COMO: Compact Mapping and Odometry

Eric Dexheimer and Andrew J. Davison

Dyson Robotics Lab, Imperial College London
{e.dexheimer21, a.davison}@imperial.ac.uk

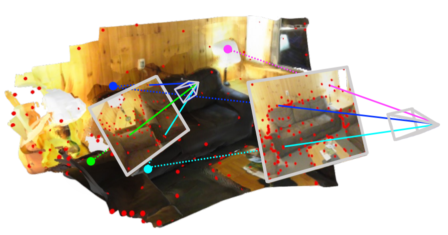
Abstract. We present COMO, a real-time monocular mapping and odometry system that encodes dense geometry via a compact set of 3D anchor points. Decoding anchor point projections into dense geometry via per-keyframe depth covariance functions guarantees that depth maps are joined together at visible anchor points. The representation enables joint optimization of camera poses and dense geometry, intrinsic 3D consistency, and efficient second-order inference. To maintain a compact yet expressive map, we introduce a frontend that leverages the covariance function for tracking and initializing potentially visually indistinct 3D points across frames. Altogether, we introduce a real-time system capable of estimating accurate poses and consistent geometry.

Keywords: SLAM · Multi-view geometry · 3D representations

1 Introduction

Achieving accurate and consistent poses and dense geometry from monocular images in real-time is a challenging yet essential undertaking to push forward state-of-the-art in robotics and augmented reality. Monocular cameras are the key to achieving low-cost, energy-efficient, and compact intelligent platforms. While images contain rich visual information, reconstructing the world is challenging due to the lack of direct geometric observations.

The ideal representation for real-time monocular visual odometry (VO) and simultaneous localization and mapping (SLAM) remains elusive. While sparse methods jointly optimize camera poses and a set of conditionally independent 3D points given poses, the map lacks consistent dense geometry for downstream tasks and pose estimation cannot benefit from all visual information. Dense SLAM uses and reconstruct all pixels, but the sheer number of variables relative to measurements renders joint optimization infeasible and inference ill-posed. Recently, learned priors over compact depth map representations, such as codes and sparse 2D depths, have enabled joint optimization. However, representations with 2D depth priors have thus far lagged in accuracy, as consistency between depth maps in 3D is not guaranteed. Volumetric representations, such as voxel grids and neural fields, directly enforce dense 3D consistency by construction, but cannot perform real-time joint optimization due to expensive rendering. Furthermore, compact, efficient, and expressive 3D priors for general scenes are not yet suitable. Achieving accurate and consistent 3D geometry in real-time would represent a significant advance for monocular SLAM.


(a) Point cloud reconstruction from the sliding-window monocular odometry defined by 240 points and 8 keyframes.

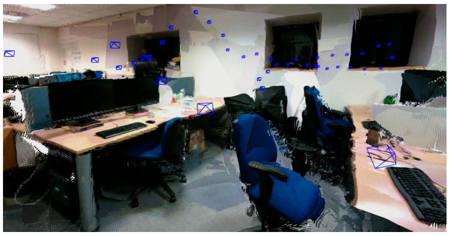

(b) Dense point cloud from 40 keyframes across a large office during live operation of a monocular camera.

Fig. 1: COMO encodes scene geometry via a compact set of 3D points and decodes points into dense geometry via per-keyframe depth covariance functions. The 3D points visualized in red anchor depth maps together from multiple views while the covariance function generates dense geometry by conditioning on sparse point projections.

In this work, we propose a representation that achieves the three desired properties: joint optimization of poses and dense geometry, intrinsic 3D consistency, and low-latency, real-time operation. We encode the scene as a compact set of 3D points, which are shared across frames. To decode into dense geometry, 3D points are first projected into keyframes. We then use image-conditioned depth covariance functions [9] per keyframe to condition on sparse depths and generate dense depth maps that intersect each corresponding 3D point. Depth maps are backprojected into 3D for calculating any-pixel photometric error. All steps are efficient and have analytic Jacobians for second-order optimization. Updates to dense geometry are propagated into our encoded scene representation and ensure suitable regularization for the ill-posed problem of dense reconstruction.

Maintaining a compact map in an incremental VO setting requires a compatible frontend. Keeping a small number of points benefits efficiency, while the distribution of points affects expressiveness. Feature-based systems focus on visually distinct features, but this often leads to many points on edges which are not suitable for modeling dense geometry. Therefore, we leverage depth covariance for determining anchor point visibility in new keyframes, actively initializing new 3D points, and encoding the current dense geometry into new 3D points.

Our system demonstrates improved robustness over methods lacking joint pose and dense geometry optimization, and significantly more accurate pose and geometry estimation compared to compact 2D representations. Altogether, we propose a real-time VO system that achieves robust and accurate odometry along with consistent dense geometry, with example reconstructions in Fig. 1.

In summary, the contributions of our work are:

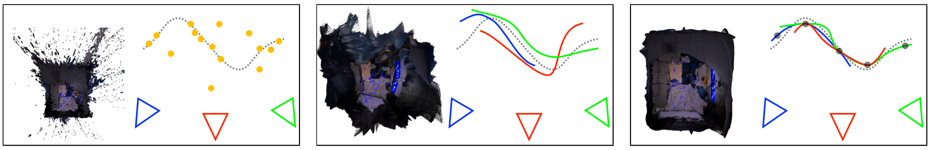
- An efficient representation of dense geometry encoded by a compact set of 3D points and decoded by depth covariance functions.
- A frontend for our compact map that leverages depth covariance for visibility, active initialization, and encoding dense geometry.
- A real-time monocular visual odometry and mapping system that produces accurate and consistent poses and dense geometry.

2 Related Work

Sparse vs. Dense Visual SLAM Sparse SLAM systems optimize 3D points that are conditionally independent given poses. Exploiting the sparsity inherent in the information form of structure-from-motion is the key to real-time algorithms with a significant number of points [16]. Modern sparse VO methods [10, 12, 22] and large-scale bundle adjustment [29] rely on the Schur complement which greatly reduces optimization complexity. Sparse systems lack dense reconstruction, which is useful for many downstream tasks. Thus, it is common to first perform pose estimation followed by dense mapping given poses and potentially sparse landmark estimates [18, 21, 23, 26]. MVS approaches estimate depth maps via a reference cost volume and multiple supporting frames, and have shown progress when integrated into a learning pipeline [17]. However, the sparse followed by dense paradigm has two major issues. Dense information is ignored for pose estimation, which can result in failure in many real-world scenarios, while the mapping step is heavily reliant on accurate pose estimation. Joint optimization of poses and dense geometry avoids the brittle two-step process and has the potential to achieve more accurate odometry and mapping [27].

Dense methods [11, 25] brought the promise of estimating poses and complete geometry simultaneously. However, joint optimization proved difficult due to the ill-posed nature of reconstructing all points. Hand-crafted geometry priors regularize the problem, but disrupt the sparsity and thus the tractability of joint optimization as discussed in [10]. In practice, these methods resort to alternating pose and geometry estimation or move away from second-order optimization. More recently, DROID-SLAM [35] reconstructs depths for all pixels, but similar to sparse systems, does so with no geometric correlation. This results in highly parallelized bundle adjustment, but results in many noisy points and no guarantee of consistent depths within a frame and across frames, as shown in Fig. 2a. All real-time state-of-the-art SLAM systems [6, 10, 35, 36] still rely on the Schur complement, and dense methods with geometric correlation have traditionally not demonstrated comparable pose accuracy, which also limits geometry estimation.

Learned Depth Priors for Compact Optimization To achieve consistent poses and geometry, monocular depth priors in SLAM focus on test-time optimization to avoid irreparable errors from single-view depth prediction. Therefore, depth priors in monocular SLAM often predict a subspace of potential depths given an image via a compact latent code [3, 7, 18], or a depth map basis [13, 34]. Even with this flexibility, depth maps are not guaranteed to be consistent across multiple views since latent variables live only in frames as shown in Fig. 2b. While per-pixel depth and keypoint reprojection losses may encourage 3D consistency [7], this may introduce bias and does not scale computationally since it requires per-pixel pairwise frame comparisons. Most importantly, it lacks intrinsic 3D consistency, and depth maps often produce a layering effect in 3D as in Fig. 2b. Predicting a view-based mesh topology permits optimizing depth vertices [4], but it is unclear how to move to a 3D topology.



(a) Dense bundle adjustment with 3D landmarks [35] **(b)** Per-frame dense depth priors via compact codes [7] **(c)** Ours: 3D points and depth covariance functions

Fig. 2: Reconstructions on Replica and geometry properties of different dense map representations. The dotted line represents the true surface. (a) Densely reconstructing a large number of conditionally independent 3D points given poses enables accurate pose estimation and many accurate points, but there is no guarantee of coherent geometry. (b) Depth priors produce smooth depth maps, but even with inter-frame consistency losses, can produce inconsistent geometry and bias preventing global consistency. (c) A compact set of 3D points and depth covariance functions anchor depth maps together, leading to consistent pose estimation and dense geometry.

In this work, we leverage the recently proposed learned depth covariance function [9]. While [9] introduces an odometry formulation, it only optimizes per-frame 2D inducing depths, which has the same limitations as per-frame codes, and lacks intrinsic 3D consistency. A key insight is that depth covariance permits sharing latent 3D points across frames, so that our representation lives in 3D rather than separate 2D image planes. While codes and bases are a fixed subspace that cannot guarantee a depth map passes through any set of 3D points, our covariance formulation guarantees that depth maps go through any desired 3D points as shown in Fig. 2c. Therefore, we can anchor depth maps together by construction for dense 3D consistency. We also exploit the covariance function for our frontend in tracking and initializing potentially non-visually distinct points.

Keyframe-based vs. Volumetric Maps Keyframe-based maps host dense depths in camera frames, which maintain flexible level-of-detail rooted in the camera resolution and focus only on surfaces. However, representing space using per-pixel depths cannot guarantee that inter-frame correspondences refer to the same 3D point. While pairwise depth constraints can mitigate inconsistency [7], these exhaustive per-pixel errors are expensive to compute, lack intrinsic consistency, and may introduce bias when balancing with other losses. Methods that project hosted depths into neighboring frames and predict flow updates [35] are asymmetric with no guarantee of cycle consistency.

Volumetric maps live in 3D space and unknown quantities are shared across frames. 3D consistency is achieved by construction but it is challenging to balance fidelity and efficiency. Voxel grids using backward sensor models [15, 24] assume known poses and independence between cells, and thus are unsuitable for efficient and consistent optimization. While forward models in voxel grids [37] and neural fields [19, 33] permit more consistent geometry with pose optimization in-the-loop, expensive rendering, a large number of correlated variables per ray, and alternating or first-order optimization limit its use in low-latency monocular VO and SLAM. In addition, volumetric methods are often subject to a resolution trade-off. Higher resolution permits greater detail, but requires more memory, greater runtime complexity, and increasing levels of noise.

Our 3D representation achieves the benefits of both frame-based and volumetric maps for odometry and mapping: the per-pixel resolution and efficiency of depth maps with the intrinsic 3D consistency of volumetric representations. The efficient compact-to-dense model permits joint optimization of poses and geometry via any-pixel constraints in real-time.

3 Compact Mapping Backend

The mapping backend maintains a compact set of 3D world points along with keyframes hosting per-pixel covariance function parameters. A dense world-centric point cloud is decoded from compact points and poses, which is refined by second-order minimization of photometric error. VO operates in a sliding-window fashion with a fixed number of keyframes. Fig. 3 provides an overview.

Quantities in the world frame and camera frame, such as points, are denoted by subscripts $\mathbf{P}_{\mathrm{W}}$ and $\mathbf{P}_{\mathrm{C}}$, respectively. Compact points are indexed by superscripts m, while dense points are indexed n, such as pixels $\mathbf{p}^m$ and $\mathbf{p}^n$. Stacked vector forms of sparse and dense variables use capital M and N, as in log-depth vectors $\mathbf{d}_{\mathrm{M}}$ and $\mathbf{d}_{\mathrm{N}}$. The notation $||\mathbf{r}||^2_{\Sigma}$ denotes Mahalanobis distance of residual vector $\mathbf{r}$ with measurement covariance matrix Σ.

3.1 Preliminaries: Depth Covariance Function

First, we introduce the depth covariance function from [9]. This covariance function models the distribution over all possible log-depth functions for any finite set of pixels via a Gaussian process (GP) [28]. First, a CNN takes in an RGB image and outputs a per-pixel feature map ϕ. Compared to [9], we use a zero-mean GP prior for simplicity, such that the distribution is defined only by the covariance function k that takes in two pixels and respective CNN features:

$$d(\mathbf{p}) \sim \mathcal{GP}\left(0,\, k\left([\mathbf{p};\phi(\mathbf{p})],[\mathbf{p}';\phi(\mathbf{p}')]\right)\right). \tag{1}$$

We found that the mean log-depth variable proposed in [9] is unnecessary, as the largest eigenvector of the covariance function corresponds to scale of the entire depth map. More information on the exact covariance function k is detailed in [9], but the key idea is that larger covariance is an indicator of similarity in log-depth. Since the covariance function takes in both pixel locations and CNN features, it is nonstationary which permits varying levels of smoothness and discontinuities conditioned on the image content. The zero-mean prior defines a planar depth prior in the absence of observations. To define a covariance matrix $\mathbf{K}$ for a set of pixels, the $(i^{\mathrm{th}}, j^{\mathrm{th}})$ entry of $\mathbf{K}$ is filled in by $k\left([\mathbf{p}^i;\phi(\mathbf{p}^i)],[\mathbf{p}^j;\phi(\mathbf{p}^j)]\right)$.

3.2 Compact-to-Dense Geometry

Given anchor point to keyframe correspondences, we generate a dense depth map given anchor points $\mathbf{P}^m_{\mathrm{W}}$, the pose of keyframe r with rotation $\mathbf{R}_{\mathrm{WC}_r}$ and

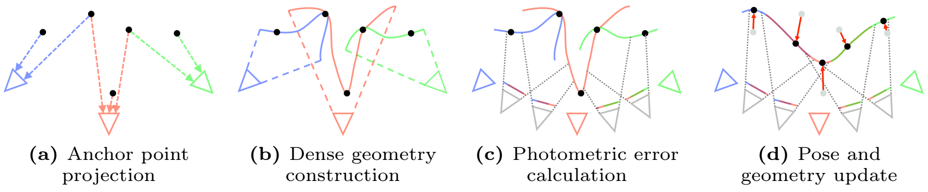

(a) Anchor point projection
(b) Dense geometry construction
(c) Photometric error calculation
(d) Pose and geometry update

Fig. 3: Overview of compact mapping framework. (a) Anchor points are projected to corresponding keyframes. (b) Keyframes decode into dense depth and backproject geometry to 3D. (c) Target frames enforce dense photo-consistency. (d) Optimizing dense alignment error leads to updated poses and geometry with greater 3D consistency.

translation $\mathbf{t}_{\mathrm{WC}_r}$, and the keyframe image. Compact points are transformed and projected to pixel coordinates $\mathbf{p}$ via camera projection π:

$$\mathbf{P}^m_{\mathrm{C}_r} = \mathbf{R}^T_{\mathrm{WC}_r}(\mathbf{P}^m_{\mathrm{W}} - \mathbf{t}_{\mathrm{WC}_r}), \qquad \mathbf{p}^m = \pi(\mathbf{P}^m_{\mathrm{C}_r}), \tag{2}$$

which is also illustrated in Fig. 3a. At this point, we create a dense log-depth map from log-depth observations $d^m \triangleq \log\left([\mathbf{P}^m_{\mathrm{C}_r}]_z\right)$ and projected pixel coordinates $\mathbf{p}^m$. Given an image, the covariances between pixels can be queried as in Section 3.1. The per-frame covariance matrix between all visible anchor point pixels is defined as $\mathbf{K}_{\mathrm{MM}}$, and the covariance between all image pixels and anchor point pixels as $\mathbf{K}_{\mathrm{NM}}$. We also stack all training d^m and test d^n log-depths into $\mathbf{d}_{\mathrm{M}}$ and $\mathbf{d}_{\mathrm{N}}$, respectively. The GP formulation allows efficient dense log-depth prediction via the linear Gaussian conditioning equation:

$$\mathbf{d}_{\mathrm{N}} = \mathbf{K}_{\mathrm{NM}} \left(\mathbf{K}_{\mathrm{MM}}\right)^{-1} \mathbf{d}_{\mathrm{M}}. \tag{3}$$

The GP guarantees that each dense log-depth map goes through the anchor points, creating a consistent surface across multiple views. Next, we backproject points for each pixel and transform to world coordinates:

$$\mathbf{P}^n_{\mathrm{W}} = \mathbf{R}_{\mathrm{WC}_r}\pi^{-1}(\mathbf{p}^n, e^{d^n}) + \mathbf{t}_{\mathrm{WC}_r}, \tag{4}$$

where n indexes a query pixel. Note that each keyframe generates a dense depth map via its observed anchor points and covariance parameter feature maps as in Fig. 3b. This compact-to-dense formulation only requires transformations, camera projection, and linear GP conditioning. Therefore, we can efficiently calculate analytical Jacobians for dense per-pixel constraints with respect to poses and anchor points. We show the chain rule for these steps in Section A.1.

In practice, we ignore the Jacobians with respect to pixel coordinates when constructing $\mathbf{K}_{\mathrm{MM}}$ and $\mathbf{K}_{\mathrm{NM}}$ and calculate the covariance matrices once at initialization. In addition to being more efficient, this ensures that depth maps do not undergo unstable changes if a point moves across a depth discontinuity. If an anchor point correspondence moves behind a camera during optimization, we reinitialize it to the median depth of the keyframe.

3.3 Photometric Residuals

While only keyframes are used to create dense geometry, additional small baseline support frames are included to improve convergence. We define a set of edges $\mathcal{E}$ for photometric error that consists of temporally adjacent keyframe-keyframe pairs, and support frames to the two nearest keyframes in time. Given a target image I_t, which may be either a keyframe or support frame, with pose $\mathbf{T}_{\mathrm{WC}_t}$, we first transform world points from keyframe view r and project:

$$\mathbf{p}_t^n = \pi(\mathbf{R}_{\mathrm{WC}_t}^T(\mathbf{P}_{\mathrm{W}}^n - \mathbf{t}_{\mathrm{WC}_t})). \tag{5}$$

Given these projective correspondences and accounting for exposure and global illumination changes via affine brightness parameters, a and b, similar to [10], we calculate the photometric error across pixels and frame edges:

$$E = \sum_{r,t \in \mathcal{E}} \sum_{n} ||\mathbf{r}_{r,t}^n||_{\sigma_{\mathbf{r}}^2 I}^2, \qquad \mathbf{r}_{r,t}^n = I_t(\mathbf{p}_t^n) + b_t - \left(\frac{e^{-a_r}}{e^{-a_t}} I_r(\mathbf{p}_r^n) + b_r\right), \tag{6}$$

which is illustrated in Fig. 3c. These constraints depend on each frame's pose and affine parameters, and the anchor points viewed by the reference keyframe. For robustness, we perform iteratively reweighted least squares (IRLS) with a Huber cost function and set the photometric standard deviation $\sigma_{\mathbf{r}}$ to be proportional to the median absolute residual [1] to handle diverse scenes with different error scales. Compared to RGB residuals, we found that grayscale images were more efficient and similarly robust. The Jacobian chain rule for photometric residuals is in Section A.2.

3.4 Additional Constraints

While photometric error is the only data term, we introduce additional priors to remove ambiguities. Priors on the oldest keyframe's pose and affine brightness parameters remove gauge freedom in the sliding-window. After removing a keyframe from the window, we include priors on its still-observed landmarks to ensure global consistency as a simpler alternative to first-estimate Jacobians [10].

Since keyframes may have regions unobserved by other frames, which causes anchor points to lack photometric constraints, we include two geometry priors. The first is a weak prior on anchor points to match the log median depth s of the keyframe it was first observed in. The second is a GP prior $||\mathbf{d}_{\mathrm{M}} - s||_{\mathbf{K}_{\mathrm{MM}}}^2$ on every keyframe to regularize anchor points that have large correlation, which gives improved estimates for underconstrained anchor points lying on surfaces with well-constrained ones, such as on textureless walls.

During optimization, there is ambiguity in whether an anchor point changes depth or moves laterally along a surface, such as when viewing a plane at an angle. Thus, we include a pixel prior encouraging the projection of the point in its initializing keyframe not to change from its first observation, which defines a ray-surface intersection. As mentioned previously, this is more efficient than calculating Jacobians through Sec. 3.2 and more stable since GP inducing point optimization has little benefit when points start in suitable locations [5] as ensured by our frontend in Section 4.

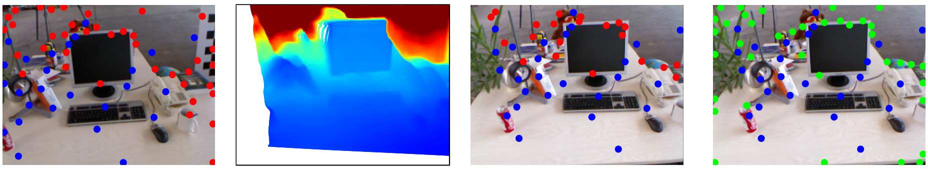

(a) Anchor points visible in KF #1 **(b)** KF #1 depths projected to KF #2 **(c)** Anchor point matches in KF #2 **(d)** Newly initialized points in KF #2

Fig. 4: Overview of visibility checks between two keyframes (KF). Matches are shown in blue, rejected matches in red, and newly initialized points in green. Note that occluded edges are rejected, while non-visually distinct points are often matched. New points are allocated to geometrically complex regions while the table is already well-represented.

3.5 Second-Order Optimization

We calculate analytical Jacobians for all constraints in the optimization. The compactness of our linear system permits real-time joint optimization of poses and correlated geometry that is infeasible for traditional dense VO. Stacking all Jacobians into $\mathbf{J}$, weights from IRLS and noise models into the diagonal matrix $\mathbf{W}$, and residuals into $\mathbf{r}$, we solve for parameter updates $\Delta\mathbf{x}$ shown in Fig. 3d via dense Cholesky factorization of the Gauss-Newton normal equations:

$$(\mathbf{J}^T\mathbf{W}\mathbf{J})\Delta\mathbf{x} = -\mathbf{J}^T\mathbf{W}\mathbf{r}. \tag{7}$$

Pose updates are minimally parameterized as Lie-algebra elements $\mathfrak{se}(3)$ as is standard in SLAM [30]. We found that the most time consuming step of backend optimization is accumulating geometry blocks into the Hessian $(\mathbf{J}^T\mathbf{W}\mathbf{J})$. This can be sped up by factoring out constant components of the Jacobians and reducing the dimensionality of the accumulation as detailed in Section A.4.

4 Visual Frontend

The visual frontend tracks the current frame with respect to the map, selects keyframes and support frames for the backend, determines 3D anchor point visibility in new keyframes, and initializes new 3D anchor points. Compared to feature-based systems, the ideal points for an expressive yet compact map may be visually indistinct, so we utilize depth covariance for robust correspondence.

4.1 Dense Photometric Tracking

Given the current estimate of the newest keyframe's depth, we optimize the pose and affine brightness parameters using grayscale image alignment in the IRLS framework [2]. For efficiency, we perform inverse compositional tracking and use a multi-scale image pyramid. After tracking, a new keyframe is added to the backend if there is either significant translation relative to the median scene depth or if the number of unique projected pixels in the tracked frame falls below a threshold to handle rotation and occlusion. Support frames are added via modified thresholds according to the desired frequency between keyframes.

4.2 Anchor Point Visibility

When a new keyframe is added, we perform explicit correspondence to 3D anchor points viewed in the last keyframe to achieve 3D consistency. Note that correspondence here means whether a point is visible to a frame, as there is no explicit constraint on its location in the image. Compared to feature-based methods [22], anchor points often lack visual distinctiveness, so matching via image information is problematic. Instead, we leverage the interpretability of the depth covariance function to determine visibility. In general, a correspondence should be compatible with all depth covariance functions that use it to generate dense geometry.

First, the 3D anchor points and dense depth map from the last keyframe, shown in Fig. 4a, are projected into the current frame, shown in Fig. 4b. Given the projected sparse and dense pixel coordinates, we may query covariance parameters $\phi(\mathbf{p})$ and construct $\mathbf{K}_{\mathrm{MM}}$ and $\mathbf{K}_{\mathrm{MN}}$ for the new keyframe. We leverage the linearity of the GP to compress dense log-depth observations into the sparse projected anchor point coordinates. Assuming conditional independence of log-depth observations $\mathbf{d}_n$ given the sparse log-depths $\mathbf{d}_m$, we can solve an efficient least-squares problem:

$$\min_{\mathbf{d}_{\mathrm{M}}} ||\mathbf{K}_{\mathrm{NM}} \left(\mathbf{K}_{\mathrm{MM}}\right)^{-1} \mathbf{d}_{\mathrm{M}} - \mathbf{d}_{\mathrm{N}}||^2_{\sigma_d^2} + ||\mathbf{d}_{\mathrm{M}}||^2_{\mathbf{K}_{\mathrm{MM}}}, \tag{8}$$

where the first conditional term ensures that $\mathbf{d}_{\mathrm{M}}$ fits the projected dense observations with standard deviation σ_d, while the second prior term regularizes sparse log-depths according to the GP training covariance $\mathbf{K}_{\mathrm{MM}}$. We perform a consistency check on the log-depth difference between the current 3D anchor points and those solved in Eq. 8, and prune points at depth discontinuities. This yields potential matches the are consistent between the two keyframes.

In practice, we set a maximum number of visible anchor points per-frame to allow efficient batched mapping operations on the GPU. If all points are matched, it could prevent new ones from being initialized and parts of the scene would lack capacity to adequately represent geometry. Thus, we perform pruning of matches so that we have a suitable distribution of points before initializing new anchors. Similar to the GP conditional mean from Eq. 3, we may also obtain conditional variances of all pixels with respect to the first j pixels via:

$$\boldsymbol{\sigma}_j^2 = \operatorname{diag}\left[\mathbf{K}_{\mathrm{NN}}\right] - \mathbf{K}_{\mathrm{NM}_{1:j}} \left(\mathbf{K}_{\mathrm{M}_{1:j}\mathrm{M}_{1:j}}\right)^{-1} \mathbf{K}_{\mathrm{M}_{1:j}\mathrm{N}}. \tag{9}$$

We perform active sampling via conditional variance reduction (CVR) [14, 20], where at the $(j+1)^{\mathrm{th}}$ iteration, the pixel in domain P with the largest variance is selected $\mathbf{p}_{j+1} = \arg\max_{\mathbf{p}\in P} \boldsymbol{\sigma}_j^2(\mathbf{p})$. We actively sample from the set of potential matches until a variance threshold is reached, which gives the final set of anchor point correspondences for the current keyframe. Compared to [9], we found that a minimum distance threshold between pixel locations and from the image border is required to avoid accumulating pixels near the same discontinuity. This is essential for improving depth estimates and maintaining a good set of candidate points for correspondence, as otherwise many points near edges may be occluded at the same time. We show the set of matches between two keyframes in Fig. 4c.

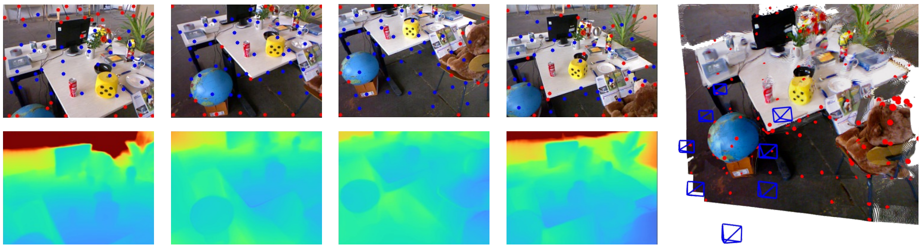


Fig. 5: Visualization of sliding-window odometry on TUM *fr2/desk*. Top images show tracked and newly initialized anchor point correspondence in blue and red, respectively. Depth maps are below, while the dense point cloud with 248 anchor points is shown on the right. Note the correspondence tracks on textureless surfaces, such as the table and plate, as well as the long-term correspondence on the cube, mug, and book.

4.3 Anchor Point Initialization

As the camera explores unobserved parts of the scene, new anchor points must be initialized. First, we again perform active sampling, but now across all pixels given the current anchor point locations from the previous section. This ensures new points will be initialized for parts of the scene that are unobserved, lack matches, or require additional geometric capacity, as shown in Fig. 4d.

Once the pixel locations of new anchor points are known, we initialize the log-depth and backproject into 3D. Defining the set of known log-depths from the anchor points $\mathbf{d}_{\mathrm{m}_1}$, the unknown log-depths of newly initialized points $\mathbf{d}_{\mathrm{m}_2}$, and all log-depths as $\mathbf{d}_{\mathrm{M}} = (\mathbf{d}_{\mathrm{m}_1}^T, \mathbf{d}_{\mathrm{m}_2}^T)^T$, we solve another least-squares problem:

$$\min_{\mathbf{d}_{\mathrm{m}_2}} ||\mathbf{K}_{\mathrm{NM}} \left(\mathbf{K}_{\mathrm{MM}}\right)^{-1} \mathbf{d}_{\mathrm{M}} - \mathbf{d}_{\mathrm{N}}||^2_{\sigma_d^2} + ||\mathbf{d}_{\mathrm{m}_2} - s||_{\sigma_s^2}, \tag{10}$$

where the first term promotes fitting reprojected dense depth observations given matched anchor log-depths and the new log-depth initializations, while the second term is a prior that new log-depths are close to the log median depth s of the previous keyframe. We found that using an isotropic prior with standard deviation σ_d, rather than $\mathbf{K}_{\mathrm{MM}}$, avoids bias for parts of the image that were not previously observed and would otherwise be underconstrained. We define σ_d as the standard deviation of the residuals from Eq. 8 to better calibrate the model. Given the new pixel locations, log-depth initializations, and the new keyframe's pose estimate, 3D points are initialized for the backend. A qualitative example of tracked anchor points, depth maps, and reconstruction is shown in Fig. 5.

5 Experiments

We evaluate both the pose and geometry estimation of our method on multiple datasets. For sparse methods, we compare against feature-based ORB-SLAM3

		office0	office1	office2	office3	office4	room0	room1	room2	mean
S	ORB-SLAM3 [6]	0.4	0.3	8.5	0.4	4.5	0.3	0.3	0.4	1.9
	DSO [10]	0.3	0.5	0.3	0.2	5.3	0.2	0.8	0.2	**1.0**
D	DeepFactors [7]	42.9	38.6	45.6	41.4	50.1	32.4	26.1	44.1	40.2
	DROID-VO [35]	5.3	4.1	6.5	11.2	7.1	9.0	4.8	7.0	6.9
	COMO-NC	2.9	5.0	7.4	6.1	8.1	4.4	4.5	3.7	5.3
	COMO	2.4	2.4	4.3	3.6	5.5	2.5	2.7	3.4	**3.4**

Table 1: Mean ATE (cm) over 5 runs for monocular VO on the Replica dataset. S and D in the first column refer to sparse and dense methods, respectively.

[6], direct DSO [10], and learning-based DPVO [36] in trajectory estimation. We also compare against a representative set of dense methods for trajectory and depth estimation. TANDEM [17] uses DSO for odometry, a learned MVS network for local mapping, and a globally fused TSDF for tracking. DeepFactors [7] is a full code-based SLAM system with photometric, geometric, and keypoint losses to ensure consistency across depth maps. DROID-VO [35] uses learned correspondence and flow updates in a traditional bundle adjustment framework, such that all depths are reconstructed but lack geometric correlation. DepthCov [9] optimizes sparse 2D depths under the depth covariance framework, and in a related manner, COMO-NC (no correspondence) is our full framework but no 3D points are tracked across frames. We disable loop closure for DeepFactors and DROID-SLAM to isolate the odometry and mapping performance.

5.1 Implementation Details

We use a fixed configuration across all three datasets to demonstrate the robustness of the system. We use the off-the-shelf depth covariance function [9] that was trained on the ScanNet training set. Our sliding window odometry uses 9 keyframes and 3 support frames between keyframes for a total of 24. Each keyframe views a maximum of 64 anchor points and initializes new points if it fails to track all from the previous keyframe. We use 256x192 images for all operations, and the mapping backend samples pixels with the highest gradient magnitude in 4x4 images patches for photometric error.

5.2 Trajectory Evaluation

Replica We first test our monocular odometry on the Replica [31] sequences recorded in [33], which provides a photorealistic environment with ground-truth geometry. As shown in Table 1, sparse methods without priors perform best, as conditions are favorable for SLAM: few image artifacts, no exposure changes, and sufficient baselines for multi-view geometry. Among methods with learned priors, COMO achieves the lowest ATE. The representation has little bias as compared to the code-based DeepFactors, which drifts despite photometric, geometric, and keypoint constraints. Sharing 3D points between frames in COMO shows significant improvement over COMO-NC which disables correspondence.
TUM The TUM RGBD dataset [32] is a challenging dataset for monocular VO due to significant motion blur, exposure changes, rolling shutter artifacts,

		fr1								fr2		fr3	
		360	desk	desk2	plant	room	rpy	teddy	xyz	xyz	desk	long	mean
S	ORB-SLAM3 [6]	X	2.0	X	11.8	X	5.6	X	1.0	0.5	1.3	1.7	X
	DSO [10]	X	27.2	66.0	6.0	58.6	X	X	3.8	0.3	2.2	9.9	X
	DPVO [36]	13.1	9.4	6.5	3.0	39.8	3.5	6.2	1.3	0.5	3.5	5.5	**8.4**
D	TANDEM [17]	X	4.3	33.7	X	X	4.9	43.1	2.4	0.3	2.0	8.3	X
	DeepFactors [7]	17.9	15.9	20.2	31.9	38.3	3.8	56.0	5.9	8.4	26.3	49.0	24.9
	DepthCov [9]	12.8	5.6	4.8	26.1	25.7	5.2	47.5	5.6	1.2	15.9	68.8	19.9
	DROID-VO [35]	15.7	5.2	11.1	6.0	33.4	3.2	19.1	5.6	10.7	7.9	7.3	11.4
	COMO-NC	16.1	4.2	10.9	19.3	28.6	5.2	68.7	4.1	0.7	8.8	46.8	19.4
	COMO	12.9	4.9	9.5	13.8	27.0	4.8	24.5	4.0	0.7	6.3	10.5	**10.8**

Table 2: Mean ATE (cm) over 5 runs for monocular VO on the TUM dataset. S and D in the first column refer to sparse and dense methods, respectively.

and heavy rotations. Results comparing our method against state-of-the-art VO methods on 11 sequences is shown in Table 2. For sparse methods, ORB-SLAM3 and DSO both lose tracking on multiple sequences, while DPVO, which uses learned context for tracking 64 sparse patches and performing bundle adjustment, performs best among all methods. In this work, we are focused on also reconstructing dense geometry, but it would be interesting to combine the learned features of DPVO with our 3D representation. For dense methods, TANDEM fails due to relying on DSO for odometry, which can still fail despite relying on the TSDF with MVS fusion for tracking. This demonstrates the need for joint optimization of poses and dense image information. Despite promoting consistency between neighboring depth maps, DeepFactors has relatively large error. Both DepthCov and COMO-NC are similar in terms of optimizing sparse depths in each frame, while leveraging a true 3D representation in COMO reduces error by almost 50%. COMO outperforms DROID-VO, which also uses learned features for matching and reconstructs all pixels from downsized images via sparse bundle adjustment. As the assumptions of brightness constancy are violated in many real-world datasets, learned features could further the accuracy of COMO, but it is interesting that our compact representation and simple photometric error can produce the lowest ATE of dense methods.

ScanNet Test Lastly, we evaluate the trajectory error on 18 diverse medium-length scenes from the ScanNet [8] test set. Sequences include homes, offices, schools, businesses, and outdoors. Methods that fail are assigned a maximum 100 cm error for averages. Sequences have challenging rotational motion, high image noise, and specular surfaces. We summarize the results in Fig. 6, which shows the number of sequences below varying ATE thresholds, while Table 3 contains summary statistics. COMO has the lowest mean ATE and area-under-the-curve (AUC). While DPVO has a higher number of trajectories under low ATE thresholds, it also shows high variance results on some sequences. Despite using photometric error that is often violated due to image noise and specular surfaces, our representation with 3D consistency proves valuable in challenging real-world data. An example reconstruction is shown in Fig. 1a.

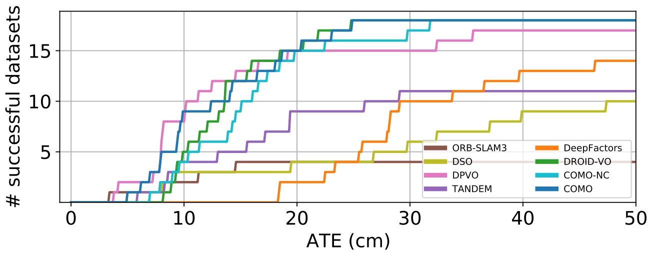


Fig. 6: Number of successful trajectories below ATE threshold for selected ScanNet test sequences.

Method	ATE	AUC
ORB-SLAM3	79.9	1.63
DSO	54.7	2.46
DPVO	15.1	6.34
TANDEM	41.7	3.79
DeepFactors	36.1	2.97
DROID-VO	13.9	6.51
COMO-NC	15.8	6.16
COMO	**13.0**	**6.67**

Table 3: ScanNet Mean ATE (cm) and AUC.

Method	Replica	ScanNet
TANDEM	N/A	0.325
DeepFactors	0.263	0.232
DROID-VO	1.129	1.389
COMO-NC	0.069	0.155
COMO	**0.046**	**0.128**

Table 4: Depth absolute relative error.

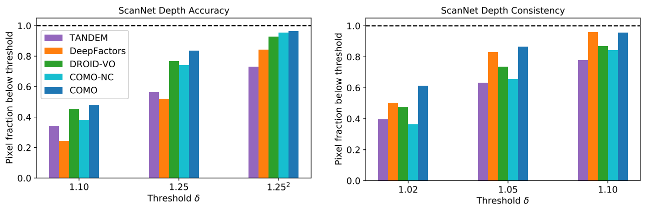


Fig. 7: ScanNet depth accuracy and consistency metrics for fraction of pixels falling below threshold δ.

5.3 Geometry Evaluation

Since we are interested in both consistent pose estimation and dense geometry, we evaluate dense depth predictions on Replica and ScanNet. To evaluate accuracy, we perform the same global scale alignment that is used to produce ATE for monocular methods and transform estimated poses and depth maps to be aligned to the metric ground-truth. Beyond global accuracy, we also measure consistency, which checks whether neighboring frames viewing the same 3D points have consistent depth estimates. Note that consistency does not care about the correctness of depth. To check valid pixels, we backproject ground-truth depth maps and project these points into neighboring frames, and check if the depths agree within 1cm. Then, for all valid 3D correspondences, we calculate metrics between pairs of estimated depth images. If N is the number of keyframes for that sequence, then N depth images are used for accuracy metrics, while $2(N-1)$ depth images are used for consistency since pairs require both directions.

For estimated depth $\hat{D}$ and ground-truth depth D we show the mean absolute relative error $(1/N)\sum_i |\hat{D}_i - D_i|/D_i$ across all sequences for Replica and ScanNet in Table 4. We do not evaluate TANDEM on Replica since it is used in the training data. COMO outperforms other methods due to its ability to jointly optimize dense, correlated geometry along with poses. Despite accurate pose estimation, DROID-VO has outliers with significant depth errors. To further evaluate the distribution of depth accuracy and remove the effect of large outliers, in Fig. 7, we use the δ threshold commonly used in depth estimation,

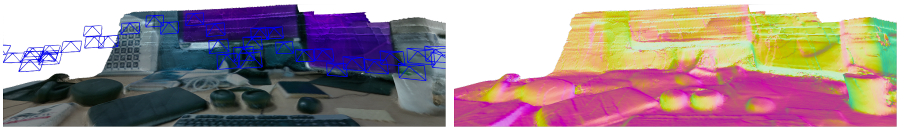


Fig. 8: Mesh and surface normals from TSDF fusion of a live desk sequence.

which measures what fraction of pixels satisfy $\max\left(\hat{D}/D, D/\hat{D}\right) < \delta$. Interestingly, accurate depth maps does not necessarily imply consistency. For example, DROID-VO has the second best accuracy behind COMO, while DeepFactors has the most consistent geometry other than COMO. DeepFactors uses explicit geometric and keypoint constraints in the optimization to ensure depth maps are close to each other. However, our compact mapping framework achieves both the best accuracy and consistency without explicit constraints.

5.4 Real-time System and Runtime Details

When evaluating our system, we use a sequential, single-threaded mode that runs at around 14 FPS with an RTX 3080. For live demos and reconstructions, we also develop a multiprocessing version that runs at 35-50 FPS as shown in the FPS counter in the top right of the supplementary video, with tracking flexible to be on either CPU or GPU. We show examples of reconstructions while operating a monocular camera in real-time. Fig. 1b displays a large-scale raw point cloud of an office, while Fig. 8 displays consistent TSDF fusion of a desk sequence.

Average times for tracking a frame, adding a keyframe (KF), and updating the map are 20.1, 50.4, and 39.8 ms. Keyframing is the bottleneck during fast motion, but 35-50 FPS in multiprocessing mode for normal motion is standard as shown in the video. For comparison, we measure the wall time of several methods. For the TUM *fr3/long* sequence of 1m27s, DeepFactors requires 6m52s, COMO single-thread 3m04s, DROID-VO 1m45s, and COMO multiprocessing 1m35s. We also believe further optimization is possible since almost all code is implemented in PyTorch and could be further customized in CUDA.

6 Conclusion

In this work, we present a compact representation of 3D anchor points and frame-based depth covariance functions to achieve efficient estimation of consistent poses and geometry in real-time. Future work includes training the depth covariance function on more diverse data and replacing photometric constraints with learned correspondence from DROID-VO to achieve the best of its learned correspondence with our representation. Lastly, we believe the compact point representation could be used in a full map-centric SLAM system with relocalization, which would allow continually improving geometry upon revisiting. With the advance of learned priors, exploiting geometric correlation is an interesting direction to achieve efficient, robust, and consistent algorithms.

7 Acknowledgements

Research presented in this paper has been supported by Dyson Technology Ltd. We would like to thank Riku Murai, Hidenobu Matsuki, Gwangbin Bae, and members of the Dyson Robotics Lab for insightful discussions and feedback.

References

1. Alismail, H.: Direct pose estimation and refinement. Ph.D. thesis, Robotics Institute, Carnegie Mellon University. (2016) 7
2. Baker, S., Matthews, I.: Lucas-Kanade 20 years on: A unifying framework. International Journal of Computer Vision (IJCV) (2004) 8
3. Bloesch, M., Czarnowski, J., Clark, R., Leutenegger, S., Davison, A.J.: CodeSLAM - learning a compact, optimisable representation for dense visual SLAM. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018) 3
4. Bloesch, M., Laidlow, T., Clark, R., Leutenegger, S., Davison, A.: Learning meshes for dense visual SLAM. In: Proceedings of the International Conference on Computer Vision (ICCV) (2019) 3
5. Burt, D.R., Rasmussen, C.E., van der Wilk, M.: Convergence of sparse variational inference in Gaussian processes regression. The Journal of Machine Learning Research (2020) 7
6. Campos, C., Elvira, R., Rodríguez, J.J.G., M. Montiel, J.M., D. Tardós, J.: ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM. IEEE Transactions on Robotics (T-RO) (2021) 3, 11, 12
7. Czarnowski, J., Laidlow, T., Clark, R., Davison, A.J.: DeepFactors: Real-time probabilistic dense monocular SLAM. IEEE Robotics and Automation Letters (RA-L) (2020) 3, 4, 11, 12, 5, 6
8. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017) 12
9. Dexheimer, E., Davison, A.J.: Learning a depth covariance function. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2023) 2, 4, 5, 9, 11, 12
10. Engel, J., Koltun, V., Cremers, D.: Direct sparse odometry. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) (2018) 3, 7, 11, 12, 1
11. Engel, J., Schöps, T., Cremers, D.: LSD-SLAM: Large-scale direct monocular SLAM. In: Proceedings of the European Conference on Computer Vision (ECCV) (2014) 3
12. Forster, C., Zhang, Z., Gassner, M., Werlberger, M., Scaramuzza, D.: SVO: Semidirect visual odometry for monocular and multicamera systems. IEEE Transactions on Robotics (T-RO) (2017) 3
13. Graham, B., Novotny, D.: RidgeSfM: Structure from motion via robust pairwise matching under depth uncertainty. In: Proceedings of the International Conference on 3D Vision (3DV) (2020) 3
14. Guestrin, C., Krause, A., Singh, A.P.: Near-optimal sensor placements in Gaussian processes. In: Proceedings of the International Conference on Machine Learning (ICML) (2005) 9

15. Hornung, A., Wurm, K.M., Bennewitz, M., Stachniss, C., Burgard, W.: OctoMap: An efficient probabilistic 3D mapping framework based on octrees. Autonomous Robots (2013) 4
16. Klein, G., Murray, D.: Parallel tracking and mapping for small AR workspaces. In: Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR) (2007) 3
17. Koestler, L., Yang, N., Zeller, N., Cremers, D.: TANDEM: Tracking and dense mapping in real-time using deep multi-view stereo. In: Conference on Robot Learning (CoRL) (2022) 3, 11, 12, 5, 6
18. Matsuki, H., Scona, R., Czarnowski, J., Davison, A.J.: CodeMapping: Real-time dense mapping for sparse SLAM using compact scene representations. IEEE Robotics and Automation Letters (RA-L) (2021) 3
19. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: NeRF: Representing scenes as neural radiance fields for view synthesis. In: Proceedings of the European Conference on Computer Vision (ECCV) (2020) 4
20. Moss, H.B., Ober, S.W., Picheny, V.: Inducing point allocation for sparse Gaussian processes in high-throughput bayesian optimisation. In: International Conference on Artificial Intelligence and Statistics (AISTATS) (2023) 9
21. Mur-Artal, R., Tardós, J.D.: Probabilistic semi-dense mapping from highly accurate feature-based monocular SLAM. In: Proceedings of Robotics: Science and Systems (RSS) (2015) 3
22. Mur-Artal, R., Tardós, J.D.: ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras. IEEE Transactions on Robotics (T-RO) (2017) 3, 9
23. Newcombe, R.A., Davison, A.J.: Live dense reconstruction with a single moving camera. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2010) 3
24. Newcombe, R.A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A.J., Kohi, P., Shotton, J., Hodges, S., Fitzgibbon, A.: KinectFusion: Real-time dense surface mapping and tracking. In: Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR) (2011) 4
25. Newcombe, R.A., Lovegrove, S.J., Davison, A.J.: DTAM: Dense tracking and mapping in real-time. In: Proceedings of the International Conference on Computer Vision (ICCV) (2011) 3
26. Pizzoli, M., Forster, C., Scaramuzza, D.: REMODE: Probabilistic, monocular dense reconstruction in real time. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) (2014) 3
27. Platinsky, L., Davison, A.J., Leutenegger, S.: Monocular visual odometry: Sparse joint optimisation or dense alternation? In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) (2017) 3
28. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. MIT Press (2005) 5
29. Schonberger, J.L., Frahm, J.M.: Structure-from-motion revisited. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016) 3
30. Sola, J., Deray, J., Atchuthan, D.: A micro lie theory for state estimation in robotics. arXiv preprint arXiv:1812.01537 (2018) 8
31. Straub, J., Whelan, T., Ma, L., Chen, Y., Wijmans, E., Green, S., Engel, J.J., Mur-Artal, R., Ren, C., Verma, S., Clarkson, A., Yan, M., Budge, B., Yan, Y., Pan, X., Yon, J., Zou, Y., Leon, K., Carter, N., Briales, J., Gillingham, T., Mueggler,

E., Pesqueira, L., Savva, M., Batra, D., Strasdat, H.M., Nardi, R.D., Goesele, M., Lovegrove, S., Newcombe, R.: The Replica dataset: A digital replica of indoor spaces. arXiv preprint arXiv:1906.05797 (2019) 11

32. Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A benchmark for the evaluation of RGB-D SLAM systems. In: Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS) (2012) 11
33. Sucar, E., Liu, S., Ortiz, J., Davison, A.J.: iMAP: Implicit mapping and positioning in real-time. In: Proceedings of the International Conference on Computer Vision (ICCV) (2021) 4, 11
34. Tang, C., Tan, P.: BA-Net: Dense bundle adjustment networks. In: Proceedings of the International Conference on Learning Representations (ICLR) (2019) 3
35. Teed, Z., Deng, J.: DROID-SLAM: Deep visual SLAM for monocular, stereo, and RGB-D cameras. In: Neural Information Processing Systems (NeurIPS) (2021) 3, 4, 11, 12, 5, 6
36. Teed, Z., Lipson, L., Deng, J.: Deep patch visual odometry. In: Neural Information Processing Systems (NeurIPS) (2023) 3, 11, 12
37. Thrun, S.: Learning occupancy grids with forward models. In: Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS) (2001) 4

A Analytical Jacobians

One of the main benefits of formulating compact-to-dense geometry in the manner presented is the capability to efficiently calculate analytical Jacobians, which is key to the second-order optimization. Here, we demonstrate the chain rule required, with each Jacobian being straightforward nonlinear functions that are common in SLAM, such as camera projection and point transformation, linear matrix multiplication from the GP, and element-wise operations, such as exponentiation and logarithm for converting between log-depth and depth.

A.1 Compact-to-Dense Geometry

Note that the GP conditional in Equation 3 means that each dense point n for a given image is dependent on all m anchor points visible to that image. We show the chain rule for a single dense geometry point n with respect to one visible anchor point m viewed in the keyframe

$$\frac{\partial \mathbf{P}^n_{\mathrm{W}}}{\partial \mathbf{P}^m_{\mathrm{W}}} = \frac{\partial \mathbf{P}^n_{\mathrm{W}}}{\partial \mathbf{P}^n_{\mathrm{C}_r}} \frac{\partial \mathbf{P}^n_{\mathrm{C}_r}}{\partial z^n} \frac{\partial z^n}{\partial d^n} \frac{\partial d^n}{\partial \mathbf{d}_{\mathrm{M}}} \frac{\partial \mathbf{d}_{\mathrm{M}}}{\partial z^m} \frac{\partial z^m}{\partial \mathbf{P}^m_{\mathrm{C}_r}} \frac{\partial \mathbf{P}^m_{\mathrm{C}_r}}{\partial \mathbf{P}^m_{\mathrm{W}}}, \tag{11}$$

but note that this can be calculated efficiently by broadcasting and using the full $\mathbf{K}_{\mathrm{NM}} \left(\mathbf{K}_{\mathrm{MM}}\right)^{-1}$ matrix. Similarly, we can calculate the Jacobian of a dense geometry point with respect to the keyframe pose it is generated by. However, note that there are two dependencies on the keyframe pose $\mathbf{T}_{\mathrm{WC}_r}$: one for transforming an anchor point into the camera frame, and the other for transforming a dense geometry point in the camera into the world frame so that other poses may use it in the photometric error calculation. Therefore, denoting the minimal 6 DoF parameterization of $\mathfrak{se}(3)$ with $\mathbf{T}_{\mathrm{WC}}$, and with an abuse of notation of a pose transforming a point as $\mathbf{P}_{\mathrm{W}} = \mathbf{T}_{\mathrm{WC}} \mathbf{P}_{\mathrm{C}}$, the Jacobian becomes:

$$\frac{\partial \mathbf{P}^n_{\mathrm{W}}}{\partial \mathbf{T}_{\mathrm{WC}_r}} = \frac{\partial \left(\mathbf{T}_{\mathrm{WC}_r} \mathbf{P}^n_{\mathrm{C}_r}\right)}{\partial \mathbf{T}_{\mathrm{WC}_r}} + \frac{\partial \left(\mathbf{T}_{\mathrm{WC}_r} \mathbf{P}^n_{\mathrm{C}_r}\right)}{\partial \mathbf{P}^n_{\mathrm{C}_r}} \frac{\partial \mathbf{P}^n_{\mathrm{C}_r}}{\partial \mathbf{T}_{\mathrm{WC}_r}} \tag{12}$$

$$= \frac{\partial \left(\mathbf{T}_{\mathrm{WC}_r} \mathbf{P}^n_{\mathrm{C}_r}\right)}{\partial \mathbf{T}_{\mathrm{WC}_r}} + \frac{\partial \left(\mathbf{T}_{\mathrm{WC}_r} \mathbf{P}^n_{\mathrm{C}_r}\right)}{\partial \mathbf{P}^n_{\mathrm{C}_r}} \frac{\partial \mathbf{P}^n_{\mathrm{C}_r}}{\partial z^n} \frac{\partial z^n}{\partial d^n} \frac{\partial d^n}{\partial \mathbf{d}_{\mathrm{M}}} \frac{\partial \mathbf{d}_{\mathrm{M}}}{\partial z^m} \frac{\partial z^m}{\partial \mathbf{P}^m_{\mathrm{C}_r}} \frac{\partial \mathbf{P}^m_{\mathrm{C}_r}}{\partial \mathbf{T}_{\mathrm{WC}_r}}. \tag{13}$$

One important design decision was whether to parameterize anchor points as 3D points in the world frame, or depths hosted in a reference camera frame. While the latter was used in DSO [10] to reduce the dimensionality of the unknown geometry variables, it would complicate our case. Using depth hosted in a camera frame includes both the target pose and the reference pose itself in the cost function, and Hessian blocks must now include the reference pose. In our case, since dense depth maps involve many anchor points, all reference poses associated with corresponding anchor points for a given keyframe would have dependencies.

This would significantly complicate Jacobian calculations, since potentially all poses would be involved in each error computation, rather than just reference keyframe poses and the targets for photometric error. By keeping the anchor points as world points and projecting, we avoid any host frame dependencies and maintain more efficient and less complex Jacobians. Furthermore, we also show in Section A.4 how to exploit our formulation to reduce the dimensionality of dense Hessian block accumulation.

A.2 Photometric Residuals

Image gradients are calculated via finite differences with a 3x3 Scharr filter and bilinearly interpolated along with the grayscale image values.

The Jacobians of the target image value with respect to an anchor point, the reference keyframe pose, and the target image pose are:

$$\frac{\partial (I_t(\mathbf{p}_t^n))}{\partial \mathbf{P}_{\mathrm{W}}^m} = \frac{\partial I_t}{\partial \mathbf{p}_t^n} \frac{\partial \mathbf{p}_t^n}{\partial \mathbf{P}_{\mathrm{C}_t}^n} \frac{\partial \mathbf{P}_{\mathrm{C}_t}^n}{\partial \mathbf{P}_{\mathrm{W}}^n} \frac{\partial \mathbf{P}_{\mathrm{W}}^n}{\partial \mathbf{P}_{\mathrm{W}}^m}, \tag{14}$$

$$\frac{\partial (I_t(\mathbf{p}_t^n))}{\partial \mathbf{T}_{\mathrm{WC}_r}} = \frac{\partial I_t}{\partial \mathbf{p}_t^n} \frac{\partial \mathbf{p}_t^n}{\partial \mathbf{P}_{\mathrm{C}_t}^n} \frac{\partial \mathbf{P}_{\mathrm{C}_t}^n}{\partial \mathbf{P}_{\mathrm{W}}^n} \frac{\partial \mathbf{P}_{\mathrm{W}}^n}{\partial \mathbf{T}_{\mathrm{WC}_r}}, \tag{15}$$

$$\frac{\partial (I_t(\mathbf{p}_t^n))}{\partial \mathbf{T}_{\mathrm{WC}_t}} = \frac{\partial I_t}{\partial \mathbf{p}_t^n} \frac{\partial \mathbf{p}_t^n}{\partial \mathbf{P}_{\mathrm{C}_t}^n} \frac{\partial \mathbf{P}_{\mathrm{C}_t}^n}{\partial \mathbf{T}_{\mathrm{WC}_t}}. \tag{16}$$

A.3 Additional Constraints

For the two depth priors, we require the Jacobian of the log-depth of each anchor point projection with respect to the anchor point and keyframe pose, which is

$$\frac{\partial d^m}{\partial \mathbf{P}_{\mathrm{W}}^m} = \frac{\partial d^m}{\partial z^m} \frac{\partial z^m}{\partial \mathbf{P}_{\mathrm{C}_r}^m} \frac{\partial \mathbf{P}_{\mathrm{C}_r}^m}{\partial \mathbf{P}_{\mathrm{W}}^m}, \tag{17}$$

$$\frac{\partial d^m}{\partial \mathbf{T}_{\mathrm{WC}_r}} = \frac{\partial d^m}{\partial z^m} \frac{\partial z^m}{\partial \mathbf{P}_{\mathrm{C}_r}^m} \frac{\partial \mathbf{P}_{\mathrm{C}_r}^m}{\partial \mathbf{T}_{\mathrm{WC}_r}}. \tag{18}$$

The pixel prior requires the Jacobian of the pixel projection of the anchor points with repsect to the anchor point and keyframe pose

$$\frac{\partial \mathbf{p}^m}{\partial \mathbf{P}_{\mathrm{W}}^m} = \frac{\partial \mathbf{p}^m}{\partial \mathbf{P}_{\mathrm{C}_r}^m} \frac{\partial \mathbf{P}_{\mathrm{C}_r}^m}{\partial \mathbf{P}_{\mathrm{W}}^m}, \tag{19}$$

$$\frac{\partial \mathbf{p}^m}{\partial \mathbf{T}_{\mathrm{WC}_r}} = \frac{\partial \mathbf{p}^m}{\partial \mathbf{P}_{\mathrm{C}_r}^m} \frac{\partial \mathbf{P}_{\mathrm{C}_r}^m}{\partial \mathbf{T}_{\mathrm{WC}_r}}, \tag{20}$$

which requires the camera projection Jacobian.

A.4 Geometry Hessian Block Trick

One of the most expensive steps in the mapping backend is accumulating the Hessian geometry blocks for the photometric error. For example, with the 64 points per-frame, this is requires a sum of 192x192 matrices over the number of pixels involved in the error. One interesting component of the geometry Jacobian is the the last term, which has a very simple form:

$$\frac{\partial \mathbf{P}^m_{\mathrm{C}_r}}{\partial \mathbf{P}^m_{\mathrm{W}}} = \mathbf{R}^T_{\mathrm{WC}_r}. \tag{21}$$

Note that for all anchor points in a given keyframe, this Jacobian is identical, as it is just the rotation from world to keyframe coordinates. Since this is the outer term when accumulating the Hessian, we may factor it out of the sum over all pixels. Furthermore, the Jacobian for the depth term is also very simple, since it just indexes the z-component of the anchor point in the camera frame:

$$\frac{\partial z^m}{\partial \mathbf{P}^m_{\mathrm{C}_r}} = [0, 0, 1]. \tag{22}$$

We can also factor this out of the sum, which most importantly, reduces the dimensionality of the Hessian sum to be 64x64 instead of the original 192x192, which greatly improves efficiency by decreasing the memory needed as part of the reduction by 9x. Furthermore, this reduces pose-geometry blocks from 8x192 to 8x64, which is a 3x reduction. Note that we include each frame's affine brightness parameters in the pose block so it is of dimension 8 instead of 6.

B ScanNet Sequences

We list the ScanNet test sequences used in Table 5. We selected these sequences due to the diversity of environments and the fact that they are similar in length. Some sequences are longer or only rotational motion, which causes all methods to have much larger ATE, at which point it is difficult to tell whether global trajectory measures are useful for evaluating odometry.

C Baseline Details

To set up DROID-VO from the DROID-SLAM code, we disable the post-process at the end of the sequence that performs global bundle adjustment. More specifically, *terminate* function call that first performs two sets of bundle adjustment on all keyframes, and then also the *PoseTrajectoryFiller* that performs motion-only bundle adjustment on not only keyframes, but all frames. We use the public code for evaluating on TUM, as we found that removing the code that uses every other frame to cause significant errors on *fr1/desk2*, which skews the overall error. Since DROID-SLAM does not have public code for Replica and ScanNet, we tried different configurations. For Replica, we found that a resolution of 512x320

Sequence	Environment
709	Kitchen
710	Home office
715	Reception
716	Laundry
719	Dorm
722	Dorm
733	Living room
741	Bedroom
760	Office
780	Laundry
787	Storage
788	Gym
790	Copy room
792	Stairs
794	Outside tables
800	Bookstore
803	Supply room
804	Copy room

Table 5: ScanNet test sequences used for evaluation.

to perform better than 320x240, and is closer to the original aspect ratio of the images. For ScanNet, we found that the images are similar in resolution to TUM, so we use 320x240. For ScanNet, we must apply a crop of 10 pixels on all sides to handle undistorted regions with invalid values that remain visible.

For DeepFactors, we found that global loop closure could cause large errors, so we disabled it. As mentioned in the paper, we use all three recommended factors: photometric, reprojection, and geometric. For Replica, which has a different aspect ratio than TUM and ScanNet, we attempted both cropping and resizing the full image, with the latter giving better overall accuracy. Since DSO assumes undistorted images, we preprocessed TUM and ScanNet by first undistorting and then cropping any invalid pixels on the edges. For our method with and without correspondence, we use a 256x192 resolution for all datasets as this resolution is the same as used by the depth covariance UNet.

We compare all methods with only keyframes in the trajectory error, so we extract the up-to date keyframe estimates for evaluation. For comparing dense depth maps on ScanNet, we performed the same cropping with 10 pixel borders and resizing to a 4:3 ratio for all methods. This ensures that all predicted depth maps could be registered to the ground-truth depth maps.

D Depth Map Comparison

Since DROID-VO largely treats per-pixel depths independently, this can skew commonly used depth error metrics such as RMSE. For this reason, we omitted these results from the main paper, but include them here. We show accuracy and consistency metrics for Replica in Table 6 and Table 7, respectively. For ScanNet, we show the results in Tables 8 and 9. Bold indicates the best for a given metric, while underline indicates second-best. Note that for the error metrics (RMSE, MAE, and AbsRel) lower is better, while higher fractions for δ is better. COMO achieves the most accurate depths while also being first or second across all consistency metrics.

	RMSE (↓)	MAE (↓)	AbsRel (↓)	$\delta = 1.02$	$\delta = 1.05$	$\delta = 1.10$	$\delta = 1.25$	$\delta = 1.25^2$	$\delta = 1.25^3$
DeepFactors [7]	0.707	0.567	0.263	0.052	0.126	0.245	0.531	0.794	0.871
DROID-VO [35]	305.131	2.534	1.129	$\underline{0.368}$	$\underline{0.706}$	$\underline{0.887}$	$\underline{0.964}$	0.985	0.993
COMO-NC	$\underline{0.256}$	$\underline{0.160}$	$\underline{0.069}$	0.228	0.513	0.773	0.963	$\underline{0.992}$	**0.998**
COMO	**0.191**	**0.111**	**0.046**	**0.371**	**0.716**	**0.895**	**0.976**	**0.994**	**0.998**

Table 6: Depth accuracy evaluation on Replica.

	RMSE (↓)	MAE (↓)	AbsRel (↓)	$\delta = 1.02$	$\delta = 1.05$	$\delta = 1.10$	$\delta = 1.25$	$\delta = 1.25^2$	$\delta = 1.25^3$
DeepFactors [7]	**0.097**	**0.057**	$\underline{0.029}$	0.500	0.845	**0.968**	**0.997**	**0.999**	**0.999**
DROID-VO [35]	258.511	1.689	0.248	**0.729**	**0.901**	0.953	0.976	0.982	0.985
COMO-NC	0.181	0.098	0.040	0.455	0.760	0.912	0.984	0.997	**0.999**
COMO	$\underline{0.119}$	$\underline{0.060}$	**0.025**	$\underline{0.630}$	$\underline{0.879}$	$\underline{0.962}$	$\underline{0.993}$	$\underline{0.998}$	**0.999**

Table 7: Depth consistency evaluation on Replica.

	RMSE (↓)	MAE (↓)	AbsRel (↓)	$\delta = 1.02$	$\delta = 1.05$	$\delta = 1.10$	$\delta = 1.25$	$\delta = 1.25^2$	$\delta = 1.25^3$
TANDEM [17]	0.923	0.655	0.325	0.084	0.193	0.343	0.562	0.729	0.793
DeepFactors [7]	0.684	0.499	0.232	0.053	0.128	0.243	0.519	0.842	0.959
DROID-VO [35]	481.428	3.138	1.389	$\underline{0.114}$	$\underline{0.264}$	$\underline{0.454}$	$\underline{0.766}$	0.928	0.964
COMO-NC	$\underline{0.464}$	$\underline{0.332}$	$\underline{0.155}$	0.086	0.210	0.381	0.741	$\underline{0.955}$	**0.990**
COMO	**0.416**	**0.278**	**0.128**	**0.123**	**0.281**	**0.480**	**0.835**	**0.965**	$\underline{0.987}$

Table 8: Depth accuracy evaluation on ScanNet.

	RMSE (↓)	MAE (↓)	AbsRel (↓)	$\delta = 1.02$	$\delta = 1.05$	$\delta = 1.10$	$\delta = 1.25$	$\delta = 1.25^2$	$\delta = 1.25^3$
TANDEM [17]	0.366	0.150	0.160	0.395	0.632	0.778	0.891	0.938	0.956
DeepFactors [7]	**0.097**	**0.051**	<u>0.031</u>	<u>0.503</u>	<u>0.831</u>	**0.958**	**0.996**	**0.998**	**0.998**
DROID-VO [35]	244.693	1.140	0.258	0.474	0.736	0.869	0.947	0.969	0.975
COMO-NC	0.201	0.106	0.056	0.363	0.654	0.842	0.965	0.993	0.997
COMO	<u>0.114</u>	**0.051**	**0.028**	**0.612**	**0.867**	<u>0.955</u>	<u>0.991</u>	<u>0.997</u>	**0.998**

Table 9: Depth consistency evaluation on ScanNet.