MASt3R-SLAM: Real-Time Dense SLAM with 3D Reconstruction Priors

Supplementary Material

8. Analytical Jacobians

In this section, we derive analytical Jacobians used in second-order optimisation for both the tracking and backend. For more information on Lie algebra and relevant Jacobians, please see the following [37, 44].

To take the derivatives on Lie groups with respect to the minimal parameterisation, we use the left-Jacobian definition:

$$\frac{\mathcal{D}f(\mathbf{T})}{\mathcal{D}\mathbf{T}} \triangleq \lim_{\boldsymbol{\tau}\to 0} \frac{f(\boldsymbol{\tau}\oplus\mathbf{T})\ominus f(\mathbf{T})}{\boldsymbol{\tau}}, \tag{11}$$

$$= \lim_{\boldsymbol{\tau}\to 0} \frac{\mathrm{Log}\left(f\left(\mathrm{Exp}(\boldsymbol{\tau})\circ\mathbf{T}\right)\circ f(\mathbf{T})^{-1}\right)}{\boldsymbol{\tau}}. \tag{12}$$

8.1. Points

For the point alignment used in both tracking and mapping, we have a residual defined between a measured point in one frame and a transformed point matched from a different frame. Using the general notation from the backend for point alignment, and switching the order of the residual which does not affect the cost function, the residual is:

$$r_p = \mathbf{T}_{ij}\tilde{\mathbf{X}}^j_{j,n} - \tilde{\mathbf{X}}^i_{i,m}. \tag{13}$$

Defining $\mathbf{x} = \mathbf{T}_{ij}\tilde{\mathbf{X}}^j_{j,n}$ for brevity in deriving Jacobians for a single point, we take the partial derivatives with respect to the Lie algebra perturbation of the relative pose $\mathbf{T}_{ij}$:

$$\frac{\mathcal{D}r_p}{\mathcal{D}\mathbf{T}_{ij}} = \begin{bmatrix}\mathbf{I}_{3\times 3} & -[\mathbf{x}]_\times & \mathbf{x}\end{bmatrix} \tag{14}$$

where $[\mathbf{x}]_\times$ is the 3×3 skew-symmetric matrix.

8.2. Rays and Distance

Compared to the point residual, the ray residual minimises the error in normalised space, which is equivalent to minimising the angle between rays in the camera's frame:

$$r_\psi = \psi\left(\mathbf{T}_{ij}\tilde{\mathbf{X}}^j_{j,n}\right) - \psi\left(\tilde{\mathbf{X}}^i_{i,m}\right). \tag{15}$$

The Jacobian now is the chain rule of the Jacobian for normalising a point to a unit vector and Jacobian of the the pose acting on the point:

$$\frac{\mathcal{D}_\psi}{\mathcal{D}\mathbf{T}_{ij}} = \frac{\partial r_\psi}{\partial \mathbf{x}}\frac{\mathcal{D}\mathbf{x}}{\mathcal{D}\mathbf{T}_{ij}}. \tag{16}$$

Defining the distance from the origin of camera i to point $\mathbf{x}$ as $d_\mathbf{x}$, the Jacobian of the first term becomes:

$$\frac{\partial r_\psi}{\partial \mathbf{x}} = \frac{1}{d_\mathbf{x}}\left(\mathbf{I}_{3\times 3} - \frac{\mathbf{x}\mathbf{x}^T}{d_\mathbf{x}^2}\right). \tag{17}$$

Using the chain rule with Eq. (14), the first term becomes Eq. (17) itself. Since the cross product of a point with itself is a zero vector, the second term becomes a scaled version of the skew-symmetric matrix. Lastly, as Eq. (17) has the form of an operator that takes the difference between a point and its orthogonal projection onto a subspace, and projecting a point onto its own subspace preserves the point, this cancels to a zero vector. In matrix form, this is:

$$\frac{\mathcal{D}r_\psi}{\mathcal{D}\mathbf{T}_{ij}} = \begin{bmatrix}\frac{\partial r_\psi}{\partial \mathbf{x}} & -\frac{1}{d_\mathbf{x}}[\mathbf{x}]_\times & \mathbf{0}_{3\times 1}\end{bmatrix}. \tag{18}$$

As mentioned in the main paper, we also include an error based on the distance between the transformed point and its match so that cases with pure rotation do not result in a degenerate optimisation problem. This error is:

$$r_d = d\left(\mathbf{T}_{ij}\tilde{\mathbf{X}}^j_{j,n}\right) - d\left(\tilde{\mathbf{X}}^i_{i,m}\right) \tag{19}$$

and its corresponding Jacobians are:

$$\frac{\partial r_d}{\partial \mathbf{x}} = \frac{\mathbf{x}^T}{d_\mathbf{x}}, \tag{20}$$

$$\frac{\mathcal{D}r_d}{\mathcal{D}\mathbf{T}_{ij}} = \begin{bmatrix}\frac{\mathbf{x}^T}{d_\mathbf{x}} & \mathbf{0}_{1\times 3} & d_\mathbf{x}\end{bmatrix}. \tag{21}$$

8.3. Projection and Depth

In the case of known calibration, we instead use a pixel error instead of ray error. While the rays could also be constrained to the known camera model, we chose to use pixel error as this better models the noise distribution in pixel-level correspondence and is standard in bundle adjustment. The pixel error is defined as:

$$r_\Pi = \Pi\left(\mathbf{T}_{ij}\tilde{\mathbf{X}}^j_{j,n}\right) - \mathbf{p}^i_{i,m}. \tag{22}$$

Using a pinhole camera model with calibration

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1\end{bmatrix}, \tag{23}$$

the projection Jacobian of point $\mathbf{x} = [x, y, z]^T$ is

$$\frac{\partial r_\Pi}{\partial \mathbf{x}} = \frac{1}{z}\begin{bmatrix} f_x & 0 & -f_x\frac{x}{z} \\ 0 & f_y & -f_y\frac{y}{z}\end{bmatrix}. \tag{24}$$

We can then obtain $\frac{\mathcal{D}r_\Pi}{\mathcal{D}\mathbf{T}_{ij}}$ via the chain rule with Eq. (14). We also include a small error on the predicted and measured depth with similar motivation to Eq. (21) in cases of pure rotation. In the future, any parametric camera model and its corresponding Jacobian could be used here.

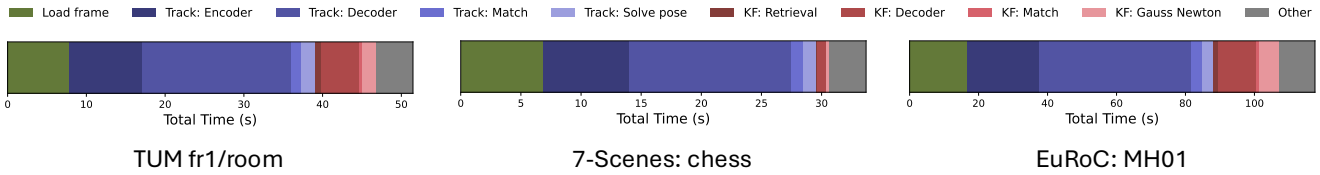

Figure 8. Total runtime in seconds for representative datasets showing cumulative time spent in significant components. The network encoder and decoder are the majority of the runtime at an average of 64% of the total runtime. Datasets with more loop closures like fr1/room and MH01 show more time spent in the backend.

Table 8. Average runtimes in milliseconds of different components for our single-threaded system.

	Data	Per-Frame Tracking					Per Keyframe					Summary	
	Load frame	Encoder	Decoder	Match	Solve pose	Total	Retrieval	Decoder	Match	Gauss-Newton	Total	Total time (s)	FPS
TUM: fr1/room	11.4	13.8	27.7	1.9	2.7	48.6	14.4	97.7	5.8	37.8	157.4	51.5	13.2
7-Scenes: chess	13.8	14.4	26.9	2.0	2.1	47.9	14.4	70.6	4.5	22.7	114.0	33.7	14.8
EuRoC: MH01	9.1	11.4	23.9	1.7	1.8	41.1	15.1	130.4	8.4	66.8	223.3	117.6	15.7
Average	11.4	13.2	26.2	1.9	2.2	45.9	14.6	99.5	6.2	42.4	164.9	67.6	14.6

8.4. From Relative Pose to Global Pose

While the above derivations show the Jacobians with respect to relative camera poses, we ultimately need updates with respect to camera poses in the world frame. Using $\mathbf{T}_{ij} = \mathbf{T}_{WC_i}^{-1}\mathbf{T}_{WC_j}$ and the identities for the left Jacobian of the group inverse and composition

$$\frac{\mathcal{D}\mathbf{T}_{WC_i}^{-1}}{\mathcal{D}\mathbf{T}_{WC_i}} = -\mathrm{Ad}_{\mathbf{T}_{WC_i}^{-1}}, \tag{25}$$

$$\frac{\mathcal{D}\mathbf{T}_{ij}}{\mathcal{D}\mathbf{T}_{WC_i}^{-1}} = \mathbf{I}_{7\times 7}, \tag{26}$$

$$\frac{\mathcal{D}\mathbf{T}_{ij}}{\mathcal{D}\mathbf{T}_{WC_j}} = \mathrm{Ad}_{\mathbf{T}_{WC_i}^{-1}}, \tag{27}$$

we can then solve for updates to each pose:

$$\frac{\mathcal{D}r_\psi}{\mathcal{D}\mathbf{T}_{WC_i}} = \frac{\mathcal{D}r_\psi}{\mathcal{D}\mathbf{T}_{ij}}\frac{\mathcal{D}\mathbf{T}_{ij}}{\mathcal{D}\mathbf{T}_{WC_i}} = -\frac{\mathcal{D}r_\psi}{\mathcal{D}\mathbf{T}_{ij}}\mathrm{Ad}_{\mathbf{T}_{WC_i}^{-1}}, \tag{28}$$

$$\frac{\mathcal{D}r_\psi}{\mathcal{D}\mathbf{T}_{WC_j}} = \frac{\mathcal{D}r_\psi}{\mathcal{D}\mathbf{T}_{ij}}\frac{\mathcal{D}\mathbf{T}_{ij}}{\mathcal{D}\mathbf{T}_{WC_j}} = \frac{\mathcal{D}r_\psi}{\mathcal{D}\mathbf{T}_{ij}}\mathrm{Ad}_{\mathbf{T}_{WC_i}^{-1}}. \tag{29}$$

9. Initialisation

As mentioned in Sec. 3.3, to minimise the number of network passes required for tracking, we re-use the last keyframe's pointmap estimate $\tilde{\mathbf{X}}_k^k$. Such pointmap is always available, apart from at the initialisation. To initialise the system, we simply feed the same image into MASt3R to perform monocular prediction of the pointmap. While such monocular predictions are often inaccurate, the pointmap incorporates multiview information and is refined using the running weighted average filter.

10. Runtime Breakdown

We report the cumulative runtime for different components of our system across three representative datasets in Fig. 8. We also show average runtimes of different components in Tab. 8. Note that tracking, which operates at greater than 20 FPS, occurs for every frame while keyframing is dependent on the motion and thus occurs at a lower frequency. In general, the network encoder and decoder are the most significant in terms of time spent for both the tracking and backend at around 64% of the total runtime. As a large number of loop closures are detected in TUM fr1/room and EuRoC MH01, the time spent in the backend increases compared to the more linear trajectory in 7-Scenes chess. Our efficient matching, tracking, and backend optimisation ensure that we can achieve real-time performance, with the network currently being the limiting factor on lower-latency SLAM. The combination of the modular prior and principled backend optimisation achieves global consistency in real-time.

11. Evaluation Setup

11.1. Trajectory Evaluation [Sec. 4.1]]

For all the datasets, we use the same parameters with keyframe threshold $\omega_k = 0.333$, loop-closure threshold $\omega_l = 0.1$, and $\omega_r = 0.005$. For relocalisation, we have a stricter check to allow for the current frame to be attached to the graph. The match fraction must be greater than 0.3 for all datasets apart from in ETH3D where we set the threshold higher to 0.5.

For trajectory evaluation, we run DROID-SLAM using the open-source code with the configuration files given for each dataset. For 7-Scenes, we use the TUM configuration file since it is the most similar. For TUM and EuRoC, the remaining entries are from the tables in Deep Patch Visual

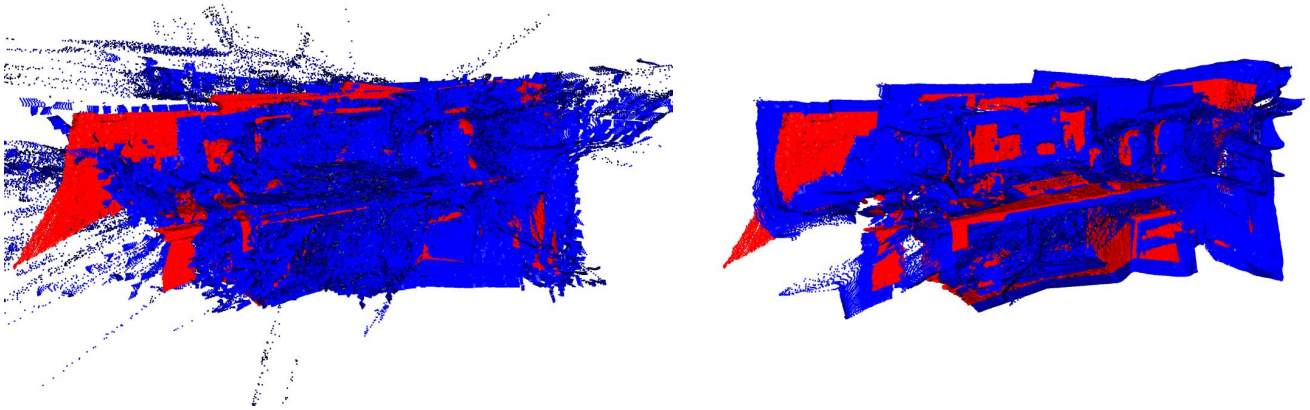

Figure 9. Reconstruction comparison on 7-Scenes heads, with red indicating the ground-truth point cloud and blue the estimated point cloud. While mean Chamfer distance does not significantly penalise inconsistent points, RMSE Chamfer is a better reflection of the quality of the geometry.

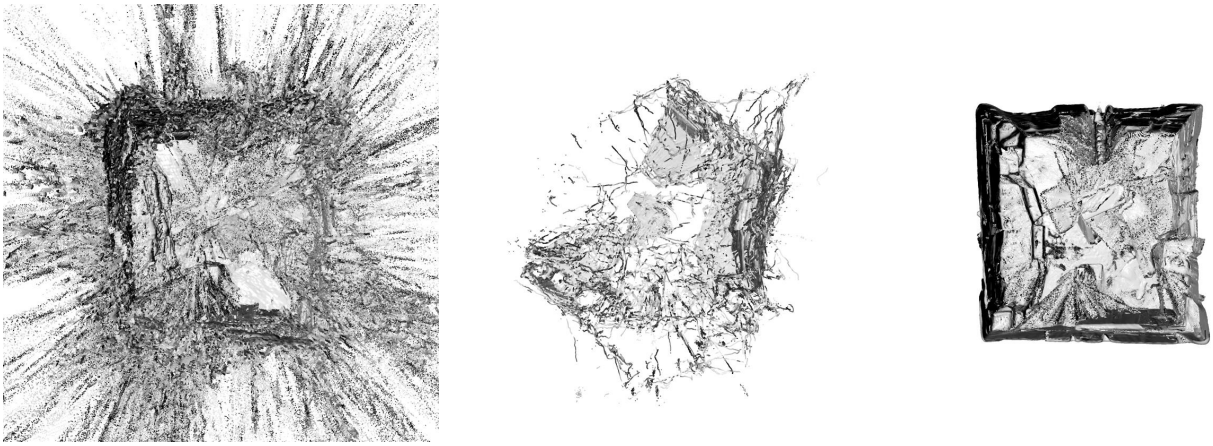

Figure 10. Reconstruction comparison on EuRoC V102.

SLAM [22], which also uses some results from DROID-SLAM [45]. For 7-Scenes, we include the results reported from NICER-SLAM [58]. For ETH3D, we ran all methods locally as the dataset was not previously attempted with monocular SLAM methods.

11.2. Geometry Evaluation [Sec. 4.2]

For evaluation, points that are unobservable are removed from the reference point cloud. Additionally, for the 7-Scenes dataset, we filter out depths which are marked as invalid. For all methods, we do not filter any estimated point, as in an incremental problem setting like SLAM, reprojection-based filtering is not always possible and downstream applications benefit from per-pixel dense prediction.

For the metrics, we report the RMSE which penalises outlying measurements. Fig. 9 is an illustrative example, where DROID-SLAM and MASt3R-SLAM achieve a similar mean Chamfer distance. Qualitatively, however, MASt3R-SLAM clearly produces more coherent and accurate geometry, and this difference is reflected in the RMSE Chamfer distance.

We report the qualitative result of EuRoC reconstruction in Fig. 10. Spann3R fails as the sequence is not object-centric, and DROID-SLAM produces many more outliers compared to MASt3R-SLAM . Compared to Spann3R which maintains a memory buffer, our keyframing system ensures that viewed parts of the scene are not discarded. Furthermore, our efficient global optimisation can create globally consistent maps in real-time.

Table 9. Absolute trajectory error (ATE (m)) on EuRoC [3].

		MH01	MH02	MH03	MH04	MH05	V101	V102	V103	V201	V202	V203	avg
Calibrated	**ORB-SLAM**	0.071	0.067	0.071	0.082	0.060	**0.015**	0.020	X	0.021	0.018	X	-
	DeepV2D [42]	0.739	1.144	0.752	1.492	1.567	0.981	0.801	1.570	0.290	2.202	2.743	1.298
	DeepFactors [6]	1.587	1.479	3.139	5.331	4.002	1.520	0.679	0.900	0.876	1.905	1.021	2.040
	DPV-SLAM [22]	**0.013**	0.016	0.022	0.043	**0.041**	0.035	**0.008**	**0.015**	0.020	0.011	0.040	0.024
	DPV-SLAM++ [22]	**0.013**	0.016	**0.021**	**0.041**	**0.041**	0.035	0.010	**0.015**	0.021	0.011	0.023	0.023
	GO-SLAM [54]	0.016	0.014	0.023	0.045	0.045	0.037	0.011	0.023	**0.016**	**0.010**	0.022	0.024
	DROID-SLAM [45]	**0.013**	**0.012**	0.022	0.048	0.044	0.037	0.013	0.019	0.017	**0.010**	**0.013**	**0.022**
	Ours	0.023	0.017	0.057	0.113	0.067	0.040	0.019	0.027	0.020	0.025	0.043	0.041
Uncalibrated	**Ours***	0.180	0.124	0.156	0.282	0.327	0.101	0.134	0.096	0.133	0.100	0.170	0.164

12. EuRoC Results

We summarise the average ATE for EuRoC in the main paper, and show the results for each sequence in Tab. 9. While our system does not outperform DROID-SLAM and methods that leverage its matching architecture, EuRoC has traditionally been challenging for monocular systems due to aggressive motion, large-scale trajectories, and varying exposure. As noted previously, DROID-SLAM was trained with explicit greyscale augmentation which may account for the gap in performance. Compared to previous systems with geometric priors, such as DeepV2D and DeepFactors, we demonstrate significant improvements in trajectory estimation. Furthermore, the results from the main paper highlight the additional benefits of using such a prior, as the dense geometry is more accurate and consistent as shown in Tab. 3, even for our uncalibrated system.

13. Comparison to Other SLAM/SfM Methods

13.1. DROID and DPV SLAM

Our system uses a two-view geometric prior in a modular system, while DROID and DPV SLAM learn a matching prior as part of an end-to-end system with differentiable bundle adjustment. While these systems are very accurate for pose estimation, there are fundamental limitations for geometry and generality.

First, bundle adjustment cannot guarantee coherent geometry even with accurate poses, as it lacks smoothness regularisation and constraints under low parallax. Given MASt3R, we find local fusion and scale optimisation to be sufficient for consistency and coherence, while the BA of DROID loses the latter. Second, beyond improved geometry, geometric priors enable new capabilities like continuously changing intrinsics. DROID fixes the model to pinhole during training, and also cannot efficiently handle time-varying intrinsics as this slows down backend optimisation.

13.2. MASt3R-SfM

MASt3R-SfM uses sparse correspondences (subsampling 1/64 pixels) due to MASt3R's brute-force matching. Our dense projective pointmap matching formulates search as local optimisation and achieves a 1000x speedup without compromising accuracy as shown in Tab. 4. Global optimisation in MASt3R-SfM uses a 1st-order optimiser, lacks minimal rotation updates, and introduces degenerate solutions that require scale renormalisation. To avoid such problems, we formulate a nonlinear least-squares problem and develop a 2nd-order optimiser with minimal pose updates and gauge fixing. Our uncalibrated ray formulation achieves a similar accuracy as MASt3R-SfM's procedure of fitting pinhole models and minimising reprojection error, but we avoid selecting a specific camera model. This maintains generality of our SLAM system in order to handle all types of distortion, such as fisheye in the future.