MambaVO: Deep Visual Odometry Based on Sequential Matching Refinement and Training Smoothing

Appendix

A. Implementation Details

A.1. Geometric Initialization Module

The images are resized to 448x224. Metric3D only runs once per keyframe. We use pretrained EfficientLoFTR [59] with mixed precision in the Geometric Initialization Module (GIM) to obtain initial matches. EfficientLoFTR performs both coarse and fine-level matching. The fine-level stage provides pixel-level correspondences along with their features, which serve as our initial matches and geometric features. Since semi-dense matching can produce numerous matching pixels per frame, we randomly select 100 matches per frame for subsequent pose estimation.

We employ the lightweight small version of Dino-v2 [39] to generate generalized and robust semantic features. Specifically, we adopt the small version with registers as the pretrained weights of Dino-v2 model. For each input frame, Dino-v2 produces patch-level features with 384 channels. These patch features are reorganized and deconvolved to the pixel level to obtain semantic features for each pixel. Using the initial matching coordinates, the corresponding semantic features are queried and fused with geometric features to form the matching feature (Sec. 3.1). The resulting matching feature is then fed into the Geometric Mamba Module.

A.2. Geometric Mamba Module

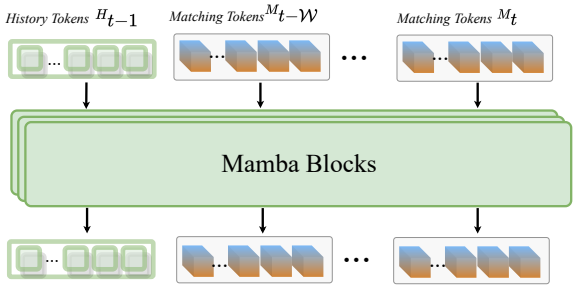


Figure 7. Illustration of Mamba blocks in Geometric Mamba Module.

In the Geometric Mamba Module, we set the number of Mamba blocks B to 3. The input to Mamba at frame t consists of the history tokens from frame $t-1$ and all matching tokens of frames $t-\mathcal{W}$ to t, in chronological order. Mamba blocks output updated history tokens and matching tokens.

This design of input leverages Mamba's long-distance modeling capabilities, utilizing historical interactions to adjust matching tokens and estimate the refinement and weight of each matching pixel.

B. Qualitative Results

B.1. Matching Comparison

The accuracy of visual odometry heavily depends on computing visual correspondences between frames. In MambaVO, we first use semi-dense matching for initial correspondences, then refine them using the Mamba architecture with temporal information. Experimental results demonstrate that our matching approach outperforms previous visual odometry methods in terms of pose accuracy between frames (Sec. 5.2).

To intuitively demonstrate the matching quality of MambaVO, we visualize the matching pixels of MambaVO and the previous SOTA method DPVO [51] in Fig. 8. DPVO initializes 96 patches for matching per frame. During visualization, we remove invalid matches (those beyond the image coordinate range) and mark only the center points for clarity.

Compared to DPVO, which often produces ambiguous, inaccurate matches, MambaVO delivers more accurate, consistent, and evenly distributed matching points on the image plane, benefiting subsequent pose estimation.

B.2. Trajectory and Map Visualization

We visualize the results of MambaVO++ on EuRoC [1], TartanAir [57], KITTI [15] and TUM-RGBD [46] and the trajectories are illustrated in Fig. 9 and Fig. 10 respectively. The sparse maps are visualized in Fig. 11.

Among the four datasets, TartanAir is a simulation dataset, while the other three are collected from real-world environments. TartanAir covers both indoor and outdoor scenes, with trajectories ranging from a few meters to several tens of meters. The EuRoC dataset, collected by drones in indoor environments, shows that our method's results align closely with the ground truth in most sequences. TUM-RGBD is a challenging indoor dataset characterized by erratic camera motion and significant motion blur. As seen in the experimental results (Tab. 4) and trajectory plots (Fig. 10), our method remains effective under these demanding conditions. KITTI, an outdoor autonomous driving dataset, typically features large-scale, high-speed movements. To evaluate our method's performance on long-distance scenarios, we selected five sequences, including

both those with and without loop closures, as in Fig. 10.

C. Additional Experiments

C.1. Architecture Comparison

To further verify the rationality of the Mamba architecture, we conduct a comparative experiment between the Mamba and Transformer architectures. We replace Mamba with Transformer and the result is shown Tab. 8. Mamba is, light-weight, computationally efficient, and produces more accurate results.

Dataset	EuRoC	
Metirc	ATE↓	FPS↑
TransfomerVO	0.112	17Hz
MambaVO	**0.094**	**22Hz**

Table 8. Comparative experiment between Mamba architecture and Transformer architecture.

C.2. Time Analysis

We have made a lot of efforts to try to improve the efficiency of our system (Appendix A). Following other SLAM methods (TartanVO, DROID-SLAM, DPVO), we skip every other frame, as consecutive frames have similar images and poses. This nearly doubles the frame rate. We evaluate the average per-frame time (ms) on EuRoC (Tab. 9).

Module	E-LoFTR	Metric3D	DINO-v2	PnP	cross-attention	Mamba	BA
Avg. time†	14.2	18.3	28.8	11.6	3.1	12.0	11.7
Avg. time∗	7.1	3.5	14.4	5.8	1.55	6.0	5.85

Table 9. The average time for each part of MambaVO on EuRoC.

† indicates the running time it takes for the module to process one image. ∗ represents the total time of each module of a trajectory divided by the number of frames, which represents the actual average rates.

D. Limitations

Our proposed MambaVO can only obtain sparse maps because only semi-dense matching pixels in GIM are used to build the map. In the future, we will utilize dense information and 3D Gaussian Splatting [26] technology to obtain high-fidelity maps for rich scene representation while maintaining localization accuracy.

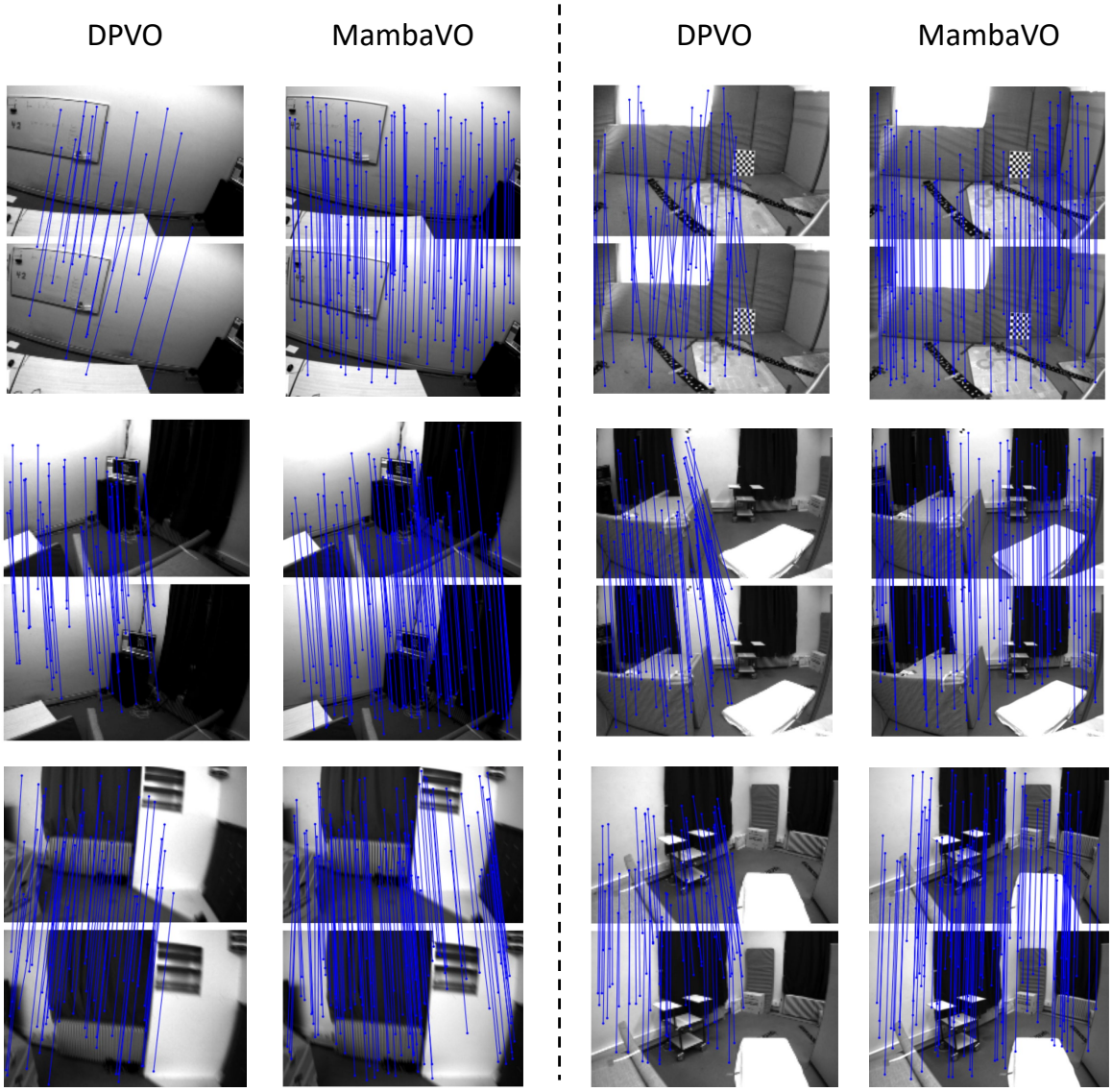


Figure 8. Matching results of two consecutive frames in DPVO and MambaVO. The matching quality of MambaVO, including accuracy and distribution, is better than DPVO [51], which is the previous SOTA deep visual odometry.

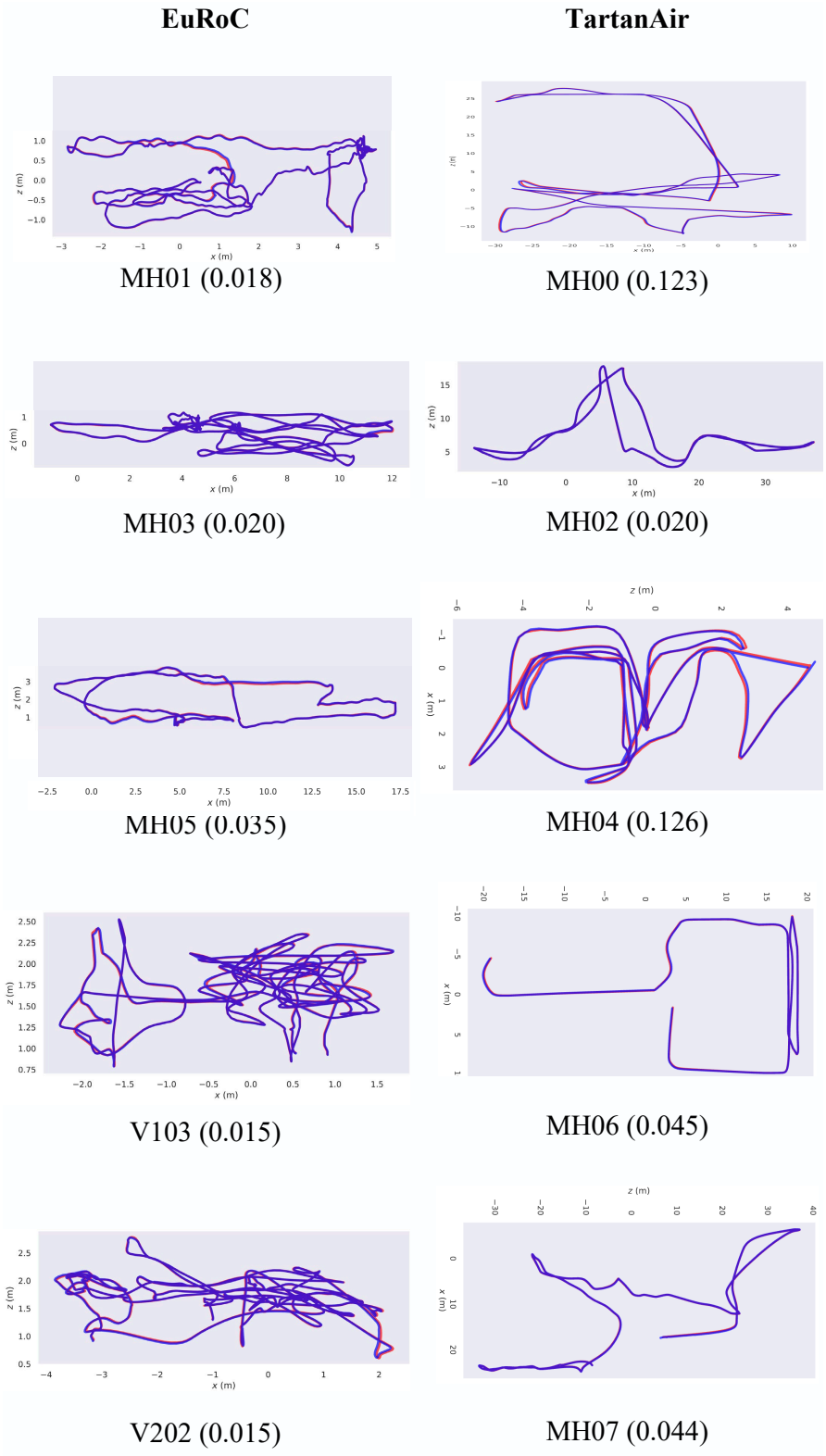


Figure 9. Qualitative visualization of MambaVO++ on EuRoC and TartanAir. The blue line represents the trajectory estimated by MambaVO++, and the red line represents the ground truth. The results show that our estimated trajectory almost completely coincides with the ground truth.

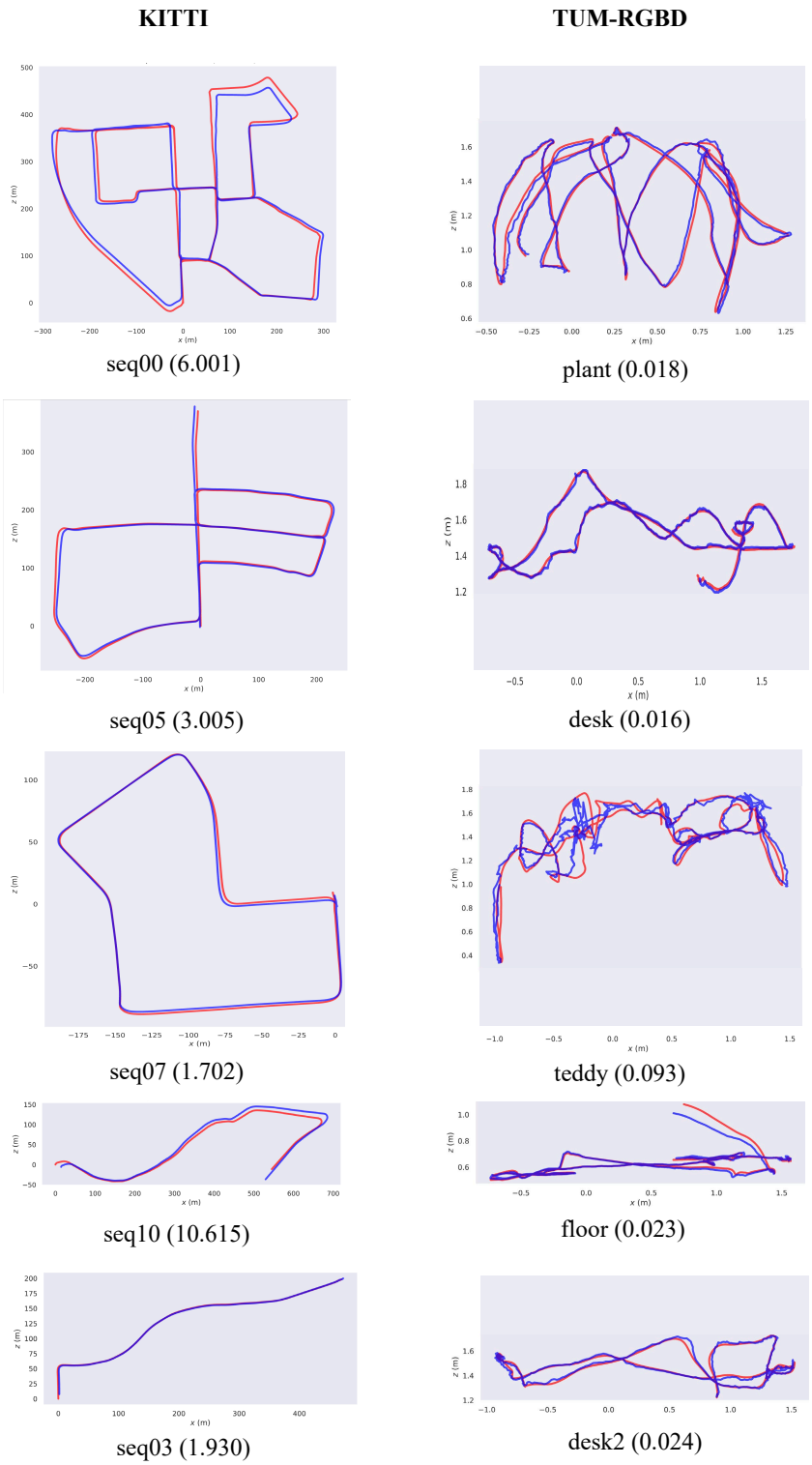
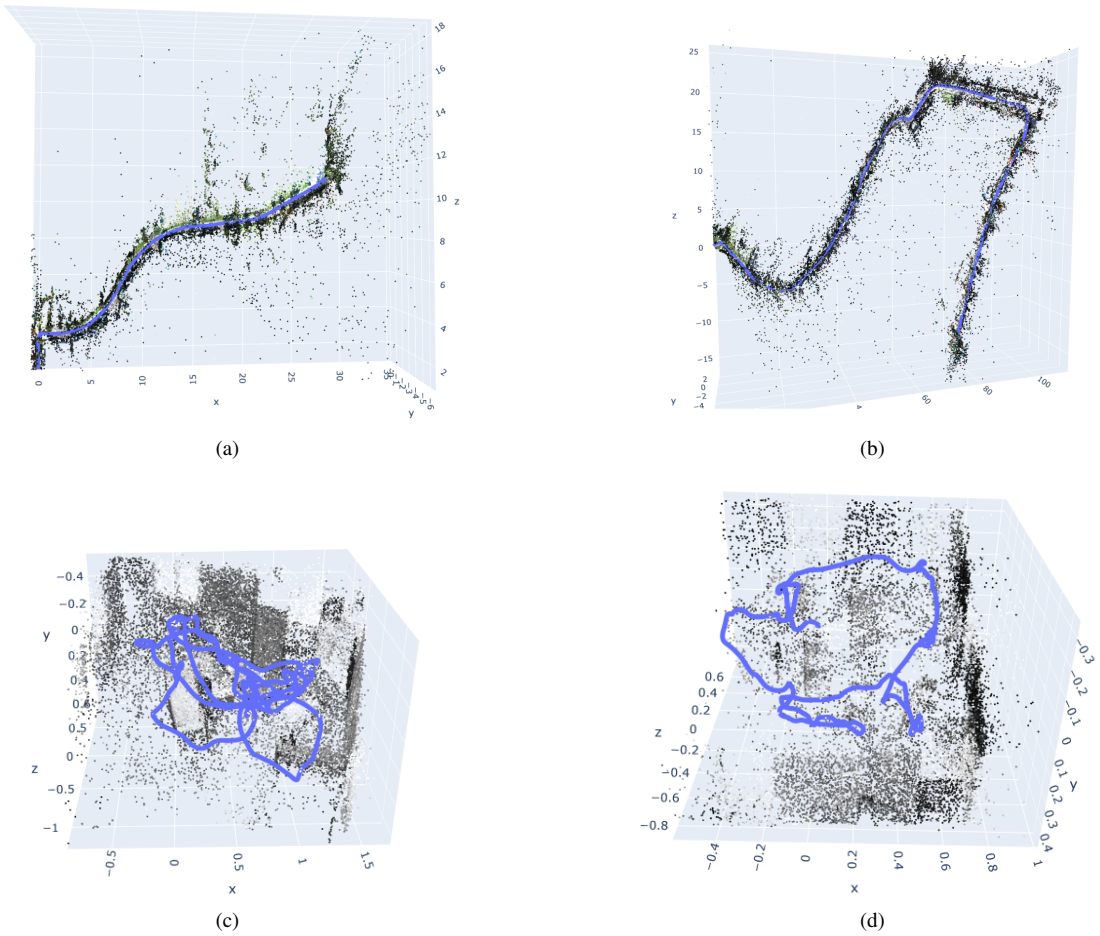


Figure 10. Qualitative visualization of MambaVO++ on KITTI and TUM-RGBD. The blue line represents the trajectory estimated by MambaVO++, and the red line represents the ground truth. The results show that our estimated trajectory almost completely coincides with the ground truth.



(a) (b) (c) (d)

Figure 11. The proposed MambaVO can produce trajectories and sparse maps. We visualize the trajectories and sparse maps of KITTI (a and b) and EuRoC (c and d) respectively. The blue line represents the trajectories and the points respresent the map points.