TurboMap: GPU-Accelerated Local Mapping for Visual SLAM

Parsa Hosseininejad [1], Kimia Khabiri [1], Shishir Gopinath [1], Soudabeh Mohammadhashemi [1], Karthik Dantu [2], Steven Y. Ko [1]

[1]Simon Fraser University [2]University at Buffalo

Introduction

TurboMap is a GPU-accelerated and CPU-optimized **local mapping** backend for real-time **visual-inertial SLAM**, built on ORB-SLAM3 [1].

- Visual-inertial SLAM has three interdependent components: **tracking**, **local mapping**, and **loop closing**.
- Local mapping builds and refines the shared map.
- Slow local mapping suppresses keyframe creation and skips optimization, **degrading the map** and risking **tracking loss**.
- Prior work parallelizes bundle adjustment [2], but other stages remain serialized because map modification forces synchronization.

TurboMap accelerates local mapping **holistically**, tackling both synchronization-induced serialization and CPU-GPU transfers.

Local Mapping Pipeline

Local mapping processes each incoming keyframe through a sequence of stages that build, maintain, and optimize the map.

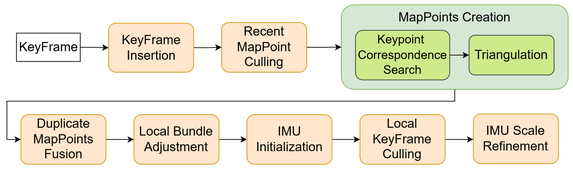


Figure 1. Local mapping workflow in ORB-SLAM3.

Where the Time Goes

Roughly 95% of local mapping time is spent in just four stages: Map Point Creation, Map Point Fusion, Local BA, and Keyframe Culling.

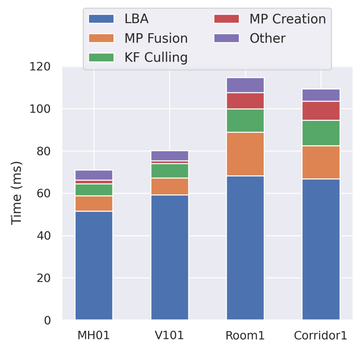


Figure 2. Local mapping breakdown across EuRoC and TUM-VI.

- **Map Point Creation** — finds keypoint correspondences across neighbouring keyframes, then triangulates matches into new 3D map points.
- **Map Point Fusion** — finds and merges duplicate map points across neighboring keyframes to keep the map sparse and consistent.
- **Local Bundle Adjustment (LBA)** — jointly refines local keyframe poses, associated inertial variables, and map point positions.
- **Keyframe Culling** — removes keyframes whose observations are already covered by others, keeping the map compact.

Barriers to Parallel Execution

1. **Shared-state updates serialize execution.**

- Local mapping modifies keyframes, map points, observations, and the covisibility graph.
- These are shared with **tracking** and **loop closing**, so every update needs synchronization via locks.
- Search and map modification are **interleaved** in the original code, creating serialization points throughout.

2. **Map data is too large to transfer to the GPU repeatedly.**

- A keyframe is roughly **1 MB**, holding thousands of descriptors. Each iteration may access tens of keyframes.
- Transferring keyframes each time would erase any gain from parallel execution.

TurboMap Architecture

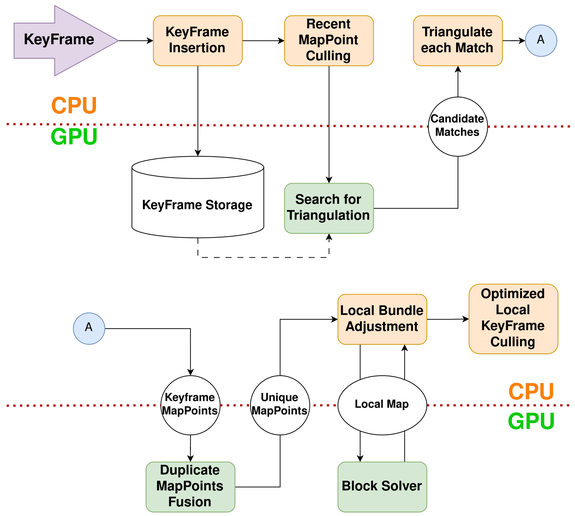


Figure 3. Local mapping dataflow in TurboMap.

TurboMap parallelizes key bottlenecks on the GPU. We parallelize a stage only when it is expensive, its synchronization can be removed, and it avoids repeated heavy transfers.

- **GPU keyframe storage.** Each keyframe is copied to the GPU **once** on insertion and stays resident. Only **immutable** parts (keypoints and descriptors) are stored, avoiding excessive synchronization.
- **Keypoint Correspondence Search on the GPU.** Search is decoupled from triangulation, so all matches are found in parallel before any map point is created. New map points are then created on the CPU.
- **Map Point Fusion on the GPU.** Similar to keypoint correspondence search, all duplicate candidates are first identified in parallel on the GPU, then merged sequentially on the CPU.
- **Keyframe Culling optimized on the CPU.** A per-map-point counter array replaces the nested scan over keyframe observations.
- **Local bundle adjustment accelerated by the GPU.** We use a GPU block solver [2] which computes the Schur complement, the major bottleneck, more quickly.

Evaluation Setup

We evaluate TurboMap and ORB-SLAM3 in stereo-inertial mode using EuRoC [3] and TUM-VI [4] SLAM datasets, averaging over 5 runs.

Table 1. Evaluation platforms.

Machine	Specifications
Desktop	20-core Intel Core i7-12700K CPU @ 5.0 GHz 10496-core NVIDIA RTX 3090 GPU 64 GB RAM
Jetson Xavier NX	6-core ARM Carmel CPU @ 1.4 GHz 384-core NVIDIA Volta GPU 8 GB RAM

Runtime Performance and Trajectory Error

Table 2. Average local mapping performance, reported as ORB-SLAM3 → TurboMap. Latency decreases while ATE remains consistent.

Platform	Dataset	Time (ms)	Component Speedup				ATE (m)
			KP	Fusion	LBA	Cull	
Desktop	EuRoC	65.7 → 51.6	2.2×	1.5×	1.2×	1.8×	0.036 → 0.034
	TUM-VI	109.8 → 69.8	3.3×	2.3×	1.4×	3.3×	0.012 → 0.013
Jetson	EuRoC	372.7 → 287.9	3.5×	1.4×	1.3×	1.8×	0.034 → 0.031
	TUM-VI	551.6 → 356.9	5.2×	2.2×	1.3×	3.2×	0.013 → 0.013

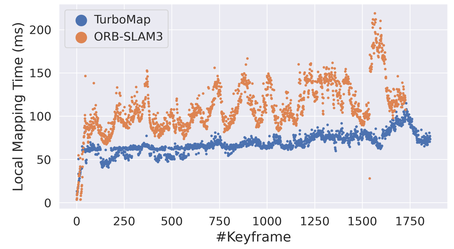


Figure 4. TurboMap reduces the local mapping keyframe processing time and its variance (corridor1, desktop).

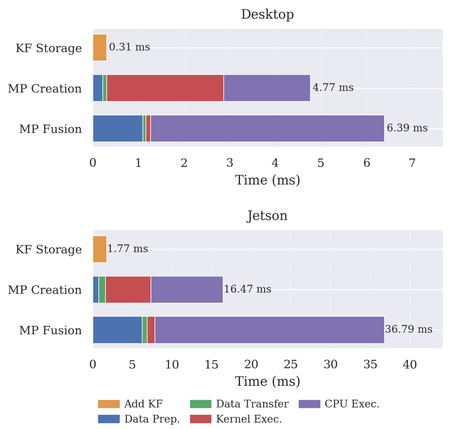


Figure 5. Timing breakdown of Map Point Creation and Fusion stages, with keyframe storage on corridor1. For the GPU part, Map Point Creation is kernel-bound while data preparation dominates Fusion.

Stress Test

We process sequences faster than real-time to force a high keyframe insertion rate.

Table 3. Comparison with high keyframe rate (avg. 5 runs, desktop). TurboMap shows less degradation since LBA and KF Culling skip less.

Sequence	TurboMap		ORB-SLAM3	
	ATE	#Skipped LBA & KF Culling	ATE	#Skipped LBA & KF Culling
MH01	0.040	3.4	1.105	58
V101	0.039	18.8	0.091	102.6
room1	0.011	5.8	1.005	140.4
corridor1	0.045	9.2	10.136	343.8

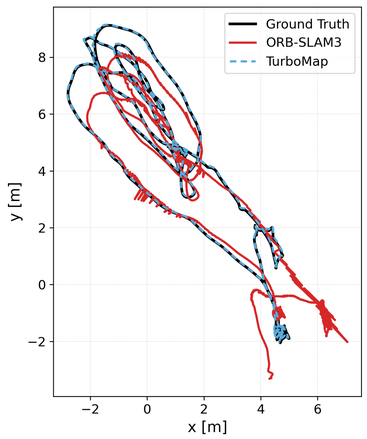


Figure 6. MH01 trajectory comparison under a high keyframe rate.

Summary

TurboMap achieves faster local mapping.

- TurboMap is **1.3**× faster on EuRoC and **1.6**× faster on TUM-VI (avg. both platforms) with **no accuracy loss**.
- Under heavy load, ORB-SLAM3's trajectory error reaches **10 m** while TurboMap stays below **5 cm**.

`https://github.com/sfu-rsl/TurboMap`

References

[1] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[2] S. Gopinath, K. Dantu, and S. Y. Ko, "Improving the performance of local bundle adjustment for visual-inertial slam with efficient use of gpu resources," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 6239–6245.

[3] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," *Int. J. Rob. Res.*, vol. 35, no. 10, p. 1157–1163, Sep. 2016. [Online]. Available: https://doi.org/10.1177/0278364915620033

[4] D. Schubert, T. Goll, N. Demmel, V. Usenko, J. Stueckler, and D. Cremers, "The TUM VI Benchmark for Evaluating Visual-Inertial Odometry," in *IROS*, October 2018.