Revisiting Monocular SLAM with Spatio-Temporal Scene Modeling

SUPPLEMENTARY MATERIALS

Valter Piedade[1,2], Lalit Manam[1], Masashi Yamazaki[3], Pedro Miraldo[1,✉]

[1]Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA
[2]Instituto Superior Técnico, Lisbon [3]Mitsubishi Electric Corporation, Tokyo

*These supplementary materials present: (1) details on the Kullback–Leibler divergence score used for keyframe creation; (2) implementation details; (3) experiments demonstrating the modularity of the proposed Simultaneous Localization and Mapping (SLAM) framework (Mitsubishi Electric Research framework for visual SLAM (***SLAMMER***)); (4) results on the ETH3D-SLAM dataset and further performance analysis; (5) ablation studies; (6) an explanation for the omission of the* `pumpkin` *sequence from 7-Scenes.*

A. Note on using KL Divergence

For the third rule for keyframe creation (Sec. 3.6 in the main paper), we use the Kullback–Leibler (KL) divergence, denoted $D_{KL}(\mathbf{H}_k||\mathbf{H}_{k-1})$, to compare histograms ($\mathbf{H}_k$) of counts of 3D map-points, seen by each keyframe, between two consecutive frames. With the two histograms of consecutive frames, $\mathbf{H}_k$ and $\mathbf{H}_{k-1}$, we add a small value $\epsilon = 10^{-12}$ to each bin in the histogram (to avoid zero divisions), normalize each one so that all bin values add to 1, and compute $D_{KL}(\mathbf{H}_k||\mathbf{H}_{k-1})$ as follows

$$D_{KL}(\mathbf{H}_k||\mathbf{H}_{k-1}) = \sum_i \mathbf{H}_k(i) \cdot \log\left(\frac{\mathbf{H}_k(i)}{\mathbf{H}_{k-1}(i)}\right). \quad \text{(A.1)}$$

The higher the value of $D_{KL}(\mathbf{H}_k||\mathbf{H}_{k-1})$, the more different the histograms are. We use this score to make a keyframe creation decision by identifying when the current frame suddenly observes a previously mapped area. This typically happens when the camera completes a loop. Fig. 4 of the main paper exemplifies the histogram behaviour we look for, in our third rule for keyframe creation.

B. Implementation Details

This section provides implementation details on the **SLAMMER** pipeline, namely the settings used to obatin the experimental results reported in Sec. 4 of the main paper. The settings are as follows.

- **Keypoint extraction:** Maximum of 700 feature points using ALIKED [19];
- **Query map-points:** Map points are queried from the temporal query, whose buffer has a size of 100 frames, and from the spatial query. Each spatial query retieves a maximum of 200 cells;
- **3D-2D pose estimation:** Pose estimation is based on PoseLib [7] using default settings for Random Sample Consensus (RANSAC) and final refinement. The estimation is considered valid if it has a minimum of 40 inliers and 20% inlier ratio;
- **Keyframe creation:** To create a keyframe, we set (1) the minimum number of inliers from pose estimation to 100, (2) the point spread score threshold to 0.35 and (3) the KL divergence threshold to 1. For every validated keyframe, we run MASt3R to obtain depth and filter 3D points with a confidence lower than 0.25;
- **Loop closure and relocalization:** We use MegaLoc [1] as the Visual Place Recognition (VPR) approach to obtain

Table C.1. Performance comparison between two alternative methods for keypoints extraction: CudaSift and ALIKED. Results obtained for the 360 sequence of the TUM RGB-D dataset [16].

Method	Keypoint Extraction		Pose Estimation	
	# Keypoints	Time [ms]	% Inliers	Time [ms]
CudaSift [2]	418.2	2.02	54	1.84
ALIKED [19]	488.3	7.73	76	1.12

the top 20 loop closure candidates, which are validated with inlier 3D-2D coorespondences in pose estimation.

C. Pipeline Modularity

In Sec. 3 of the main paper, we described our pipeline. Here, we showcase its modularity by utilizing different off-the-shelf methods for (1) keypoint extraction, (2) feed-forward geometry estimation, and (3) loop detection, and also discuss the impact of their specific choices.

C.1. Keypoint Extraction

Image keypoint extraction is the first step in the localization module (Sec. 3.3 in the main paper) of our pipeline when a new frame is received. Several keypoint extractors are available, typically categorized as either hand-crafted (*e.g.*, SIFT[9], ORB [13]) or learning-based (*e.g.*, SuperPoint [5], ALIKED [19]) approaches. Our modular pipeline can incorporate any keypoint extraction method; however, the choice of extractor can affect the accuracy and efficiency of the SLAM system, particularly within the localization module.

Table **C.1** presents an evaluation of the localization module—specifically the keypoint extraction and pose estimation steps—using the ALIKED and CudaSift [2] keypoint extractors. The results report the average (1) number of extracted keypoints, (2) keypoint extraction time, (3) percentage of pose estimation inliers, and (4) pose estimation runtime for the 360 sequence of the TUM RGB-D dataset [16]. For the keypoint extraction step, we set the maximum number of keypoints to 500. We observe that ALIKED reaches close to this target, whereas CudaSift produces considerably fewer keypoints. ALIKED also leads to more pose estimation inliers, indicating that keypoints are tracked more consistently. A higher number of inliers not only speeds up pose estimation but also improves point tracking against the map, thereby enhancing localization and reducing the need to create new keyframes. Beyond quantitative results, we also observe more stable keypoint tracking across consecutive frames when using ALIKED compared to CudaSift. The only downside of ALIKED (implemented in libTorch C++) compared to CudaSift (implemented in CUDA C++) is computational time, which is roughly $3\times$ longer, but still allows real-time capabilities.

C.2. Feed-Forward Depth Inference

During keyframe creation in the localization module (Sec. 3.3 in the main paper), our pipeline takes input from a monocular camera and uses a feed-forward depth inference method to obtain a point cloud. Several approaches have been proposed recently; here, we evaluate MASt3R [8], DUSt3R [18], and VGGT [17]. For each method, we exported the original model—together with the weights provided by the authors—to ONNX and ran inference in C++ using ONNXRuntime [6]. Similar to the keypoint extraction step, our pipeline can incorporate any feed-forward depth inference method; however, this choice has a significant impact on the system. Figure **C.1** presents results obtained using the three methods on two sequences.

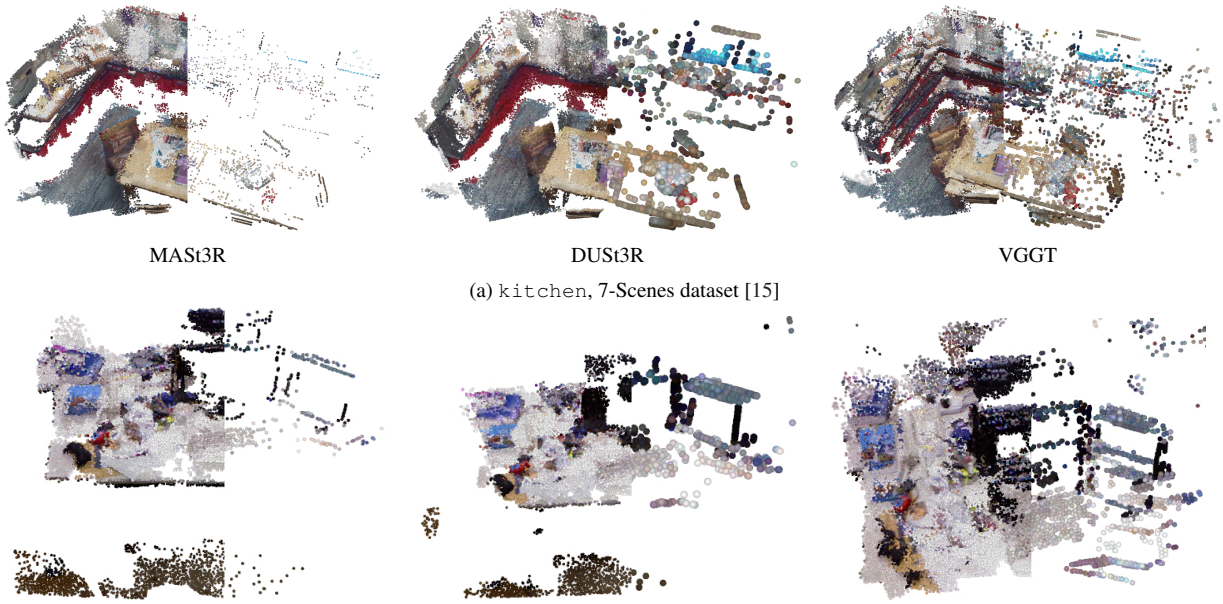
We observe that MASt3R provides the overall best reconstruction results. We note that we use the same pipeline parameters for all reported experiments. The local 3D points produced by different feed-forward depth inference methods are scale-free—*i.e.*, they are not metric—as discussed in Sec. 3.3 of the main paper. Because different methods produce 3D points at different scales, our parameters could, in principle, be tuned to better match each approach. In this work, we chose to fine-tune all parameters for MASt3R because it is less computationally expensive than VGGT (as detailed below). After tuning for MASt3R, we fixed all parameters for the remainder of the experiments, including those shown here with alternative depth inference models. Our goal in this paper is to demonstrate the benefits of various design choices in our pipeline for real-time operation with accuracy comparable to current state-of-the-art methods. Additional parameter tuning for other depth inference approaches would not change the overall conclusions of the paper.

Regarding efficiency, there are notable differences among the three methods. MASt3R requires 112 ms for inference, while DUSt3R requires 98 ms; both use two images. For VGGT, inference takes 209 ms with one image, 481 ms with two images, 780 ms with three images, and 990 ms with four images. For VGGT, we use up to four images for inference.

C.3. Loop Detection

Next, we demonstrate the interchangeability of the loop detection approach in our pipeline (Sec. 3.7 of the main paper). The loop detection module uses a VPR method to identify the keyframes most similar to the current one. The module receives either a keyframe image or the keypoints detected in that image and outputs a list of the K-most similar keyframes, which we refer to as candidate keyframes. The better the VPR method performs, the more accurate the candidate keyframes will be, allowing more loops to be closed and resulting in a more accurate map.

We integrated the state-of-the-art MegaLoc [1] and



(a) `kitchen`, 7-Scenes dataset [15]

(b) `xyz`, TUM RGB-D dataset [16]

Figure C.1. Showcasing the modularity of our pipeline for the feed-forward depth inference. Our pipeline only runs it to generate local 3D points for every keyframe. This figure shows our hybrid reconstruction results using MASt3R [8], DUSt3R [18], and VGGT [17] for the `kitchen` and `xyz` sequences. In each image, we show the semi-dense 3D reconstruction (left) and the sparse map points used for localization (right). While any approach can be used, MASt3R gives the best results among the three.

Table C.2. Demonstration of the modularity of **SLAM-MER** for the VPR approach using MegaLoc and BoW. We report the average time to create an image descriptor, the loop detection average time, and the Absolute Trajectory Error (ATE) and number of successfully closed loops (# Loops) for two sequences.

Method	Descriptor Creation [ms]	Detection Time [ms]	`360`		`room`	
			ATE	# Loops	ATE	# Loops
MegaLoc [1]	9	0.05	0.107	10	0.095	5
Bag-of-words [4]	2.5	0.10	0.097	7	0.115	3

the traditional Bag-of-Words (BoW) approach [4] into our pipeline. For MegaLoc, we exported the original model—together with the weights provided by the authors—to ONNX and ran inference in C++ using ONNXRuntime [6]. For BoW, we used the implementation available in OpenCV [3]. The BoW vocabulary was trained using the `360` sequence from the TUM RGB-D dataset [16]. Results from both approaches are reported in Table **C.2**.

Both BoW and MegaLoc require training. MegaLoc was trained on a wide range of datasets, making it generalizable across different scenes and ready to use off the shelf. BoW, on the other hand, requires a vocabulary to be built; this vocabulary is then used to create the image-level descriptor. We created the BoW vocabulary using the `360` sequence from TUM. We observe that BoW is faster than MegaLoc when creating the image-level descriptor. However, MegaLoc is faster when querying the database for similar descriptors and can close more loops overall. Since the loop closure module runs in parallel, efficiency becomes a concern only if one method is significantly slower than the other, which is not the case here. We use MegaLoc as the default, as it generalizes well across scenes and closes more loops. Additional VPR methods can be easily integrated into our pipeline.

C.4. Input Sensor

Another flexibility provided by our pipeline is the ability to use different types of vision sensors—either sensors that output depth directly, such as RGB-D cameras, or sensors that can infer depth indirectly, such as stereo RGB systems. Using sensors that provide depth significantly reduces the computational burden of our pipeline, since no depth estimation is required when a new keyframe is created. In this case, the depth is obtained simply by reading the sensor data.

Figure **D.2** shows results using our framework—without any modifications (including the parameters)—running visual SLAM with RGB-D sensors, directly using the raw depth information from the TUM RGB-D and 7-Scenes datasets (both of which provide RGB-D data). We highlight

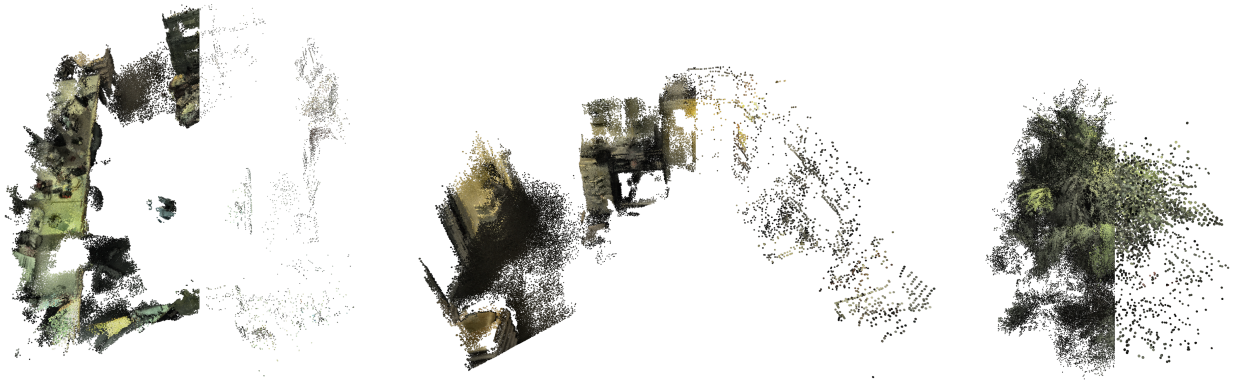


Figure C.3. Visualization of point clouds of ETH3D-SLAM [14] sequences, obtained with our pipeline. Each image shows the semi-dense 3D reconstruction (left) and the sparse 3D map-points used for localization (right).

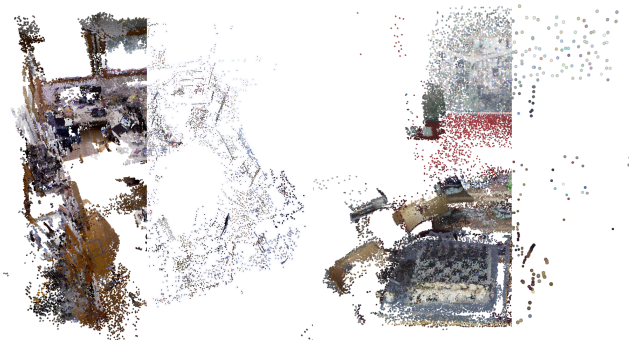


Figure D.2. Examples from the TUM and 7-Scenes sequences (`room` and `chess`), using RGB-D depth instead of depth predicted by a feed-forward model.

that our pipeline creates a set of local 3D points for each keyframe with the depth information provided by the sensor, and does not directly consider them as the map-points. This inherently allows some noise tolerance across different sensors, which is corrected by the adjustment module of our pipeline when we create new 3D map-points from these local 3D points. Our pipeline can work with any sensor that provides depth information along with an RGB image, without any change in the core modules, demonstrating the modularity across multiple sensors.

D. Additional Results

Sec. 4 of the main paper provides results with the 7-Scenes [15] and TUM RGB-D [16] datasets. This section complements it by providing (1) results for the ETH3D-SLAM [14] dataset in Sec. **D.1**, (2) a runtime breakdown of each module of the SLAM pipeline in Sec. **D.2**, and (3) a memory footprint analysis in Sec. **D.3**.

D.1. ETH3D-SLAM dataset

This subsection provides results on one additional dataset, ETH3D-SLAM [14]. We note that the ETH3D-SLAM dataset contains sequences that were not taken under ambient light conditions; all sequences were obtained in dark environments with minimal lighting and low texture, more suitable for RGB-D methods. Since our pipeline relies on keypoint extractors, which often fail to provide a good descriptor under these conditions, we perform the Contrast Limited Adaptive Histogram Equalization (CLAHE) [12] on the input images, as an additional preprocessing step only on this dataset to obtain reliable keypoint descriptors. We use OpenCV's implementation [3] for our experiments. Still, many sequences are very challenging for keypoint-based methods, with very aggressive changes in lighting conditions–the matching between descriptors from CudaSift and ALIKED fails, which means that the entire SLAM tends to fails. In these scenarios, a 3D-to-3D registration approach for camera localization, such as MASt3R-SLAM or VGGT-SLAM, is more appropriate, although it is significantly slower.

In Fig. **C.3**, we visualize point clouds (both sparse and dense) on sequences of the ETH3D-SLAM dataset. It can be seen that our pipeline provides good-quality point clouds even in these datasets with challenging lighting conditions.

D.2. Runtime Breakdown

Table **D.3** shows a runtime breakdown per module of the SLAM pipeline in two TUM RGB-D [16] sequences. The runtimes for Localization, Keyframe creation, and Covisibility graph are similar for both sequences. Localization does not include the runtime for creating keyframes, since it runs as a parallel process, and Localization does not wait for it to end. We only observe a change in runtime for Loop closure, which is expected since `room` is a larger sequence

Table D.3. Runtime breakdown per module of the SLAM pipeline in TUM RGB-D [16]. *w/o keyframe creation; **detecting and closing all loops per keyframe.

Module	Avg. Runtime [ms]	
	`desk`	`room`
Localization*	11.8	11.9
Keyframe creation	80.0	79.4
Covisibility graph	3.04	4.50
Loop closure**	14.1	20.9

Table D.4. Memory footprint comparison agaisnt MASt3R-SLAM [11] and VGGT-SLAM [10] in TUM RGB-D [16].

Map Components	MASt3R-SLAM		VGGT-SLAM		SLAM-MER [Ours]	
	`desk`	`room`	`desk`	`room`	`desk`	`room`
# Map-points	N/A	N/A	N/A	N/A	$8k$	$31k$
# Keyframes	12	53	57	197	16	74
# Cells	N/A	N/A	N/A	N/A	90	773

with a long range loop unlike `desk`.

D.3. Memory Footprint

Table **D.4** provides a memory footprint comparison betwen MASt3R-SLAM [11], VGGT-SLAM [10] and **SLAM-MER** in terms of the size of the map each approach builds. Since MASt3R-SLAM and VGGT-SLAM do not have the same concept of map-points (dense approaches) and do not create cells, the main comparison aspect is the number of created keyframes. **SLAM-MER** creates much fewer keyframes than VGGT-SLAM and slightly more than MASt3R-SLAM.

To manage memory footprint as map size increases, we (1) delete duplicated map points after merging observations in loop closure and (2) ignore empty cells for spatio-temporal querying of 3D points. We do not cull keyframes since they are created only when new map regions are visited. Moreover, culling is not required because our system avoids creating keyframes in previously mapped areas due to the spatio-temporal localization design and keyframe creation criteria.

E. Ablation Studies

In Sec. 4.1 of the main paper, we presented an ablation study on the temporal buffer size and on the cell dimensions used to query 3D points. Here, we provide three additional ablation studies. The first, in Sec. **E.1**, analyzes the performance of our **SLAM-MER** pipeline for different cell dimensions. The second, in Sec. **E.2**, analyzes the keyframe creation criteria. The final ablation, in Sec. **E.3** study analyzes dense reconstruction results for different numbers of anchor points and points per anchor.

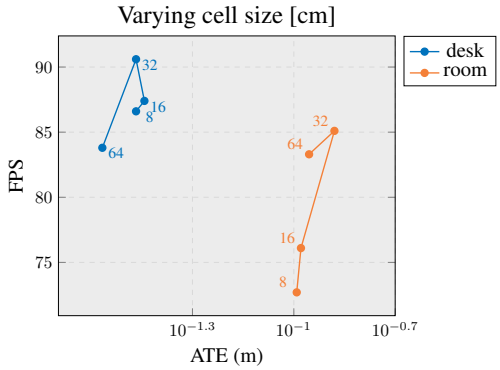


Figure E.4. Ablation study on the size of the cells (annotated next to each marker) used for spatially querying 3D points.

Table E.5. Ablation study on keyframe creation criteria in TUM RGB-D [16].

# Tracked Map-Points	Points Spread	D_{KL} Score	`desk`				`room`			
			ATE	$\lvert\mathcal{K}\rvert$	$\lvert\mathcal{P}^w\rvert$	FPS	ATE	$\lvert\mathcal{K}\rvert$	$\lvert\mathcal{P}^w\rvert$	FPS
✓	✗	✗	0.041	17	4796	92.1	0.119	63	16764	75.2
✓	✓	✗	0.039	18	4871	92.4	0.107	63	16642	78.6
✓	✗	✓	0.039	18	4973	93.3	0.112	66	16927	78.8
✓	✓	✓	0.038	19	5195	89.3	0.096	68	17377	78.6

E.1. Cell Dimensions

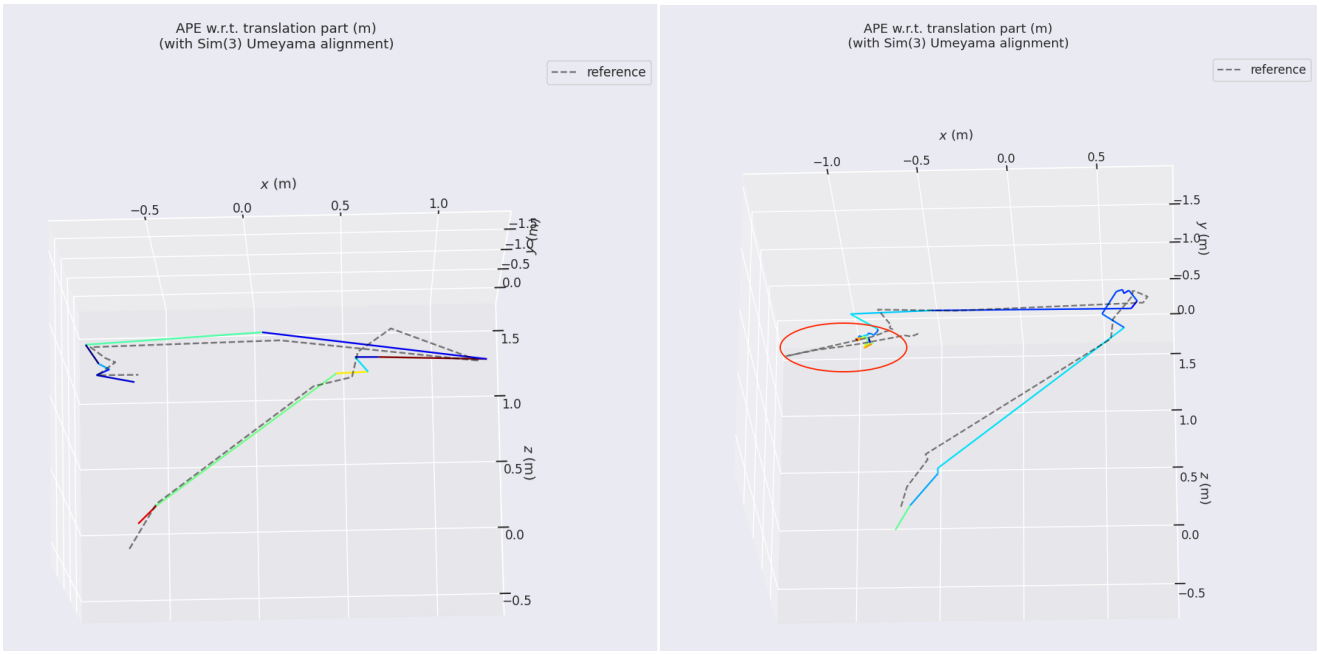
In the ablation study presented in Sec. 4.1 of the main paper, which analyzes the effect of cell size on the spatial querying of 3D points, we observed that increasing the cell size from 8 to 32 cm slightly increases the trajectory error but significantly improves the Frames Per Second (FPS). Based on those results and our goal of maximizing efficiency, we set the cell size to 32 cm for all of our experiments.

Although the cell-related ablation in the main paper focuses solely on spatial querying, here we provide additional results in which the cell size is varied while also utilizing the temporal query ($|\mathcal{B}|= 100$) and loop closure. Results are shown in Fig. **E.4**. When cell size inceases beyond 32 cm, we observe a decrease in FPS. This occurs because larger cells cause more 3D map-points to be selected during the spatial query–including points that are not in the camera's current field of view. This increases the time required to establish 3D–2D correspondences and raises the likelihood of outliers, which in turn degrades the performance of camera pose estimation.

Furthermore, we rasterize per cell to determine whether a cell lies within the camera's view. With larger cells, we may encounter situations where many visible points are discarded due to occlusions affecting entire cells.

E.2. Keyframe Creation

Sec. 3.6 of the main paper detailed the three keyframe creation rules we utilize in our pipeline. To summarize, the



(a) Trajectory obtained using MASt3R-SLAM

(b) Trajectory obtained using SLAM-MER

Figure E.5. Issue in the ground truth trajectory (dashed line) of the `pumpkin` sequence from 7-Scenes. The red ellipse highlights the part of the trajectory that does not match the actual camera motion–as seen from the input frames. We note that the ground truth trajectory shown here only consists of the frames used for evaluation.

first rule ensures we create a keyframe when the number of inliers from pose estimation is low, *i.e.*, the current frame does not track enough 3D map-points. We refer to this criterion as the number of tracked map-points (# tracked map-points). The second rule checks how well the tracked map-points are spread across the image. When map-points are not well spread, we create a new keyframe. The final rule compares the distribution of the tracked map-points across keyframes. Keyframes are created when there is a sudden variation in consecutive frames of the keyframes that observe the tracked map-points. To measure this sudden variation, we compute the Kullback–Leibler divergence D_{KL} of the histogram of the number of tracked 3D points per keyframe.

Table **E.5** compares the performance of combinations of the keyframe creation rules. The rule based on the number of tracked map-points is always switched on for the experiments, because the lack of it results in pose estimation being more prone to failure due to less coverage of 3D map-points in different scene parts. Thus, we can easily lose the camera pose, triggering the re-localization operation mode more often. The remaining two rules are switched on and off in turn. We observe that using the point spread and the KL divergence score slightly increases the number of keyframes in both sequences we tested on. This is exactly the behaviour we expected from these criteria, *i.e.*, create keyframes not only when needed because there are few points but also based on the (1) spread of the 2D locations of tracked map-points, and (2) distribution per keyframe of the points being tracked (D_{KL} score). Besides creating more keyframes, using these criteria reduces trajectory error, and FPS differences are not significant.

Table E.6. Ablation study on semi-dense reconstruction results. We vary the number of anchor points (*i.e.*, map points) and the number of nearby points sampled per anchor on the 7-Scenes dataset [15]. Our default setting is highlighted.

# Anchors	# Points Per Anchor	Acc. ↓	Comp. ↓	Chamfer ↓
$4k$	20	0.034	0.135	0.084
$4k$	5	0.034	0.145	0.090
$4k$	60	0.033	0.097	0.065
$8k$	20	0.036	0.132	0.084

E.3. Dense Reconstruction

Sec. 3.9 of the main paper details how we obtain a semi-dense scene representation from sparse map points. The main paper also shows that the sparse map points lead to a good reconstruction accuracy but lack scene completion since we rely on the map points for anchoring the semi-dense representation. In this experiment, we enforce the creation of additional map points indirectly by extracting more features from the input images, and vary the number of points sampled per anchor point. Results are shown

in Tab. **E.6**. We can observe that (1) increasing the number of anchor points only leads to a marginal variation in completion and accuracy results, and (2) the number of points per anchor higly affects completion. Additionaly, increasing the number of points per anchor has a negligible effect on FPS of SLAM since they are sampled during keyframe creation and updated in the Adjustment module, which runs in parallel to the SLAM main thread.

F. Discussion on 7-Scenes: `pumpkin` sequence

In Tab. 2b of the main paper, we mentioned that we omitted the `pumpkin` sequence from the 7-Scenes dataset for evaluation. During the experiments on the `pumpkin` sequence, we noticed that a part of the ground truth trajectory seemed grossly incorrect–please see Fig. **E.5**. In the final part of the sequence, the camera undergoes an almost pure rotation, which is consistent with the trajectory recovered by both our method Fig. **E.5b** and MASt3R-SLAM Fig. **E.5a**. However, the reference ground-truth trajectory from the dataset shows a large jump of almost 1 meter in the negative x direction, which is not consistent with the camera motion visible in the images. This is evident from the ground truth trajectory shown in the red ellipse in Fig. **E.5b**.

This issue appears to be related to drift at specific timestamps–note that this is an unrealistically large motion for a single keyframe, based on the dataset creation strategy publicly available. We believe MASt3R-SLAM does not show this artifact because, unlike our pipeline that takes all frames as input, MASt3R-SLAM skips frames to maintain an input rate of around 10 FPS.

References

[1] Gabriele Berton and Carlo Masone. Megaloc: One retrieval to place them all. In *Conference on Computer Vision and Pattern Recognition Workshops*, pages 2886–2892, 2025. 1, 2, 3

[2] Mårten Björkman, Niklas Bergström, and Danica Kragic. Detecting, segmenting and tracking unknown objects using multi-label mrf inference. *Computer Vision and Image Understanding*, 118:111–127, 2014. 2

[3] G. Bradski. The OpenCV Library, 2000. 3, 4

[4] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, pages 1–2, 2004. 3

[5] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Conference on Computer Vision and Pattern Recognition Workshops*, pages 224–236, 2018. 2

[6] ONNX Runtime developers. Onnx runtime. `https://onnxruntime.ai/`, 2021. 2, 3

[7] Viktor Larsson and contributors. PoseLib - Minimal Solvers for Camera Pose Estimation, 2020. 1

[8] Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Grounding image matching in 3D with MASt3R. In *European Conference on Computer Vision (ECCV)*, pages 71–91, 2024. 2, 3

[9] David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–110, 2004. 2

[10] Dominic Maggio, Hyungtae Lim, and Luca Carlone. Vggt-slam: Dense rgb slam optimized on the sl (4) manifold. *Advances in Neural Information Processing Systems (NeurIPS)*, 39, 2025. 5

[11] Riku Murai, Eric Dexheimer, and Andrew J. Davison. Mast3r-slam: Real-time dense slam with 3d reconstruction priors. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16695–16705, 2025. 5

[12] S.M. Pizer, R.E. Johnston, J.P. Ericksen, B.C. Yankaskas, and K.E. Muller. Contrast-limited adaptive histogram equalization: speed and effectiveness. In *Conference on Visualization in Biomedical Computing*, pages 337–345, 1990. 4

[13] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2564–2571, 2011. 2

[14] Thomas Schops, Torsten Sattler, and Marc Pollefeys. Bad slam: Bundle adjusted direct rgb-d slam. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 134–144, 2019. 4

[15] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. Scene coordinate regression forests for camera relocalization in rgb-d images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2930–2937, 2013. 3, 4, 6

[16] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 573–580, 2012. 2, 3, 4, 5

[17] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. VGGT: Visual geometry grounded transformer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5294–5306, 2025. 2, 3

[18] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. DUSt3R: Geometric 3D vision made easy. In *IEEE/CVF Conference*

on Computer Vision and Pattern Recognition (CVPR), pages 20697–20709, 2024. 2, 3

[19] Xiaoming Zhao, Xingming Wu, Weihai Chen, Peter CY Chen, Qingsong Xu, and Zhengguo Li. ALIKED: A lighter keypoint and descriptor extraction network via deformable transformation. *IEEE Transactions on Instrumentation and Measurement*, 72:1–16, 2023. 1, 2