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

Riku Murai* Eric Dexheimer* Andrew J. Davison
Imperial College London
{riku.murai15, e.dexheimer21, a.davison}@imperial.ac.uk
Project Page: https://edexheim.github.io/mast3r-slam
Video: https://youtu.be/wozt71NBFTQ

Abstract

We present a real-time monocular dense SLAM system designed bottom-up from MASt3R, a two-view 3D reconstruction and matching prior. Equipped with this strong prior, our system is robust on in-the-wild video sequences despite making no assumption on a fixed or parametric camera model beyond a unique camera centre. We introduce efficient methods for pointmap matching, camera tracking and local fusion, graph construction and loop closure, and second-order global optimisation. With known calibration, a simple modification to the system achieves state-of-the-art performance across various benchmarks. Altogether, we propose a plug-and-play monocular SLAM system capable of producing globally consistent poses and dense geometry while operating at 15 FPS.

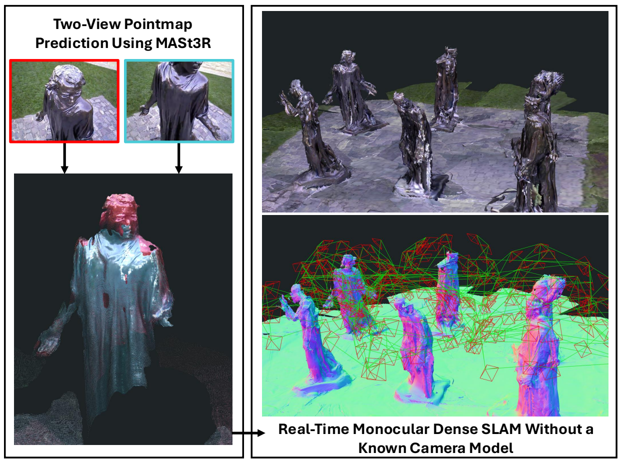

Two-View Pointmap Prediction Using MASt3R
Real-Time Monocular Dense SLAM Without a Known Camera Model

Figure 1. Reconstruction from our dense monocular SLAM system on the Burghers sequence [55]. Using the two-view predictions from MASt3R shown on the left, our system achieves globally consistent poses and geometry in real-time even without a known camera model.

1. Introduction

Visual simultaneous localisation and mapping (SLAM) is a foundational building block for today's robotics and augmented reality products. With careful design of an integrated hardware and software stack, robust and accurate visual SLAM is now possible. However, SLAM is yet to become a plug-and-play algorithm as it requires hardware expertise and calibration. Even for a minimal single camera setup with no additional sensing such as an IMU, a SLAM solution that reliably provides both accurate poses and consistent dense maps in-the-wild does not exist. Achieving such a reliable dense SLAM system would open new research avenues for spatial intelligence.

Performing dense SLAM from only 2D images requires reasoning over time-varying poses and camera models, as well as 3D scene geometry. To solve such an inverse problem of large dimensionality, a variety of priors, from handcrafted to data-driven, have been proposed. Single-view priors, such as monocular depth and normals, attempt to predict geometry from a single image, but these contain ambiguities and lack consistency across views. While multi-view priors like optical flow reduce the ambiguity, decoupling pose and geometry is challenging since pixel motion depends on both the extrinsics and the camera model. Although these underlying causes may vary across time and different observers, the 3D scene remains invariant across views. Therefore, the unifying prior required to solve for poses, camera models, and dense geometry from images is over the space of 3D geometry in a common coordinate frame.

Recently, *two-view 3D reconstruction priors*, pioneered by DUSt3R [49] and its successor MASt3R [20], have created a paradigm shift in structure-from-motion (SfM) by capitalising on curated 3D datasets. These networks output pointmaps directly from two images in a common coordinate frame, such that the aforementioned subproblems are

*Authors contributed equally to this work.

implicitly solved in a joint framework. In the future, these priors will be trained on all varieties of camera models with significant distortion. While 3D priors could take in more views, SfM and SLAM leverage spatial sparsity and avoid redundancy to achieve large-scale consistency. A two-view architecture mirrors two-view geometry as the building block of SfM, and this modularity opens the door for both efficient decision-making and robust consensus in the backend.

In this work, we propose the first real-time SLAM framework to leverage two-view 3D reconstruction priors as a unifying foundation for tracking, mapping, and relocalisation as shown in Fig. 1. While previous work has applied these priors to SfM in an offline setting with unordered image collections [10], SLAM receives data incrementally and must maintain real-time operation. This requires new perspectives on low-latency matching, careful map maintenance, and efficient methods for large-scale optimisation. Furthermore, inspired by both filtering and optimisation techniques in SLAM, we perform local filtering of pointmaps in the frontend to enable large-scale global optimisation in the backend. Our system makes no assumption on each image's camera model beyond having a unique camera centre that all rays pass through. This results in a real-time dense monocular SLAM system capable of reconstructing scenes with generic, time-varying camera models. Given calibration, we also demonstrate state-of-the-art performance in trajectory accuracy and dense geometry estimation.

In summary, our contributions are:

- The first real-time SLAM system using the two-view 3D reconstruction prior MASt3R [20] as a foundation.
- Efficient techniques for pointmap matching, tracking and local fusion, graph construction and loop closure, and second-order global optimisation.
- A state-of-the-art dense SLAM system capable of handling generic, time-varying camera models.

2. Related Work

To obtain accurate pose estimation, **sparse monocular SLAM** focuses on jointly solving for camera poses and a select number of unbiased 3D landmarks [7]. Algorithmic advances leveraging the sparsity of the optimisation [18] and careful graph construction [25] enabled real-time pose estimation and sparse reconstructions on large scale scenes. While sparse monocular SLAM is very accurate given sufficient features and parallax, it lacks a dense scene model which is useful for both robust tracking and more explicit reasoning over geometry.

To improve robustness and provide interaction, early **dense monocular SLAM** systems demonstrated alternating optimisation of poses and dense depth with handcrafted regularisation [28]. As these systems were limited to controlled settings, recent work has attempted to combine data-driven priors with backend optimisation. While predicting geometric quantities from a single image, such as depth [11, 14, 30, 52] and surface normals [1, 50], have shown significant progress, their use has been limited in SLAM. Predicting geometry from a single-view is ambiguous, resulting in biased and inconsistent 3D geometry. SLAM literature has thus focused on predicting priors over a hypothesis space of possible depths in the form of latent spaces [2, 6], subspaces [40], local primitives [23], and distributions [8, 9]. While the flexibility of these priors can achieve greater consistency, robust correspondence across multiple views is essential.

Multi-view priors, such as multi-view stereo (MVS) [19, 32, 54] and optical flow [42], instead focus on learning correspondence from two or more views as a means to obtaining geometry. However, both require additional information: MVS fixes poses to achieve correspondence, while flow is an entangled observation of motion and geometry subject to the degeneracies mentioned previously. DROID-SLAM [44] combines learned features for matching along with a per-pixel dense bundle adjustment framework into a single end-to-end framework. This results in a robust SLAM system with a backend similar in spirit to sparse SLAM, so the lack of explicit geometric constraints can still produce inconsistent 3D geometry.

Volumetric representations have demonstrated the potential for consistent reconstruction as geometry parameters are coupled in the rendering process. A variety of SLAM systems have adopted differentiable rendering in neural fields [24] and Gaussian splatting [17] for both monocular [22, 57] and RGB-D [15, 38, 51, 56] cameras. However, these methods have lagged in real-time performance compared to alternatives, and require depth, additional 2D priors, or slow camera motion to constrain the solution. 3D priors for general scene reconstruction from images first fuse 2D features into 3D voxel grids which are then decoded into surface geometry [26, 39]. These methods assume known poses for fusion, so are unsuitable for joint tracking and mapping, while the volumetric representations require significant memory and a pre-defined resolution.

All systems mentioned thus far assume known intrinsic calibration. Classical **automatic intrinsic calibration** is possible when there are strict assumptions on scene geometry or unchanging parameters across a set of images [12], but encounters degenerate configurations and sensitivity to noise. Given an initial estimate of intrinsics, refinement via bundle adjustment can improve accuracy online [16], but this already assumes a parametric model and sufficient initialisation of all parameters. More recently, data-driven methods predict intrinsics from one or multiple images [13, 47], but are either limited in accuracy for in-the-wild SLAM or are not flexible in the camera model definition.

Recently, DUSt3R introduced a novel two-view **3D reconstruction prior** that outputs dense 3D point clouds of both images in a *common* coordinate frame. Compared to previously discussed priors that solve subproblems of the task, DUSt3R provides a direct pseudo-measurement of a two-view 3D scene by implicitly reasoning over correspondence, poses, camera models, and dense geometry. The successor MASt3R [20] predicts additional per-pixel features to improve pixel matching for localisation and SfM [10]. However, as with all priors, predictions can still have inconsistencies and correlated errors in the 3D geometry. DUSt3R and MASt3R-SfM thus require large-scale optimisation for global consistency, but the time complexity does not scale well with the number of images. Spann3R [48] forgoes backend optimisation by fine-tuning DUSt3R to predict a stream of pointmaps directly into a global coordinate system, but must maintain a limited memory of tokens which can cause drift in larger scenes.

In this work, we propose a dense SLAM system built around these two-view 3D reconstruction priors. We only assume a generic central camera model, and propose efficient methods for pointmap matching, tracking and pointmap fusion, loop closure, and global optimisation to achieve large scale consistency of the pairwise predictions in real-time.

3. Method

We provide an overview of the method in Fig. 3, which shows our main components: MASt3R prediction and pointmap matching, tracking and local fusion, loop closure, and global optimisation.

3.1. Preliminaries

DUSt3R takes in a pair of images $\mathcal{I}^i, \mathcal{I}^j \in \mathbb{R}^{H\times W\times 3}$, and outputs pointmaps $\mathbf{X}^i_i, \mathbf{X}^j_i \in \mathbb{R}^{H\times W\times 3}$ along with their confidences $\mathbf{C}^i_i, \mathbf{C}^j_i \in \mathbb{R}^{H\times W\times 1}$. Here, we use notation $\mathbf{X}^i_j$ to express the pointmap of image i represented in the coordinate frame of camera j. In MASt3R, an additional head is added to predict d-dimensional features for matching $\mathbf{D}^i_i, \mathbf{D}^j_i \in \mathbb{R}^{H\times W\times d}$ and its corresponding confidences $\mathbf{Q}^i_i, \mathbf{Q}^j_i \in \mathbb{R}^{H\times W\times 1}$. We define $\mathcal{F}_M(\mathcal{I}^i, \mathcal{I}^j)$ as the forward pass of MASt3R that yields the previously discussed outputs, and throughout the text we will use MASt3R's output directly for conciseness.

While some of the data used to train MASt3R has metric scale, we found that scale is often a large source of inconsistency across predictions. To optimise over differently scaled predictions, we define all poses as $\mathbf{T} \in \mathbf{Sim}(3)$ and updates to the poses using Lie algebra $\boldsymbol{\tau} \in \mathfrak{sim}(3)$ and a left-plus operator:

$$\mathbf{T} = \begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix}, \quad \mathbf{T} \leftarrow \boldsymbol{\tau} \oplus \mathbf{T} \triangleq \mathrm{Exp}(\boldsymbol{\tau}) \circ \mathbf{T}, \quad (1)$$

where $\mathbf{R} \in \mathbf{SO}(3)$, $\mathbf{t} \in \mathbb{R}^3$, and scale $s \in \mathbb{R}$, following the notation in [36, 43].

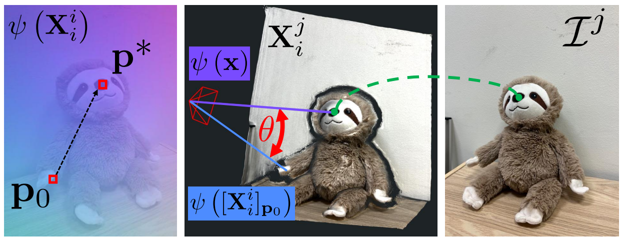

$\psi\left(\mathbf{X}^i_i\right)$
$\mathbf{p}^*$
$\mathbf{p}_0$
$\mathbf{X}^j_i$
$\psi\left(\mathbf{x}\right)$
θ
$\psi\left([\mathbf{X}^i_i]_{\mathbf{p}_0}\right)$
$\mathcal{I}^j$

Figure 2. Overview of iterative projective matching: given the two pointmap predictions from MASt3R, the reference pointmap is normalised $\psi\left(\mathbf{X}^i_i\right)$ to give a smooth pixel to ray mapping. For an initial estimate of the projection $\mathbf{p}_0$ of 3D point $\mathbf{x}$ from pointmap $\mathbf{X}^j_i$, the pixel is iteratively updated to minimise the angular difference θ between the queried ray $\psi\left([\mathbf{X}^i_i]_{\mathbf{p}}\right)$ and the target ray $\psi\left(\mathbf{x}\right)$. After finding the pixel $\mathbf{p}^*$ that achieves the minimum error, we have a pixel correspondence between $\mathcal{I}^i$ and $\mathcal{I}^j$.

Our only assumption on the camera model is that of a generic central camera [34], which means that all rays pass through a unique camera centre. We define the function $\psi\left(\mathbf{X}^i_i\right)$ that normalises a pointmap $\mathbf{X}^i_i$ into rays of unit norm such that each pointmap defines its own camera model. This enables handling both time-varying camera models, such as zoom, and distortion in a unified manner.

3.2. Pointmap Matching

Correspondence is a fundamental component of SLAM that is required for both tracking and mapping. In this case, given the pointmaps and features from MASt3R, we need to find the set of pixel matches between the two images, denoted by $\mathbf{m}_{i,j} = \mathcal{M}(\mathbf{X}^i_i, \mathbf{X}^j_i, \mathbf{D}^i_i, \mathbf{D}^j_i)$. Naive brute-force matching has quadratic complexity since it is a global search over all possible pairs of pixels. To avoid this, DUSt3R uses a k-d tree over 3D points; however, construction is non-trivial to parallelise and the nearest-neighbour search in 3D will find many inaccurate matches if there are errors in the pointmap predictions. In MASt3R, additional high-dimensional features are predicted from the network to achieve wider baseline matching and a coarse-to-fine scheme is proposed to handle the global search. However, the runtime is on the order of seconds for dense pixel matching, and sparse matching is still slower than the k-d tree. Rather than focusing on efficient methods for a global search over matches, we instead find inspiration from optimisation as a local search.

Compared to feature matching, we are motivated by the use of projective data-association commonly used in dense SLAM. However, this requires a parametric camera model with closed-form projection, while our only assumption is that each frame has a unique camera centre. Given the output pointmaps $\mathbf{X}^i_i, \mathbf{X}^j_i$, we can construct the generic camera model of $\mathcal{I}^i$ with the rays $\psi\left(\mathbf{X}^i_i\right)$. Inspired by generic camera calibration methods [31, 34] which lack closed-form projection, we project each point $\mathbf{x} \in \mathbf{X}^j_i$ independently by

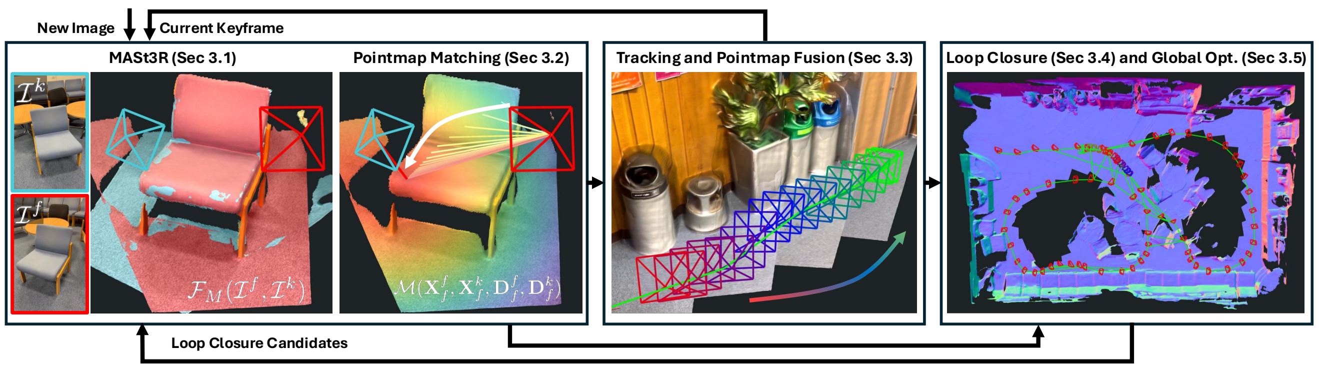

New Image
Current Keyframe
MASt3R (Sec 3.1)
Pointmap Matching (Sec 3.2)
Tracking and Pointmap Fusion (Sec 3.3)
Loop Closure (Sec 3.4) and Global Opt. (Sec 3.5)
$\mathcal{I}^k$
$\mathcal{I}^f$
$\mathcal{F}_M(\mathcal{I}^f, \mathcal{I}^k)$
$\mathcal{M}(\mathbf{X}^f_f, \mathbf{X}^k_f, \mathbf{D}^f_f, \mathbf{D}^k_f)$
Loop Closure Candidates

Figure 3. System diagram of MASt3R-SLAM. New images are tracked against the current keyframe by predicting a pointmap from MASt3R and finding pixel matches using our efficient iterative projection pointmap matching. Tracking estimates the current pose and performs local pointmap fusion. When new keyframes are added to the backend, loop closure candidates are selected by querying the retrieval database using encoded MASt3R features. Candidates are then decoded by MASt3R and if a sufficient number of matches is found, edges are added to the backend graph. Large-scale second-order optimisation achieves global consistency of poses and dense geometry.

iteratively optimising the pixel coordinates $\mathbf{p}$ in the reference frame that minimise the ray error:

$$\mathbf{p}^* = \arg\min_{\mathbf{p}} \left\| \psi\left([\mathbf{X}^i_i]_{\mathbf{p}}\right) - \psi\left(\mathbf{x}\right) \right\|^2. \tag{2}$$

We show a visual overview in Fig. 2, and note that minimising the Euclidean distance between normalised vectors is equivalent to minimising the angle θ between two normalised rays:

$$\left\|\psi_1 - \psi_2\right\|^2 = 2(1-\cos\theta), \quad \cos\theta = \psi_1^T\psi_2. \tag{3}$$

By using the nonlinear least-squares form similar to [34], we can iteratively solve for updates $\delta\mathbf{p}$ by calculating analytical Jacobians $\mathbf{J}$ and using Levenberg-Marquardt:

$$\delta\mathbf{p} = -\left(\mathbf{J}^T\mathbf{J} + \lambda\mathbf{I}\right)^{-1}\mathbf{J}^T\mathbf{r}, \quad \mathbf{p} \leftarrow \mathbf{p} + \delta\mathbf{p}. \tag{4}$$

This can be done separately for each point and converges for almost all valid pixels within 10 iterations as the ray image is smooth. At the end of this process, we now have initial matches $\mathbf{m}_{i,j}$. When there is no initial estimate for the projection $\mathbf{p}$, such as when tracking against a new keyframe or when matching loop closure edges, all pixels are initialised with the identity mapping. During tracking, since we always have the matches from the previous frame, we can use this as initialisation to further speed up the convergence. To handle occlusions and outliers, we also invalidate matches that have large distances in 3D space. Our matching is massively parallel on GPU and additionally can leverage the incremental nature of SLAM.

While these pixels give a good initial estimate of matches using the geometry, MASt3R demonstrates that leveraging per-pixel features greatly improves downstream performance on pose estimation. Since we have a good initialisation from the previous step, we conduct a coarse-to-fine image-based search by updating the pixel location to the maximum feature similarity in a local patch window.

We implement both the iterative projection and feature refinement steps in custom CUDA kernels, as both are parallelisable for each pixel. For tracking this takes only 2 milliseconds and for constructing edges in the graph this takes only a few milliseconds for all newly added edges without any initial estimates of the projections. Note that our matches are unbiased by our pose estimates as they rely purely on the MASt3R outputs, which is atypical for projective data association.

3.3. Tracking and Pointmap Fusion

A key component of SLAM is low-latency tracking of the current frame's pose against the map. As a keyframe-based system, we estimate the relative transformation $\mathbf{T}_{kf}$ between the current frame $\mathcal{I}^f$ and the last keyframe $\mathcal{I}^k$. To be efficient, we would like to use only a single pass of the network to estimate the transformation. Assuming we already have the last keyframe's pointmap estimate $\tilde{\mathbf{X}}^k_k$, we need points in the frame of $\mathcal{I}^f$ to resolve $\mathbf{T}_{kf}$. This can be obtained via $\mathcal{F}_M(\mathcal{I}^f, \mathcal{I}^k)$. One straightforward method to solve for pose is minimising the 3D point error:

$$E_p = \sum_{m,n \in \mathbf{m}_{f,k}} \left\| \frac{\tilde{\mathbf{X}}^k_{k,n} - \mathbf{T}_{kf}\mathbf{X}^f_{f,m}}{w(\mathbf{q}_{m,n}, \sigma_p^2)} \right\|_\rho, \tag{5}$$

where $\mathbf{q}_{m,n} = \sqrt{\mathbf{Q}^f_{f,m}\mathbf{Q}^k_{f,n}}$ is the match confidence weighting proposed in MASt3R-SfM [10]. For robustness, in addition to the Huber norm $\|\cdot\|_\rho$, a per-match weighting is applied:

$$w(\mathbf{q}, \sigma^2) = \begin{cases} \sigma^2/\mathbf{q} & \mathbf{q} > \mathbf{q}_{min} \\ \infty & \text{otherwise} \end{cases}. \tag{6}$$

While $\mathbf{X}_f^k$ instead of $\mathbf{X}_f^f$ could also be aligned to $\mathbf{X}_k^k$, with the benefit of no explicit matching required as they are pixel aligned, we found that explicit matching with $\mathbf{X}_f^f$ had improved accuracy for larger baseline scenarios. More importantly, although the 3D point error is suitable, it is easily skewed by errors in the pointmap predictions as inconsistent predictions in depth are relatively frequent. Since we ultimately fuse predictions into a single pointmap that averages out all the predictions, error in tracking degrades the keyframe's pointmap that will also be used in the backend.

By again exploiting that the pointmap predictions can be converted to rays under a central camera assumption, we can calculate a directional ray error instead, which is less sensitive to incorrect depth predictions. To calculate this, we simply normalise both points from Eq. (5):

$$E_r = \sum_{m,n \in \mathbf{m}_{f,k}} \left\| \frac{\psi\left(\tilde{\mathbf{X}}_{k,n}^k\right) - \psi\left(\mathbf{T}_{kf}\mathbf{X}_{f,m}^f\right)}{w(\mathbf{q}_{m,n}, \sigma_r^2)} \right\|_\rho . \quad (7)$$

This results in a similar angular error as mentioned in Eq. (3) and shown in Fig. 2, except that we now have many known correspondences and wish to find the pose that minimises all angular errors between canonical rays and corresponding predicted rays from the current frame. Since angular errors are bounded, ray-based errors are robust against outliers [29]. We also include an error term with a small weight on the difference in distances from the camera centre. This prevents the system from becoming degenerate in pure rotation cases, but the small weight avoids biasing the pose estimate in the same way that the point error does. We efficiently solve for updates to the pose using Gauss-Newton in an iteratively reweighted least-squares (IRLS) framework. We calculate analytical Jacobians of the ray and distance errors with respect to a perturbation $\boldsymbol{\tau}$ of the relative pose $\mathbf{T}_{kf}$. We stack the residuals, Jacobians, and weights into matrices $\mathbf{r}$, $\mathbf{J}$, and $\mathbf{W}$, respectively. We iteratively solve the linear system and update the pose via:

$$\left(\mathbf{J}^T\mathbf{W}\mathbf{J}\right)\boldsymbol{\tau} = -\mathbf{J}^T\mathbf{W}\mathbf{r}, \quad \mathbf{T}_{kf} \leftarrow \boldsymbol{\tau} \oplus \mathbf{T}_{kf}. \quad (8)$$

Since each pointmap may provide valuable new information, we leverage this by not only filtering over estimates of the geometry, but also over the camera model itself, since it is defined by the rays. After solving for the relative pose, we can use transform $\mathbf{X}_f^k$ and update the canonical pointmap $\tilde{\mathbf{X}}_k^k$ via a running weighted average filter [5, 27]:

$$\tilde{\mathbf{X}}_k^k \leftarrow \frac{\tilde{\mathbf{C}}_k^k\tilde{\mathbf{X}}_k^k + \mathbf{C}_f^k\left(\mathbf{T}_{kf}\mathbf{X}_f^k\right)}{\tilde{\mathbf{C}}_k^k + \mathbf{C}_f^k}, \tilde{\mathbf{C}}_k^k \leftarrow \tilde{\mathbf{C}}_k^k + \mathbf{C}_f^k . \quad (9)$$

The pointmap initially has larger errors and less confidence due to only having small baseline frames, but filtering is able to merge information from many viewpoints. We experimented with different ways of updating the canonical pointmap, and found that weighted average was best for maintaining coherence while filtering out noise. Compared to the canonical pointmap in MASt3R-SfM [10], we compute this incrementally and require transformation of the points since an additional network prediction of $\mathbf{X}_k^k$ would slow down tracking. Filtering has a rich history in SLAM, and yields the benefit of leveraging information from all frames without having to explicitly optimise for all camera poses and store all predicted pointmaps from the decoder in the backend.

3.4. Graph Construction and Loop Closure

When tracking, a new keyframe $\mathcal{K}_i$ is added if the number of valid matches or the number of unique keyframe pixels in $\mathbf{m}_{f,k}$ falls below a threshold ω_k. After adding $\mathcal{K}_i$, a bidirectional edge to the previous keyframe $\mathcal{K}_{i-1}$is added to the edge-list $\mathcal{E}$. This constrains the estimated poses sequentially in time; however, drift can still occur. To close both small and large loops, we adapt the Aggregated Selective Match Kernel (ASMK) [45, 46] framework used by MASt3R-SfM [10] for image retrieval from encoded features. While this was previously used in a batch setting where all images are available from the start, we modify it to work incrementally. We query the database with the encoded features of $\mathcal{K}_i$ to obtain the top-K images. Since the codebook only has tens of thousands of centroids, we found that a dense L2 distance calculation was sufficiently fast to quantise the features. If the retrieval scores are above a threshold ω_r, we give these pairs to the MASt3R decoder and add bidirectional edges if the number of matches determined from Sec. 3.2 is above a threshold ω_l. Lastly, we update the retrieval database by adding the new keyframe's encoded features to the inverted file index.

3.5. Backend Optimisation

Given current estimates of keyframe poses $\mathbf{T}_{WC_i}$ and canonical pointmaps $\tilde{\mathbf{X}}_i^i$ for $\mathcal{K}_i$, the goal of the backend optimisation is to achieve global consistency across all poses and geometry. While previous formulations used first-order optimisation and require rescaling after every iteration [10, 49], we introduce an efficient second-order optimisation scheme that handles the gauge freedom of the problem by fixing the first 7-DoF $\mathbf{Sim}(3)$ pose. We jointly minimise the ray error for all edges $\mathcal{E}$ in the graph:

$$E_g = \sum_{i,j \in \mathcal{E}} \sum_{m,n \in \mathbf{m}_{i,j}} \left\| \frac{\psi\left(\tilde{\mathbf{X}}_{i,m}^i\right) - \psi\left(\mathbf{T}_{ij}\tilde{\mathbf{X}}_{j,n}^j\right)}{w(\mathbf{q}_{m,n}, \sigma_r^2)} \right\|_\rho , \quad (10)$$

where $\mathbf{T}_{ij} = \mathbf{T}_{WC_i}^{-1}\mathbf{T}_{WC_j}$. Given N keyframes, Eq. (10) forms and accumulates 14×14 blocks into the $7N \times 7N$

Table 1. Absolute trajectory error (ATE (m)) on TUM RGB-D [37].

		360	**desk**	**desk2**	**floor**	**plant**	**room**	**rpy**	**teddy**	**xyz**	**avg**
Calibrated	**ORB-SLAM3 [4]**	X	0.017	0.210	X	0.034	X	X	X	**0.009**	-
	DeepV2D [41]	0.243	0.166	0.379	1.653	0.203	0.246	0.105	0.316	0.064	0.375
	DeepFactors [6]	0.159	0.170	0.253	0.169	0.305	0.364	0.043	0.601	0.035	0.233
	DPV-SLAM [21]	0.112	0.018	0.029	0.057	0.021	0.330	0.030	0.084	0.010	0.076
	DPV-SLAM++ [21]	0.132	0.018	0.029	0.050	0.022	0.096	0.032	0.098	0.010	0.054
	GO-SLAM [53]	0.089	**0.016**	0.028	0.025	0.026	0.052	**0.019**	0.048	0.010	0.035
	DROID-SLAM [44]	0.111	0.018	0.042	**0.021**	**0.016**	**0.049**	0.026	0.048	0.012	0.038
	Ours	**0.049**	**0.016**	**0.024**	0.025	0.020	0.061	0.027	**0.041**	**0.009**	**0.030**
Uncalibrated	**DROID-SLAM* [44, 47]**	0.202	0.032	0.091	0.064	0.045	0.918	0.056	0.045	0.012	0.158
	Ours*	0.070	0.035	0.055	0.056	0.035	0.118	0.041	0.114	0.020	0.060

Hessian. We solve this problem again using Gauss-Newton as in Eq. (8) but with sparse Cholesky decomposition as the system is not dense. Construction of the Hessian is made efficient through the use of analytical Jacobians and parallel reductions all implemented in CUDA. Again, a small error term on consistency in distances is added to avoid degeneracy in the pure-rotation case. At most 10 iterations of Gauss-Newton are performed for every new keyframe and optimisation terminates early upon convergence. The use of second-order information greatly speeds up the global optimisation over the alternatives, and our efficient implementation ensures that it is not the bottleneck in the overall system.

3.6. Relocalisation

If the system loses tracking due to an insufficient number of matches, relocalisation mode is triggered. For a new frame, the retrieval database is queried with a stricter threshold on the score. Once the retrieved images have a sufficient number of matches with the current frame, it is then added as a new keyframe into the graph and tracking resumes.

3.7. Known Calibration

Our system works without known camera calibration, but if we do have calibration we can make use of it to improve accuracy via two straightforward changes. First, before canonical pointmaps are used for optimisation in both tracking and mapping, we query only the depth dimension and constrain the pointmap to be backprojected along the rays defined by the known camera model. Second, we change the residuals in optimisation to be in pixel space rather than ray space. In the backend, a pixel $\mathbf{p}^i_{i,m}$ in $\mathcal{I}^i$ is compared against the projection of the 3D point it is matched with:

$$E_\Pi = \sum_{i,j\in\mathcal{E}} \sum_{m,n\in\mathbf{m}_{i,j}} \left\| \frac{\mathbf{p}^i_{i,m} - \Pi\left(\mathbf{T}_{ij}\tilde{\mathbf{X}}^j_{j,n}\right)}{w(\mathbf{q}_{m,n}, \sigma^2_\Pi)} \right\|_\rho , \quad (11)$$

where Π is the projection function to pixel space using the given camera model. Furthermore, the additional distance residuals are converted to depth for consistency.

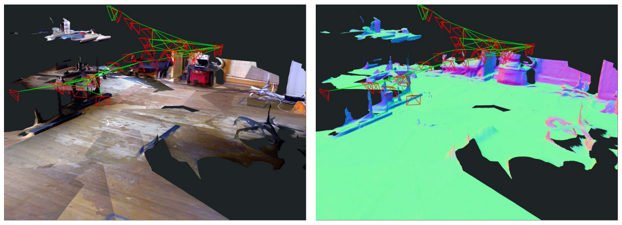


Figure 4. Reconstruction and trajectory TUM fr1/floor sequence.

4. Results

We evaluate our system on a wide range of real-world datasets. For localisation, we evaluate monocular SLAM on TUM RGB-D [37], 7-Scenes [35], ETH3D-SLAM [33], and EuRoC [3], all under monocular RGB setting. For geometry evaluation, we use the EuRoC Vicon room sequences as it provides 3D structure scan ground truth, as well as 7-Scenes since it has depth camera measurements.

We run our system on a desktop with Intel Core i9 12900K 3.50GHz and a single NVIDIA GeForce RTX 4090. As our system runs at roughly 15 FPS, we subsample every 2 frames of the datasets to simulate real-time performance. Note that we use the full resolution outputs from MASt3R, which resizes the largest dimension to size 512.

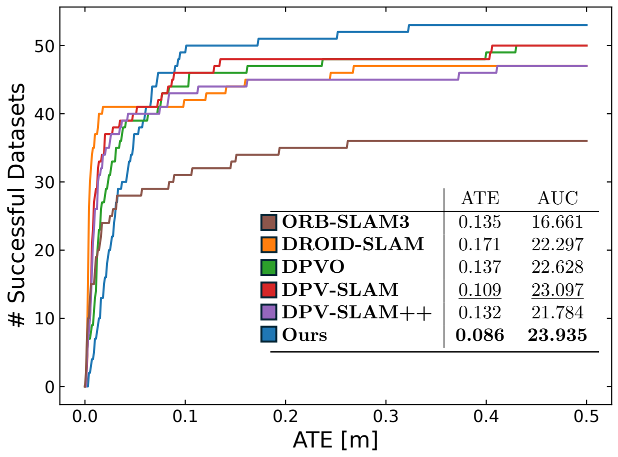
4.1. Camera Pose Estimation

For all datasets, we report the RMSE of the absolute trajectory error (ATE) in metres. Since all systems are monocular, we perform scaled trajectory alignment. We denote our system without known calibration as Ours*.

TUM RGB-D: On the TUM dataset, we demonstrate state-of-the-art trajectory error when leveraging calibration information as shown in Tab. 1. Many of the previously best performing algorithms, such as DROID-SLAM, DPV-SLAM, and GO-SLAM, build on the foundational matching and end-to-end system proposed by DROID-SLAM. In contrast, we propose a unique system that takes an off-the-shelf

Table 2. Absolute trajectory error (ATE (m)) on 7-Scenes [35].

	chess	fire	heads	office	pumpkin	kitchen	stairs	avg
NICER-SLAM	**0.033**	0.069	0.042	0.108	0.200	**0.039**	0.108	0.086
DROID-SLAM	<u>0.036</u>	<u>0.027</u>	<u>0.025</u>	**0.066**	0.127	<u>0.040</u>	<u>0.026</u>	<u>0.049</u>
Ours	0.053	**0.025**	**0.015**	<u>0.097</u>	**0.088**	0.041	**0.011**	**0.047**
Ours*	0.063	0.046	0.029	0.103	<u>0.114</u>	0.074	0.032	0.066



	ATE	AUC
🟫 **ORB-SLAM3**	0.135	16.661
🟧 **DROID-SLAM**	0.171	22.297
🟩 **DPVO**	0.137	22.628
🟥 **DPV-SLAM**	<u>0.109</u>	<u>23.097</u>
🟪 **DPV-SLAM++**	0.132	21.784
🟦 **Ours**	**0.086**	**23.935**

Figure 5. Number of successful trajectories below ATE threshold on ETH3D-SLAM (train) benchmark. The corresponding table shows the mean ATE across completed sequences, as well as the AUC up to the threshold.

two-view geometric prior and show that it can outperform all other systems while operating in real-time. Furthermore, our uncalibrated system significantly outperforms a baseline, which we denote DROID-SLAM*, that calibrates the intrinsics using GeoCalib [47] on the first image of a sequence, which is then used by DROID-SLAM. We achieve this without assuming a fixed camera model across the entire sequence, and demonstrate the value of 3D priors for dense uncalibrated SLAM over priors that solve subproblems. Our uncalibrated SLAM results are also comparable to results from other recent learned techniques such as DPV-SLAM with known calibration.

7-Scenes: We use the same sequences for evaluation following NICER-SLAM as shown in Tab. 2. Our calibrated system outperforms both NICER-SLAM [57] and DROID-SLAM. Furthermore, our real-time uncalibrated system using a single 3D reconstruction prior outperforms NICER-SLAM, which uses multiple priors in depth, normal, and optical flow networks and runs offline.

ETH3D-SLAM: Due to its difficulty, ETH3D-SLAM has only been evaluated for RGB-D methods. Since the ATE thresholds for the official private evaluation are too strict for monocular methods, we evaluate several state-of-the-art monocular systems on the train sequences and generate the ATE curves. The dataset contains sequences with fast camera motion, hence, for all methods, we do not subsample the

t

Table 3. Reconstruction Evaluation on 7-Scenes and EuRoC with all metrics in metres.

7-scenes	ATE	Accuracy	Completion	Chamfer
DROID-SLAM	<u>0.049</u>	0.141	0.048	0.094
Spann3R @20	N/A	<u>0.069</u>	0.047	<u>0.058</u>
Spann3R @2	N/A	0.124	**0.043**	0.084
Ours	**0.047**	0.089	0.085	0.087
Ours*	0.066	**0.068**	<u>0.045</u>	**0.056**
EuRoC	ATE	Accuracy	Completion	Chamfer
DROID-SLAM	**0.022**	0.173	**0.061**	0.117
Ours	<u>0.041</u>	**0.099**	<u>0.071</u>	**0.085**
Ours*	0.164	<u>0.108</u>	0.072	<u>0.090</u>

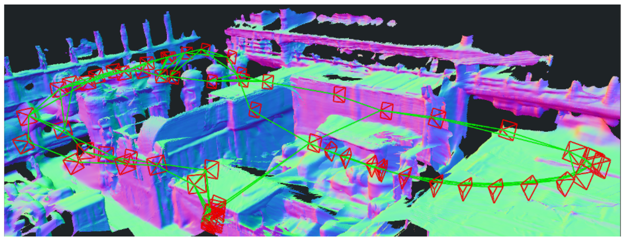


Figure 6. Reconstruction on EuRoC Machine Hall 04.

frames. While other methods can have more precise trajectories, our method has a longer tail in terms of robustness, resulting in both the best ATE and area-under-curve (AUC).

EuRoC: We report the average ATE across all 11 EuRoC sequences in Tab. 3. For the uncalibrated case, we found that the distortion was too significant as MASt3R was not yet trained on such camera models, so we undistorted the images but did not give calibration to the rest of the pipeline. In general, our system is outperformed by DROID-SLAM, but it explicitly augments its training with 10% greyscale images. However, 0.041m ATE is still very accurate, and from the comparisons in [21], all outperforming methods build on top of the foundation from DROID-SLAM, while we present a novel method using a 3D reconstruction prior.

4.2. Dense Geometry Evaluation

We evaluate our geometry against DROID-SLAM and Spann3R [48] on the EuRoC Vicon room sequences and 7-Scenes seq-01. For EuRoC, the alignment between the reference and the estimated point cloud is obtained by aligning the estimated trajectory against the Vicon trajectory. Note, that this setup favours DROID-SLAM which obtains lower trajectory error. For 7-Scenes, we backproject the depth images using poses provided by the dataset to create the reference point cloud. It is then aligned to the estimated point cloud using ICP as the extrinsic calibration between RGB and depth sensor is not provided.

We report the RMSE for accuracy, which is defined as the distance between each estimated point and its nearest

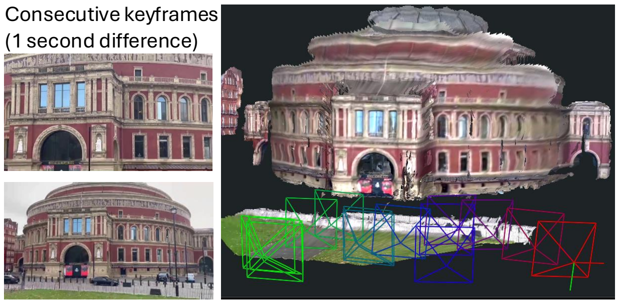


Figure 7. Dense uncalibrated SLAM with extreme zoom changes shown by two consecutive keyframes for an outdoor scene.

reference point, and completion, the distance between each reference point and its nearest estimated point. Both metrics are calculated with a maximum distance threshold of 0.5m and averaged across all sequences. We also report Chamfer Distance, the average of the two metrics.

Tab. 3 summarises the geometry evaluation on 7-Scenes and EuRoC. For 7-Scenes, both our method with and without calibration and Spann3R achieve more accurate reconstruction compared to DROID-SLAM, highlighting the advantage of the 3D prior. We run Spann3R under two different settings. In one, a keyframe is taken every 20 images and in the other every 2 images. The discrepancy in the two settings shows the challenges test-time optimisation-free approaches face to generalise. Ours without calibration performs the best in both Accuracy and Chamfer distance. This can be attributed to the fact that the intrinsic calibration 7-Scenes provides is the default factory calibration.

For EuRoC, Spann3R struggles as the sequences are not object-centric and thus is excluded. As summarised in Tab. 3, although DROID-SLAM outperforms our method in terms of ATE, our method with/without calibration obtains better geometry. DROID-SLAM obtains higher completion as it estimates a large number of noisy points which surround the reference point cloud, but our method has significantly better accuracy. It is interesting to note that our uncalibrated system has a noticeably larger ATE, but still outperforms DROID-SLAM in Chamfer distance.

4.3. Qualitative Results

Fig. 1 shows a reconstruction of the challenging Burghers sequence which has few matchable features on the specular figures. We show examples of pose estimation and dense reconstructions for TUM in Fig. 4 and for EuRoC in Fig. 6. Furthermore, we show an example with extreme zoom changes between consecutive keyframes in Fig. 7.

4.4. Ablation Studies

In Tab. 4, the ray error formulation for uncalibrated tracking and backend optimisation significantly improves performance over using the 3D point error and mitigates effects of inaccurate pointmap predictions. In Tab. 5, we test different methods for updating the canonical pointmap and report the average ATE across TUM, 7-Scenes, and EuRoC. Selecting the most recent or the latest point result in lack of sufficient baseline to resolve geometry or incur drift, respectively. Given calibration, weighted fusion performs on par with selecting the full pointmap with the highest median confidence, but it achieves the lowest ATE without calibration and improves the ATE on EuRoC by 1.3cm, indicating that fusing over camera models is significant.

We compare matching techniques in Tab. 6. In general, our parallel projective matching with feature refinement achieves the best accuracy with significantly faster runtime. Performing MASt3R matching over all pixels takes around 2 seconds, while our matching takes 2ms and makes the entire system FPS nearly 40x faster. Please refer to the supplementary for a full runtime analysis of the system.

Table 4. ATE (m) for point or ray error ablation.

	TUM	7-Scenes	EuRoC	avg
Point	0.092	0.084	0.290	0.155
Ray	**0.060**	**0.066**	**0.164**	**0.097**

Table 5. ATE (m) pointmap fusion ablation.

	w/o calib	w/ calib
Recent	0.207	0.160
First	0.114	0.059
Median conf	0.102	**0.039**
Weighted fusion	**0.097**	**0.039**

Table 6. Comparison of accuracy and runtime for different matching methods.

	ATE w/ calib	ATE w/o calib	Matching time (ms)	System FPS
k-d tree	0.061	0.115	40	8.8
MASt3R	<u>0.042</u>	0.098	2000	0.4
Ours w/o features	0.062	**0.092**	**0.5**	**15.1**
Ours w/ features	**0.039**	<u>0.097</u>	<u>2</u>	<u>14.9</u>

5. Limitations and Future Work

While we can estimate accurate geometry by filtering pointmaps in the frontend, we do not currently refine all geometry in the full global optimisation. While DROID-SLAM optimises per-pixel depth via bundle adjustment, this framework permits incoherent geometry. A method that can make pointmaps globally consistent in 3D while retaining the coherence of the original MASt3R predictions all in *real-time* would be an interesting direction for future work.

Since MASt3R is only trained on images with pinhole images, its geometry predictions degrade with increasing distortion. However, in the future, models will be trained on a variety of camera models and will be compatible with our framework that never assumes a parametric camera model. Furthermore, using the decoder at full resolution is currently a bottleneck, especially for low-latency tracking and checking loop closure candidates. Improving network throughout will benefit the total system efficiency.

6. Conclusion

We present a real-time dense SLAM system based on MASt3R that handles in-the-wild videos and achieves state-of-the-art performance. Much of the recent progress in SLAM has followed the contributions of DROID-SLAM, which trains an end-to-end framework that solves for poses and geometry from a flow update. We take a different approach by building a system around an off-the-shelf geometric prior that achieves comparable pose estimation for the first time, while also providing consistent dense geometry.

References

[1] Gwangbin Bae and Andrew J. Davison. Rethinking inductive biases for surface normal estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 2

[2] Michael Bloesch, Jan Czarnowski, Ronald Clark, Stefan Leutenegger, and Andrew J. Davison. CodeSLAM - learning a compact, optimisable representation for dense visual SLAM. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[3] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W. Achtelik, and Roland Siegwart. The EuRoC micro aerial vehicle datasets. *International Journal of Robotics Research (IJRR)*, 35(10): 1157–1163, 2016. 6, 15

[4] Carlos Campos, Richard Elvira, Juan J. Gómez Rodríguez, José M. M. Montiel, and Juan D. Tardós. ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM. *IEEE Transactions on Robotics (T-RO)*, 2021. 6

[5] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of SIGGRAPH*, page 303–312, 1996. 5

[6] Jan Czarnowski, Tristan Laidlow, Ronald Clark, and Andrew J. Davison. DeepFactors: Real-time probabilistic dense monocular SLAM. *IEEE Robotics and Automation Letters (RA-L)*, 2020. 2, 6, 15

[7] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2007. 2

[8] Eric Dexheimer and Andrew J. Davison. Learning a depth covariance function. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2

[9] Eric Dexheimer and Andrew J. Davison. COMO: Compact mapping and odometry. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024. 2

[10] Bardienus Duisterhof, Lojze Zust, Philippe Weinzaepfel, Vincent Leroy, Yohann Cabon, and Jerome Revaud. MASt3R-SfM: a fully-integrated solution for unconstrained structure-from-motion. *arXiv preprint arXiv:2409.19152*, 2024. 2, 3, 4, 5

[11] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Neural Information Processing Systems (NeurIPS)*, 2014. 2

[12] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 2

[13] Linyi Jin, Jianming Zhang, Yannick Hold-Geoffroy, Oliver Wang, Kevin Matzen, Matthew Sticha, and David F. Fouhey. Perspective fields for single image camera calibration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2

[14] Bingxin Ke, Anton Obukhov, Shengyu Huang, Nando Metzger, Rodrigo Caye Daudt, and Konrad Schindler. Repurposing diffusion-based image generators for monocular depth estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 2

[15] Nikhil Keetha, Jay Karhade, Krishna Murthy Jatavallabhula, Gengshan Yang, Sebastian Scherer, Deva Ramanan, and Jonathon Luiten. SplaTAM: Splat, track & map 3D Gaussians for dense RGB-D SLAM. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 2

[16] Nima Keivan and Gabe Sibley. Constant-time monocular self-calibration. In *2014 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2014. 2

[17] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 2023. 2

[18] Georg Klein and David Murray. Parallel tracking and mapping for small AR workspaces. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2007. 2

[19] Lukas Koestler, Nan Yang, Niclas Zeller, and Daniel Cremers. TANDEM: Tracking and dense mapping in real-time using deep multi-view stereo. In *Conference on Robot Learning (CoRL)*, 2022. 2

[20] Vincent Leroy, Yohann Cabon, and Jerome Revaud. Grounding image matching in 3D with MASt3R. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024. 1, 2, 3

[21] Lahav Lipson, Zachary Teed, and Jia Deng. Deep patch visual SLAM. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024. 6, 7, 14, 15

[22] Hidenobu Matsuki, Riku Murai, Paul H. J. Kelly, and Andrew J. Davison. Gaussian splatting SLAM. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 2

[23] Kirill Mazur, Gwangbin Bae, and Andrew J. Davison. SuperPrimitive: Scene reconstruction at a primitive level. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 2

[24] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 2

[25] Raúl Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics (T-RO)*, 31(5):1147–1163, 2015. 2

[26] Zak Murez, Tarrence van As, James Bartolozzi, Ayan Sinha, Vijay Badrinarayanan, and Andrew Rabinovich. Atlas: End-to-end 3D scene reconstruction from posed images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 2

[27] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2011. 5

[28] Richard. A. Newcombe, Steven J. Lovegrove, and Andrew J. Davison. DTAM: Dense tracking and mapping in real-time. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011. 2

[29] Linfei Pan, Daniel Barath, Marc Pollefeys, and Johannes Lutz Schönberger. Global structure-from-motion revisited. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024. 5

[30] Rene Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2022. 2

[31] Dennis Rosebrock and Friedrich M. Wahl. Generic camera calibration and modeling using spline surfaces. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, 2012. 3

[32] Mohamed Sayed, John Gibson, Jamie Watson, Victor Prisacariu, Michael Firman, and Clément Godard. SimpleRecon: 3D reconstruction without 3D convolutions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022. 2

[33] Thomas Schöps, Torsten Sattler, and Marc Pollefeys. BAD SLAM: Bundle adjusted direct RGB-D SLAM. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 6

[34] Thomas Schöps, Viktor Larsson, Marc Pollefeys, and Torsten Sattler. Why having 10,000 parameters in your camera model is better than twelve. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 3, 4

[35] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. 6, 7

[36] Joan Sola, Jeremie Deray, and Dinesh Atchuthan. A micro lie theory for state estimation in robotics. *arXiv preprint arXiv:1812.01537*, 2018. 3, 12

[37] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2012. 6

[38] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew J. Davison. iMAP: Implicit mapping and positioning in real-time. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021. 2

[39] Jiaming Sun, Yiming Xie, Linghao Chen, Xiaowei Zhou, and Hujun Bao. NeuralRecon: Real-time coherent 3D reconstruction from monocular video. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[40] Chengzhou Tang and Ping Tan. BA-Net: Dense bundle adjustment networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019. 2

[41] Zachary Teed and Jia Deng. DeepV2D: Video to depth with differentiable structure from motion. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020. 6, 15

[42] Zachary Teed and Jia Deng. RAFT: Recurrent all-pairs field transforms for optical flow. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 2

[43] Zachary Teed and Jia Deng. Tangent space backpropagation for 3D transformation groups. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 3, 12

[44] Zachary Teed and Jia Deng. DROID-SLAM: Deep visual SLAM for monocular, stereo, and RGB-D cameras. In *Neural Information Processing Systems (NeurIPS)*, 2021. 2, 6, 14, 15

[45] Giorgos Tolias, Yannis Avrithis, and Hervé Jégou. To aggregate or not to aggregate: Selective match kernels for image search. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2013. 5

[46] Giorgos Tolias, Tomas Jenicek, and Ondřej Chum. Learning and aggregating deep local descriptors for instance-level recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 5

[47] Alexander Veicht, Paul-Edouard Sarlin, Philipp Lindenberger, and Marc Pollefeys. GeoCalib: Single-image calibration with geometric optimization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024. 2, 6, 7

[48] Hengyi Wang and Lourdes Agapito. 3D reconstruction with spatial memory. *arXiv preprint arXiv:2408.16061*, 2024. 3, 7

[49] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. DUSt3R: Geometric 3D vision made easy. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 1, 5

[50] Xiaolong Wang, David Fouhey, and Abhinav Gupta. Designing deep networks for surface normal estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2

[51] Chi Yan, Delin Qu, Dan Xu, Bin Zhao, Zhigang Wang, Dong Wang, and Xuelong Li. GS-SLAM: Dense visual SLAM with 3D Gaussian splatting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 2

[52] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 2

[53] Youmin Zhang, Fabio Tosi, Stefano Mattoccia, and Matteo Poggi. GO-SLAM: Global optimization for consistent 3D instant reconstruction. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2023. 6, 15

[54] H. Zhou, B. Ummenhofer, and T. Brox. DeepTAM: Deep tracking and mapping. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. 2

[55] Qian-Yi Zhou and Vladlen Koltun. Dense scene reconstruction with points of interest. *ACM Transactions on Graphics*, 32(4), 2013. 1

[56] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R. Oswald, and Marc Pollefeys. NICE-SLAM: Neural implicit scalable encoding for SLAM. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[57] Zihan Zhu, Songyou Peng, Viktor Larsson, Zhaopeng Cui, Martin R Oswald, Andreas Geiger, and Marc Pollefeys. NICER-SLAM: Neural implicit scene encoding for RGB SLAM. In *Proceedings of the International Conference on 3D Vision (3DV)*, 2024. 2, 7, 14

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

Supplementary Material

7. 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 [36, 43].

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

$$\frac{\mathcal{D}f(\mathbf{T})}{\mathcal{D}\mathbf{T}} \triangleq \lim_{\boldsymbol{\tau}\to 0} \frac{f(\boldsymbol{\tau} \oplus \mathbf{T}) \ominus f(\mathbf{T})}{\boldsymbol{\tau}}, \tag{12}$$

$$= \lim_{\boldsymbol{\tau}\to 0} \frac{\mathrm{Log}\left(f\left(\mathrm{Exp}(\boldsymbol{\tau}) \circ \mathbf{T}\right) \circ f(\mathbf{T})^{-1}\right)}{\boldsymbol{\tau}}. \tag{13}$$

7.1. 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:

$$r_p = \mathbf{T}_{ij}\tilde{\mathbf{X}}^j_{j,n} - \tilde{\mathbf{X}}^i_{i,m}. \tag{14}$$

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 $\mathbf{T}_{ij}$:

$$\frac{\mathcal{D}r_p}{\mathcal{D}\mathbf{T}_{ij}} = \begin{bmatrix}\mathbf{I}_{3\times3} & -[\mathbf{x}]_\times & \mathbf{x}\end{bmatrix} \tag{15}$$

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

7.2. 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:

$$r_\psi = \psi\left(\mathbf{T}_{ij}\tilde{\mathbf{X}}^j_{j,n}\right) - \psi\left(\tilde{\mathbf{X}}^i_{i,m}\right). \tag{16}$$

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:

$$\frac{\mathcal{D}_\psi}{\mathcal{D}\mathbf{T}_{ij}} = \frac{\partial r_\psi}{\partial \mathbf{x}}\frac{\mathcal{D}\mathbf{x}}{\mathcal{D}\mathbf{T}_{ij}}. \tag{17}$$

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

$$\frac{\partial r_\psi}{\partial \mathbf{x}} = \frac{1}{d_{\mathbf{x}}}\left(\mathbf{I}_{3\times3} - \frac{\mathbf{x}\mathbf{x}^T}{d^2_{\mathbf{x}}}\right). \tag{18}$$

Using the chain rule with Eq. (15), the first term becomes Eq. (18) 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. (18) 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:

$$\frac{\mathcal{D}r_\psi}{\mathcal{D}\mathbf{T}_{ij}} = \begin{bmatrix}\frac{\partial r_\psi}{\partial \mathbf{x}} & -\frac{1}{d_{\mathbf{x}}}[\mathbf{x}]_\times & \mathbf{0}_{3\times1}\end{bmatrix}. \tag{19}$$

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:

$$r_d = d\left(\mathbf{T}_{ij}\tilde{\mathbf{X}}^j_{j,n}\right) - d\left(\tilde{\mathbf{X}}^i_{i,m}\right) \tag{20}$$

and its corresponding Jacobians are:

$$\frac{\partial r_d}{\partial \mathbf{x}} = \frac{\mathbf{x}^T}{d_{\mathbf{x}}}, \tag{21}$$

$$\frac{\mathcal{D}r_d}{\mathcal{D}\mathbf{T}_{ij}} = \begin{bmatrix}\frac{\mathbf{x}^T}{d_{\mathbf{x}}} & \mathbf{0}_{1\times3} & d_{\mathbf{x}}\end{bmatrix}. \tag{22}$$

7.3. 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:

$$r_\Pi = \Pi\left(\mathbf{T}_{ij}\tilde{\mathbf{X}}^j_{j,n}\right) - \mathbf{p}^i_{i,m}. \tag{23}$$

Using a pinhole camera model with calibration

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{24}$$

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

$$\frac{\partial r_\Pi}{\partial \mathbf{x}} = \frac{1}{z}\begin{bmatrix} f_x & 0 & -f_x\frac{x}{z} \\ 0 & f_y & -f_y\frac{y}{z}\end{bmatrix}. \tag{25}$$

We can then obtain $\frac{\mathcal{D}r_\Pi}{\mathcal{D}\mathbf{T}_{ij}}$ via the chain rule with Eq. (15). We also include a small error on the predicted and measured depth with similar motivation to Eq. (22) in cases of pure rotation. In the future, any parametric camera model and its corresponding Jacobian could be used here.

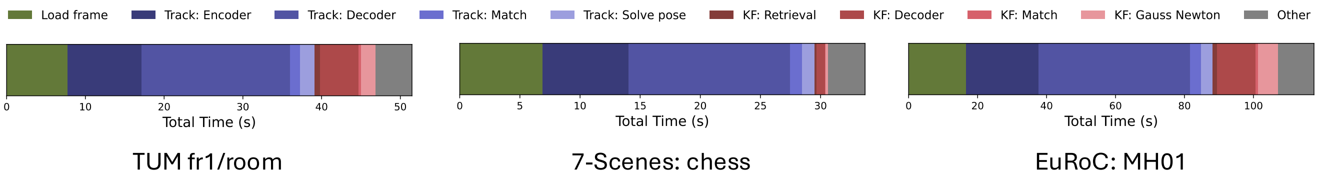


Figure 8. Total runtime in seconds for representative datasets showing cumulative time spent in significant components. The network encoder and decoder are the majority of the runtime at an average of 64% of the total runtime. Datasets with more loop closures like fr1/room and MH01 show more time spent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

Table 7. 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

7.4. 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

$$\frac{\mathcal{D}\mathbf{T}_{WC_i}^{-1}}{\mathcal{D}\mathbf{T}_{WC_i}} = -\mathrm{Ad}_{\mathbf{T}_{WC_i}^{-1}}, \tag{26}$$

$$\frac{\mathcal{D}\mathbf{T}_{ij}}{\mathcal{D}\mathbf{T}_{WC_i}^{-1}} = \mathbf{I}_{7\times 7}, \tag{27}$$

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

we can then solve for updates to each pose:

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

$$\frac{\mathcal{D}r_\psi}{\mathcal{D}\mathbf{T}_{WC_j}} = \frac{\mathcal{D}r_\psi}{\mathcal{D}\mathbf{T}_{ij}}\frac{\mathcal{D}\mathbf{T}_{ij}}{\mathcal{D}\mathbf{T}_{WC_j}} = \frac{\mathcal{D}r_\psi}{\mathcal{D}\mathbf{T}_{ij}}\mathrm{Ad}_{\mathbf{T}_{WC_i}^{-1}}. \tag{30}$$

8. 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.

9. 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. 7. 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.

10. Evaluation Setup

10.1. 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

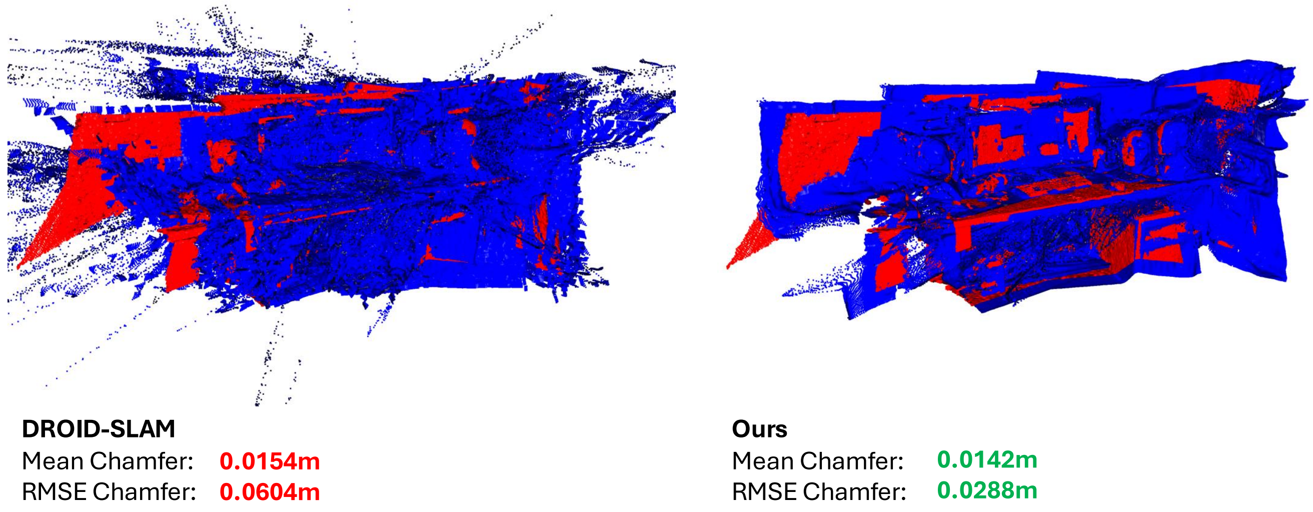


Figure 9. Reconstruction comparison on 7-Scenes heads, with red indicating the ground-truth point cloud and blue the estimated point cloud. While mean Chamfer distance does not significantly penalise inconsistent points, RMSE Chamfer is a better reflection of the quality of the geometry.

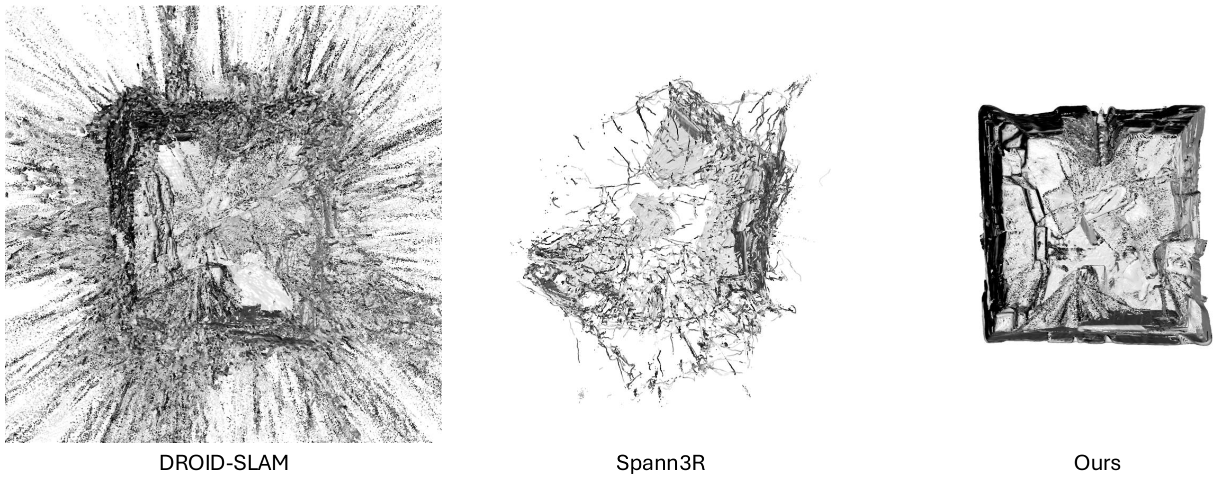


Figure 10. Reconstruction comparison on EuRoC V102.

SLAM [21], which also uses some results from DROID-SLAM [44]. For 7-Scenes, we include the results reported from NICER-SLAM [57]. For ETH3D, we ran all methods locally as the dataset was not previously attempted with monocular SLAM methods.

10.2. 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 8. 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 [41]	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 [21]	**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++ [21]	**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 [53]	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 [44]	**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

11. EuRoC Results

We summarise the average ATE for EuRoC in the main paper, and show the results for each sequence in Tab. 8. 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.