	11.71
CP-SLAM[5]	5.34	7.09	11.46
Ours (TopoMA)	6.57	5.94	10.04

Table 4. Extended reconstruction metrics on ScanNet [2]. We report average Depth L1, Accuracy (Acc), and Completeness (Comp), aggregated over all evaluated ScanNet sequences. Lower is better.

Method	Depth L1 [cm] ↓	Acc [cm] ↓	Comp [cm] ↓
Single-Agent Methods			
VGGT-Long[3]	60.83	28.81	36.40
TTT3R[1]	26.08	16.98	23.10
SLAM3R[6]	19.11	13.44	20.20
MASt3R-SLAM[8]	15.71	12.60	18.60
VGGT-SLAM[7]	14.10	12.58	18.20
Multi-Agent Methods			
MAGiC-SLAM[10]	13.80	11.90	17.80
CP-SLAM[5]	14.60	12.10	18.20
Ours (TopoMA)	12.19	11.11	16.80

Replica [9]. These additional metrics measure how well the predicted point clouds cover the ground-truth surfaces, providing a stricter assessment of multi-agent fusion quality.

ScanNet. Table 4 extends the ScanNet results with Completeness. TopoMA consistently achieves the lowest or near-lowest errors across all three metrics, confirming that the improvements reported in the main paper are not limited to a single reconstruction criterion.

Replica. Table 5 presents extended reconstruction metrics on Replica [9]. In addition to achieving the best Acc, TopoMA also attains the lowest Completeness among multi-agent methods, indicating that topology-guided fusion improves both local surface detail and global coverage.

1.3. Multi-Agent Baseline Comparison

Although the main paper already compares TopoMA with single-agent methods under a simulated multi-agent pro-

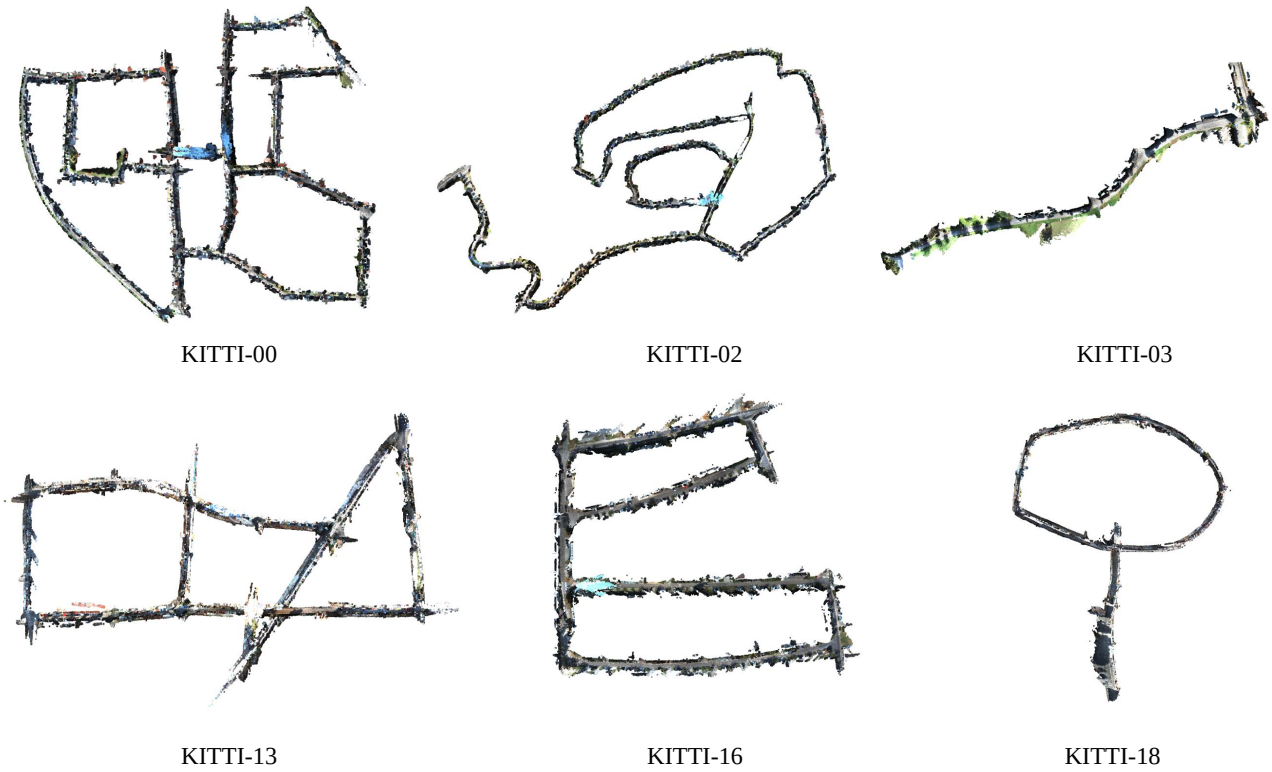

Figure 3. Reconstruction results on the KITTI dataset [4].

Table 5. Extended reconstruction metrics on Replica [9]. We report average Accuracy (Acc) and Completeness (Comp) across all Replica scenes. Lower is better.

Method	Acc [cm] ↓	Comp [cm] ↓
Single-Agent Methods		
VGGT-Long[3]	8.21	10.60
TTT3R[1]	9.11	11.20
SLAM3R[6]	9.00	11.00
MASt3R-SLAM[8]	8.29	10.80
VGGT-SLAM[7]	6.85	9.40
Multi-Agent Methods		
MAGiC-SLAM[10]	8.72	10.00
CP-SLAM[5]	10.22	11.80
Ours (TopoMA)	6.81	9.10

tocol, it is also important to directly contrast with recent multi-agent SLAM systems. Here we report explicit multi-agent comparisons against MAGiC-SLAM [10] and CP-SLAM [5] on KITTI [4] and ScanNet [2].

Table 6. Multi-agent tracking performance on KITTI [4]. We report average RMSE and Mean ATE [m] over all evaluated sequences under the same multi-agent splitting protocol.

Method	Avg. RMSE [m] ↓	Avg. Mean [m] ↓
MAGiC-SLAM[10]	23.87	19.11
CP-SLAM[5]	24.35	19.64
Ours (TopoMA)	22.51	18.32

KITTI multi-agent tracking. Table 6 compares average RMSE and Mean ATE on KITTI [4] under the same multi-agent splitting and evaluation protocol. TopoMA achieves slightly lower errors than MAGiC-SLAM [10] and CP-SLAM [5], confirming that topology-aware residual transport benefits large-scale outdoor tracking.

ScanNet multi-agent reconstruction. Finally, Table 7 reports multi-agent reconstruction performance on ScanNet [2]. TopoMA yields lower Depth L1 and Acc than both MAGiC-SLAM [10] and CP-SLAM [5], demonstrating that our topology-first design scales favorably to cluttered indoor environments.

Table 7. Multi-agent reconstruction performance on ScanNet [2]. We report average Depth L1 and Accuracy (Acc) [cm] over all sequences under the multi-agent protocol.

Method	Depth L1 [cm] ↓	Acc [cm] ↓
MAGiC-SLAM[10]	14.10	11.90
CP-SLAM[5]	14.60	12.10
Ours (TopoMA)	12.19	11.11

1.4. Qualitative Results on Additional KITTI Sequences

To demonstrate the effectiveness of our method, we further evaluate it on additional KITTI [4] sequences, and the reconstruction results are shown in Fig. 3.

Overall, these supplementary results substantiate our main claims: TopoMA not only improves trajectory and reconstruction accuracy over state-of-the-art single-agent and multi-agent baselines, but also maintains competitive runtime and memory usage, making it a practical solution for large-scale, server-free multi-agent 3D reconstruction.

2. Experimental Details

We summarize the hardware and runtime setting, topology and loop-closure hyper-parameters, loss weights, and the network architecture and training schedule used in all experiments.

2.1. Hardware and Runtime Settings

All experiments are run on an NVIDIA Jetson AGX Orin 16 GB platform with an integrated Ampere GPU (2048 CUDA cores and 64 Tensor Cores), a 12-core Arm Cortex-A78AE v8.2 64-bit CPU, and 16 GB LPDDR5 unified memory. Under the default 4-agent setting, **TopoMA** uses about 6.3 GB of unified memory, while the total system usage (including OS and other processes) typically remains in the 8.2–10.5 GB range. The full pipeline (front-end tracking, topology-guided mapping, decentralized loop closure, and residual transport) runs at roughly 6 Hz on typical sequences and 8–10 Hz on shorter indoor sequences; all FPS values refer to the end-to-end system.

2.2. Topology and Loop-closure Hyper-parameters

For completeness, we reproduce the main equations from the paper that are referenced here, and then specify the numerical hyper-parameters used in all experiments.

Eq. (4): topology similarity and fusion rule. The topological similarity between two point clouds $\mathbf{P}_{m,t}$ and $\mathbf{P}_{n,s}$ is defined as

$$S_{mn} = \psi_{\text{topo}}(\mathbf{P}_{m,t}, \mathbf{P}_{n,s}), \tag{4}$$

where $\psi_{\text{topo}}(\cdot,\cdot)$ is a persistent-homology-based similarity function. During global optimization, two views are fused only when S_{mn} exceeds a threshold τ.

Eq. (6): topology-regularized attention. Given the global token memory $\mathcal{Z}$, the query, key, and value matrices are

$$Q = W_Q\mathcal{Z}, \quad K = W_K\mathcal{Z}, \quad V = W_V\mathcal{Z}, \tag{1}$$

and the topology-regularized attention weight between tokens i and j is

$$\alpha_{ij} = \frac{\exp\Big(\frac{Q_i^\top K_j}{\sqrt{d}} - \lambda\, d_{ij}^{\text{topo}}\Big)}{\sum_{j'}\exp\Big(\frac{Q_i^\top K_{j'}}{\sqrt{d}} - \lambda\, d_{ij'}^{\text{topo}}\Big)}, \tag{6}$$

where d_{ij}^{topo} is the topological distance between tokens i and j, and λ controls the strength of topology-aware regularization.

Eq. (12): topology-gated loop closure. For a candidate loop between views (m,t) and (n,s), we first compute a loop score

$$s_{(m,t),(n,s)} = \sigma\big(f_{\text{loop}}(\mathbf{F}_{m,t}, \mathbf{F}_{n,s})\big), \tag{2}$$

where f_{loop} is a small MLP on concatenated map tokens and $\sigma(\cdot)$ is a sigmoid. Let $d^{\text{topo}}_{(m,t),(n,s)}$ be the geodesic distance on the topology skeleton; a loop edge is accepted only if

$$s_{(m,t),(n,s)} \geq \tau_{\text{loop}} \quad \text{and} \quad d^{\text{topo}}_{(m,t),(n,s)} \leq \delta_{\text{topo}}. \tag{12}$$

Eq. (22): topology-guided residual transport. On the topology skeleton $\mathcal{T} = (\mathcal{V}, \mathcal{E}_T)$, each node $v = (m,t) \in \mathcal{V}$ has a node-level residual descriptor $\mathbf{u}_v$ obtained by aggregating incident edge residuals. Residuals are transported along the skeleton via

$$\tilde{\mathbf{u}}_v = \sum_{u\in\mathcal{N}(v)} \beta_{v,u}\, \mathbf{u}_u, \tag{22}$$

where $\mathcal{N}(v)$ denotes neighbors of v on $\mathcal{T}$, the weights satisfy $\sum_{u\in\mathcal{N}(v)}\beta_{v,u} = 1$, and

$$\beta_{v,u} = f_{\text{topo}}\big(d^{\text{topo}}_{v,u}\big) \tag{3}$$

is a decreasing function of the geodesic distance $d^{\text{topo}}_{v,u}$.

We now specify the concrete hyper-parameters used in these equations.

We follow the notation in the main paper (e.g., Eq. (4), Eq. (6), Eq. (12), Eq. (22)). In the global topology optimization stage, each agent performs a mandatory topology fusion every 100 keyframes, even if the similarity score is

below the threshold, to avoid long periods without global alignment and to reduce accumulated drift. The topology similarity $S_{mn} \in [0, 1]$ between agents m and n is used in Eq. (4) with

$$S_{mn} > \tau, \quad \tau = 0.6, \tag{4}$$

which balances robustness to noisy matches and sufficient cross-agent fusion. In Eq. (6), we add a topology-aware bias to the attention logits by subtracting a penalty proportional to the geodesic distance d_{topo} on the skeleton,

$$\lambda\, d_{\text{topo}}, \quad \lambda = 0.5, \tag{5}$$

discouraging long-range attention across distant topological regions. Loop candidates in Eq. (12) are gated jointly by appearance and topology: we set

$$\tau_{\text{loop}} = 0.7, \tag{6}$$

$$\delta_{\text{topo}} = 5, \tag{7}$$

so a candidate is accepted only if the appearance score $s \geq \tau_{\text{loop}}$ and the geodesic distance on the topology skeleton does not exceed 5 hops; for each query frame we first keep the top-$K = 20$ candidates by s and then apply this gate. For residual transport in Eq. (22), we use a topology-aware weighting function

$$f_{\text{topo}}(d_{\text{topo}}) = \exp(-d_{\text{topo}}/\sigma), \quad \sigma = 2.0, \tag{8}$$

so that propagated residuals decay exponentially with the geodesic distance.

2.3. Loss Functions and Weights

For completeness, we explicitly restate the loss formulations from the main paper that are referred to as Eq. (17), Eq. (23), and Eq. (24).

Eq. (17): multi-modal pose refinement energy. Given loop and tree edges $e \in \mathcal{E}_T \cup \mathcal{E}_{\text{loop}}$ on the topology skeleton and matched samples $j \in \Omega_e$, we define depth, color, pointmap, and topology residuals

$$\begin{aligned} r_{e,j}^{\text{depth}} &= D_e - D_j, \\ r_{e,j}^{\text{color}} &= C_e - C_j, \\ r_{e,j}^{\text{pointmap}} &= P_e - P_j, \\ r_{e,j}^{\text{topo}} &= T_e - T_j, \end{aligned} \tag{9}$$

and the topology-regularized global refinement objective is

$$\begin{aligned} E_{\text{pose}} = \sum_{e \in \mathcal{E}_T \cup \mathcal{E}_{\text{loop}}} \sum_{j \in \Omega_e} \Big(& \lambda_{\text{depth}} \big\| r_{e,j}^{\text{depth}} \big\|_2^2 + \lambda_{\text{color}} \big\| r_{e,j}^{\text{color}} \big\|_2^2 \\ & + \lambda_{\text{pointmap}} \big\| r_{e,j}^{\text{pointmap}} \big\|_2^2 + \lambda_{\text{topo}} \big\| r_{e,j}^{\text{topo}} \big\|_2^2 \Big). \end{aligned} \tag{17}$$

Eq. (23): residual transport loss. Let $\mathbf{u}_v$ be the node-level residual descriptor at node $v = (m, t)$ and $\tilde{\mathbf{u}}_v$ its transported counterpart from Eq. (22). The residual-transport loss is

$$E_{\text{trans}} = \sum_{v=(m,t) \in \mathcal{V}} \big\| \tilde{\mathbf{u}}_v - \mathbf{u}_v \big\|_2^2 + \mu \big\| h_\theta(\tilde{z}_{k(m,t)}) - g_\theta(\tilde{\mathbf{u}}_v) \big\|_2^2, \tag{23}$$

where $h_\theta(\cdot)$ maps global tokens to pose-space corrections, $g_\theta(\cdot)$ maps transported residuals to the same pose space, and μ balances the two terms.

Eq. (24): total back-end objective. The total loss used to train the back-end is

$$E_{\text{total}} = E_{\text{pose}} + \lambda_{\text{trans}} E_{\text{trans}}, \tag{24}$$

where λ_{trans} controls the strength of residual transport regularization.

In the supplementary experiments we instantiate these equations with explicit scalar weights.

Eq. (17), Eq. (23), and Eq. (24) are instantiated with explicit scalar weights. The multi-modal pose refinement energy can be written as

$$E_{\text{pose}} = \lambda_{\text{depth}} E_{\text{depth}} + \lambda_{\text{color}} E_{\text{color}} + \lambda_{\text{pointmap}} E_{\text{pointmap}} + \lambda_{\text{topo}} E_{\text{topo}}, \tag{10}$$

with

$$\lambda_{\text{depth}} = 1.0, \lambda_{\text{color}} = 0.1, \tag{11}$$

$$\lambda_{\text{pointmap}} = 0.5, \lambda_{\text{topo}} = 0.5. \tag{12}$$

The residual transport loss E_{trans} in Eq. (23) uses an internal balancing factor

$$\mu = 1.0, \tag{13}$$

and the total loss in Eq. (24) is

$$E_{\text{total}} = E_{\text{pose}} + \lambda_{\text{trans}} E_{\text{trans}}, \tag{14}$$

with

$$\lambda_{\text{trans}} = 0.1, \tag{15}$$

so residual transport acts as a regularizer instead of dominating the optimization.

2.4. Network Architecture and Training Schedule

We denote the front-end module by f_θ^{causal} and the back-end module by f_θ^{global}. The front-end f_θ^{causal} is a transformer encoder with 12 layers, hidden dimension 768, and 12-headed self-attention; each frame is represented by 256 map tokens obtained by flattening a 2D feature map and linearly projecting, and each agent maintains a KV-cache of the most recent 64 frames using a sliding window. The back-end f_θ^{global} operates on the sparse topology skeleton

Table 8. Topology ablation on apartment-00 of Replica [9]. We report ATE, FPS, and unified memory usage (GPU/CPU) on Jetson AGX Orin 16 GB; all values are averaged over 5 runs.

Methods	ATE[cm] ↓	FPS[Hz] ↑	GPU[GB] ↓	CPU[GB] ↓
NoTopo	17.86	**6.92**	**5.40**	**8.80**
Topo-Attn	14.63	6.75	5.55	9.05
Topo-Loop	13.21	6.58	5.70	9.30
Topo-Trans	12.47	6.39	5.80	9.60
Full-Topo (Ours)	**10.48**	6.23	5.90	9.93

Table 9. Multi-agent scaling on apartment-00 of Replica [9]. We report ATE, FPS, and unified memory usage when varying the number of agents while keeping other settings fixed.

Agents	ATE[cm] ↓	FPS[Hz] ↑	GPU[GB] ↓	CPU[GB] ↓
2	12.81	**7.10**	**5.00**	**8.50**
3	11.57	6.72	5.40	9.10
4	10.48	6.23	5.90	9.93
5	10.32	5.82	6.40	10.60
6	**10.25**	5.43	6.90	11.20

with 4 transformer layers of hidden dimension 512 and 8-headed attention restricted to skeleton neighbors, and performs 3 unrolled gradient-descent steps per global block for topology-aware refinement. We train the full model with AdamW using an initial learning rate of 1×10^{-4}, weight decay 0.05, and 300k iterations on a mix of indoor and outdoor sequences; the learning rate follows a cosine schedule, remaining constant for the first 80% of iterations and decaying to 1×10^{-5} over the final 20%. Each optimization step samples 4 agents and 8 frames per agent (32 frames in total) with overlapping temporal windows to encourage cross-agent interactions. Data augmentation includes color jitter, random horizontal flipping, and random cropping to 640×384, and we avoid strong geometric warps that would distort the scene topology.

All experiments in the main paper use these settings unless otherwise noted.

2.5. Additional Topology Ablations and Multi-agent Scaling

Topology ablation on apartment-00. We further decompose TopoMA into three topology components on apartment-00 of Replica: topology-aware attention (Topo-Attn), topology-gated loop closure (Topo-Loop), and topology-guided residual transport (Topo-Trans). NoTopo disables all three (pure geometry), Topo-Attn enables only the attention bias, Topo-Loop enables only topology-gated loops, Topo-Trans enables only residual transport, and Full-Topo corresponds to the full model. As shown in Table 8, removing topology (NoTopo) yields the highest FPS and lowest memory but significantly degrades ATE. Adding only Topo-Attn already improves ATE with almost no cost. Topo-Loop further reduces spurious long-range matches and improves accuracy while staying real-time. Topo-Trans strengthens large-scale consistency by propagating residuals along the skeleton, with a small increase in GPU/CPU usage. Full-Topo achieves the best ATE with FPS and memory still well within the 16 GB budget of Jetson AGX Orin, showing that topology brings clear accuracy gains for modest overhead.

FPS–memory trade-off. Comparing the topology variants in Table 8, we observe a clear accuracy–efficiency trade-off. Disabling all topology cues (NoTopo) yields the highest FPS and lowest unified memory usage, but results in the worst ATE and poor global consistency. Gradually enabling topology-aware attention (Topo-Attn), topology-gated loop closure (Topo-Loop), and topology-guided residual transport (Topo-Trans) progressively improves ATE with only modest increases in GPU/CPU usage. The full **TopoMA** configuration (Full-Topo) achieves the best trajectory accuracy on apartment-00 while remaining real-time on Jetson AGX Orin and well within the 16 GB memory budget, demonstrating that topology brings substantial accuracy gains for a modest computational overhead.

Scaling to 2, 3, 5, and 6 agents. We also vary the number of agents from 2 to 6 on apartment-00 under the same simulated multi-agent protocol as in the main paper. As shown in Table 9, increasing agents from 2 to 4 consistently improves ATE thanks to richer viewpoints and denser topology, with gradually higher GPU/CPU usage and slightly lower FPS. Beyond 4 agents, additional accuracy gains are small, while resource usage continues to grow. For apartment-scale scenes, 3–4 agents offer a good balance between quality and efficiency; nonetheless, TopoMA can still handle 5–6 agents on Jetson AGX Orin without exceeding memory, at the cost of modest FPS reductions.

Multi-agent sequence splitting protocol. For fairness and reproducibility in Table 9, we keep the same simulated multi-agent protocol and vary only the number of agents $M \in \{2, 3, 4, 5, 6\}$. Given a monocular sequence with T frames indexed by $0, \ldots, T-1$, we build M temporally ordered sub-sequences by assigning each agent a contiguous interval with controlled overlap. We set the base segment length $L_{\text{seg}} = \lfloor T/M \rfloor$ and overlap ratio $\gamma = 0.3$, and define the stride $S = \lfloor L_{\text{seg}}(1-\gamma) \rfloor$. The k-th agent ($k = 0, \ldots, M-1$) receives frames

$$I_k = \left[kS,\ kS + L_{\text{seg}} - 1 \right] \cap [0, T-1], \qquad (16)$$

so consecutive agents share about γL_{seg} frames, while non-adjacent agents may overlap indirectly when M is larger. For $M = 2$ or 3, this yields long, heavily overlapping trajectories that emulate a few robots starting from different times; for $M = 4$ (our default), coverage and runtime are well balanced; for $M = 5$ and 6, each agent covers a

shorter segment but joint coverage becomes denser, leading to slightly stronger topological constraints at the cost of higher memory and lower FPS, consistent with Table 9.

Distributed execution and residual transport. Although all agents are simulated on a single Jetson AGX Orin in our experiments, we implement the system strictly in a distributed, server-free fashion to match the design in the main paper. Each agent a_m maintains its own front-end state (KV-cache $\mathbf{C}_m$, local poses $T_{m,t}$, and point clouds $\mathbf{P}_{m,t}$) and executes f_θ^{causal} independently on its incoming frames. The back-end f_θ^{global} is shared across agents but only sees a sparse view of the global state: instead of accessing all raw frames or feature maps, it operates on the topology skeleton $\mathcal{T} = (\mathcal{V}, \mathcal{E}_T)$ and on compressed node/edge descriptors derived from residuals.

Concretely, for each edge $e = ((m,t),(n,s)) \in \mathcal{E}_T \cup \mathcal{E}_{\text{loop}}$, the two incident agents first compute local multimodal residuals (depth, color, pointmap, and topology) and aggregate them into a compact edge descriptor $\mathbf{r}_e$. These $\mathbf{r}_e$ are the only quantities exchanged across agents. At each global update step, every node $v = (m,t)$ builds a node-level descriptor

$$\mathbf{u}_v = g_{\text{node}}(\{\mathbf{r}_e \mid e \in \mathcal{E}_T \cup \mathcal{E}_{\text{loop}},\ e \ni v\}), \tag{17}$$

and we then apply the residual-transport rule in Eq. (22),

$$\tilde{\mathbf{u}}_v = \sum_{u \in \mathcal{N}(v)} \beta_{v,u}\, \mathbf{u}_u, \tag{18}$$

where $\mathcal{N}(v)$ denotes skeleton neighbors of v, and $\beta_{v,u}$ is obtained from the topology-aware kernel $f_{\text{topo}}(d_{v,u}^{\text{topo}})$.

In practice, we root $\mathcal{T}$ at a designated anchor agent a_{ref} and transport residuals along the tree towards this anchor, so that all global information is concentrated in a single agent. This anchor is not fixed by the method: in our experiments we choose one agent as a_{ref} for convenience (e.g., the first agent or the one with better compute or more central coverage), but in principle any agent can be assigned as the anchor depending on task requirements or deployment constraints. In practice, the anchor role can also be reassigned at run time (e.g., when a robot enters a new sub-region or when compute resources change) without modifying the core topology or transport formulation. Non-anchor agents only maintain their local descriptors $\mathbf{u}_v$ and a small number of transported neighbors $\tilde{\mathbf{u}}_v$, whereas a_{ref} receives the full transported summary, keeping per-agent memory and bandwidth low.

The back-end then uses the transported descriptors $\tilde{\mathbf{u}}_v$ in Eq. (23) as an additional supervision signal for pose updates. For node $v = (m,t)$ belonging to agent a_m, we evaluate

$$E_{\text{trans},v} = \left\|\tilde{\mathbf{u}}_v - \mathbf{u}_v\right\|_2^2 + \mu \left\|h_\theta(\tilde{z}_{k(m,t)}) - g_\theta(\tilde{\mathbf{u}}_v)\right\|_2^2, \tag{19}$$

so that the pose increment predicted from global tokens $h_\theta(\tilde{z}_{k(m,t)})$ is encouraged to be consistent with the signal coming from residual transport $g_\theta(\tilde{\mathbf{u}}_v)$. Since only low-dimensional descriptors $\mathbf{r}_e$, $\mathbf{u}_v$, and $\tilde{\mathbf{u}}_v$ are communicated and the aggregation is concentrated at a_{ref}, the implementation closely matches the server-free, topology-guided distributed design of the main paper while keeping bandwidth and memory overhead modest.

References

[1] Xingyu Chen, Yue Chen, Yuliang Xiu, Andreas Geiger, and Anpei Chen. Ttt3r: 3d reconstruction as test-time training. *arXiv preprint arXiv:2509.26645*, 2025. 1, 2, 3

[2] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017. 1, 2, 3, 4

[3] 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. 1, 2, 3

[4] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012. 1, 3, 4

[5] Jiarui Hu, Mao Mao, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Cp-slam: Collaborative neural point-based slam system. *Advances in Neural Information Processing Systems*, 36:39429–39442, 2023. 1, 2, 3, 4

[6] 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 *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 16651–16662, 2025. 1, 2, 3

[7] Dominic Maggio, Hyungtae Lim, and Luca Carlone. Vggt-slam: Dense rgb slam optimized on the sl(4) manifold. 2025. 1, 2, 3

[8] Riku Murai, Eric Dexheimer, and Andrew J. Davison. Mast3r-slam: Real-time dense slam with 3d reconstruction priors. 2024. 1, 2, 3

[9] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, et al. The replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019. 1, 2, 3, 6

[10] Vladimir Yugay, Theo Gevers, and Martin R Oswald. Magicslam: Multi-agent gaussian globally consistent slam. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 6741–6750, 2025. 1, 2, 3, 4