AnyMap: Learning a General Camera Model for Structure-from-Motion with Unknown Distortion in Dynamic Scenes – Supplementary Material –

Andrea Porfiri Dal Cin Georgi Dikov Jihong Ju Mohsen Ghafoorian
XR Labs, Qualcomm Technologies, Inc.
{adalcin,gdikov,jihoju,mghafoor}@qti.qualcomm.com

In this document, we provide additional details concerning the main paper.

A. Computing the Projection Function using Lookup Tables

In the main manuscript and throughout the experimental evaluation in Sec. 4, we utilize an invertible neural network (INN) to compute a projection function π that is consistent with the learned unprojection π^{-1}, ensuring that $x = \pi(\pi^{-1}(x))$. As discussed, consistent unprojection and projection functions can be computed using various strategies, each with its own advantages and disadvantages depending on the configuration of our method and the system's hardware constraints.

In this section, we present a lookup table approach to compute a consistent, differentiable projection function π from a learned unprojection π^{-1}.

A.1. Lookup Table Construction and Interpolation

Our strategy does not compute or learn the projection $\pi : \mathbb{S}^2 \to \Omega$ directly. Instead, it is derived from a lookup table constructed using the unprojection function π^{-1}. We generate this lookup table with direction vectors $s_i \in \mathbb{S}^2$ as keys and their corresponding image points $x_i \in \Omega$ as values. Specifically, we uniformly sample pivot image points x_i across the image plane and compute their unprojected directions using $s_i = \pi^{-1}(x_i)$. The resulting pairs $s_i \mapsto x_i$ form the lookup table that approximates the projection function π.

To project a 3D point $X = (r, \psi, \phi)$ onto the image plane, we consider its direction $s = (\psi, \phi)$. Since our lookup table comprises discrete samples of direction vectors, an exact match for s may not be available. To estimate the projection $x = \pi(s)$, we interpolate between keys. We compute interpolation weights w_i based on the cosine similarity between the query direction vector s and each key s_i using a softmax function:

$$w_i = \frac{\exp\left(s \cdot s_i / \tau\right)}{\sum_k \exp\left(s \cdot s_k / \tau\right)} , \tag{1}$$

where τ is a temperature hyperparameter controlling the smoothness of the interpolation. The projected image coordinate x is then computed *differentiably* as a weighted sum of all pivot image points:

$$x = \sum_i w_i x_i . \tag{2}$$

A.2. Optimization: Radial Distortion Only

In practice, many fisheye lenses can be approximated as having radial distortion only. Leveraging this property allows us to simplify both the projection and unprojection functions, reducing the dimensionality of their inputs and outputs.

Under radial symmetry, the azimuthal angle θ in the image plane matches the azimuthal angle ϕ of the 3D point in spherical coordinates, *i.e.*, $\theta = \phi$. This symmetry reduces the problem to a one-dimensional mapping [3].

The unprojection function π^{-1} simplifies to mapping the radial distance ρ from the image center to the inclination angle ψ on the unit sphere:

$$\pi^{-1} : \rho \mapsto \psi . \tag{3}$$

Conversely, the projection function π reduces to mapping the inclination angle ψ back to the radial distance ρ in the image plane:

$$\pi : \psi \mapsto \rho . \tag{4}$$

The lookup table thus becomes one-dimensional, consisting of pairs (ψ_i, ρ_i).

To compute the interpolation weights, we use a softmin function based on the absolute differences between the query inclination angle ψ and the sampled angles:

$$w_i = \frac{\exp\left(-|\psi - \psi_i| / \tau\right)}{\sum_l \exp\left(-|\psi - \psi_l| / \tau\right)} , \tag{5}$$

where τ is a temperature hyperparameter. The projected radial distance ρ is obtained by a weighted sum:

$$\rho = \sum_i w_i \rho_i . \tag{6}$$

Since $\theta = \phi$, the projected image point x in polar coordinates is given by (ρ, ϕ).

The equality $\theta = \phi$ holds under the assumption of radial symmetry in the camera model. [3] provides proofs for specific camera models.

Note that the summations in Equation (1) and (2) involve all pivot points in the lookup table, which can be computationally intensive. This observation naturally guides us to the optimization strategy discussed in the following sections.

A.3. Optimization: Coarse-to-Fine Projection

While increasing the number of pivot points x_i in the lookup table can reduce the projection error $\epsilon = x - \pi\left(\pi^{-1}(x)\right)$, it also increases memory consumption. To mitigate this, we introduce a *coarse-to-fine refinement* strategy that enhances accuracy without significantly impacting computational efficiency.

Starting with an initial projection $x = \pi(s)$ computed using a coarse lookup table, we define a square neighborhood $N(x)$ centered at x, extending $\pm\delta$ along both image axes. Here, δ is the minimum distance between the sampled pivot points. We then sample additional pivot points x_l within $N(x)$ and compute refined interpolation weights w_l as in Equation (1). The refined projection is calculated as:

$$x' = \sum_{x_l \in N(x)} w_l x_l \ . \tag{7}$$

This refinement process can be iterated, each time focusing on a smaller neighborhood to further enhance the projected coordinates.

A.4. Comparison to Invertible Neural Network

Tab. 1 compares two implementations of `AnyMap`: one using a lookup table (L-UP), where the projection function is implemented via a lookup table derived from the learned unprojection, and the other using invertible neural networks (INN) as presented in the main paper. The results show that on synthetically distorted datasets with only radial distortion, both methods achieve similar reprojection errors, but the lookup table implementation runs faster. However, when evaluating on the Aria Everyday Activities (AEA) [8] dataset—which includes tangential distortions and does not allow optimizations for radial symmetries—the `AnyMap` L-UP exhibits similar reprojection error but slower performance. Therefore, depending on the use case and hardware constraints, one may choose the appropriate implementation of `AnyMap`.

	MipNeRF-360		LLFF		T & T		AEA	
Method	Re ↓	t ↓	Re ↓	t ↓	Re ↓	t ↓	Re ↓	t ↓
`AnyMap` L-UP	1.04	**8.5**	**2.34**	**5.8**	2.03	**14.5**	3.02	11.9
`AnyMap` INN	**1.02**	10.3	2.35	6.4	**2.01**	18.4	**3.01**	**9.5**

Table 1. Comparison of reprojection error (Re) and runtime (t) between `AnyMap` implementations using a lookup table (L-UP) and invertible neural networks (INN) across various datasets. Lower values indicate better performance, and the best results are highlighted in bold. Time in minutes.

B. Conventional Depth Parameterization

`AnyMap`, adopts a range parameterization where the network predicts the Euclidean distance from the camera's optical center to a 3D point, represented in spherical coordinates as $X = (r, \psi, \phi)$. The range r is defined in Cartesian coordinates $X^{\text{cart}} = (x, y, z)$ as:

$$r = \sqrt{x^2 + y^2 + z^2}. \tag{8}$$

The range represents the true straight-line distance between the camera and the point in space, encompassing both the depth along the optical axis and any lateral displacement.

Traditionally, depth refers to the component of a 3D point along the camera's optical axis, typically the z-axis in the camera coordinate system. Given a point $X^{\text{cart}} = (x, y, z)$ in the camera frame, the depth is simply:

$$\text{Depth} = z. \tag{9}$$

While depth effectively measures distance along the optical axis, it does not account for lateral displacement, which can be significant in wide field-of-view (FOV) imaging.

Considering the 3D point in spherical coordinates $X = (r, \psi, \phi) \in \mathbb{R}^+ \times \mathbb{S}^2$, where r is the range, ψ is the inclination angle (angle from the positive z-axis), and ϕ is the azimuthal angle (angle from the positive x-axis in the xy-plane). The direction vector $s = (\psi, \phi)$ defines the orientation of the point relative to the camera's optical center. The relationship between spherical and Cartesian coordinates is given by:

$$x = r\sin(\psi)\cos(\phi), \tag{10}$$
$$y = r\sin(\psi)\sin(\phi), \tag{11}$$
$$z = r\cos(\psi). \tag{12}$$

Dividing x and y by z yields:

$$\frac{x}{z} = \tan(\psi)\cos(\phi), \quad \frac{y}{z} = \tan(\psi)\sin(\phi). \tag{13}$$

This parameterization is effective for cameras with a field of view less than $180°$. However, it becomes undefined at $\psi = \frac{\pi}{2}$. At this angle, the tangent function exhibits a singularity: as ψ approaches $\frac{\pi}{2}$, $\tan(\psi)$ approaches infinity, causing $\frac{x}{z}$ and $\frac{y}{z}$ to become unbounded. These singularities lead to numerical instabilities in computations involving $\tan(\psi)$, which is particularly problematic in optimization algorithms relying on gradient descent.

Method	MipNeRF-360 [1]			LLFF [10]			Tanks & Temples [7]			Aria Everyday Activities [8]		
	LPIPS ↓	Re ↓	t ↓	LPIPS ↓	Re ↓	t ↓	LPIPS ↓	Re ↓	t ↓	LPIPS ↓	Re ↓	t ↓
GLOMAP [11]	0.054	1.50	**1.4**	0.098	3.21	**0.5**	0.069	2.87	**1.8**	0.124	4.94	**2.3**
COLMAP [13]	0.053	1.49	5.2	0.095	3.20	1.3	0.068	2.87	5.8	0.121	4.92	8.7
AnyMap w/ NICE	0.069	2.08	8.3	0.124	4.05	5.1	0.080	3.20	12.3	0.139	5.20	8.8
AnyMap w/ RealNVP	0.065	1.95	8.6	0.111	3.78	5.3	0.075	3.08	12.9	0.127	4.97	8.9
AnyMap w/ ResFlow	**0.021**	**1.02**	10.3	**0.069**	**2.35**	6.4	**0.056**	**2.01**	18.4	**0.076**	**3.01**	9.5

Table 2. Comparison of invertible neural network implementations in AnyMap using different INNs: NICE [4], RealNVP [5], and ResFlow [2]. The table reports LPIPS, reprojection error (Re), and convergence time (t in minutes). Lower values of LPIPS and Re indicate better performance.

Since AnyMap relies on gradient-based optimization for end-to-end training, avoiding numerical instabilities is crucial. Near $\psi = \frac{\pi}{2}$, the spikes in gradient computation due to infinite or undefined gradients can hinder the convergence of the optimization algorithm. Such instabilities propagate through the network, leading to unreliable parameter updates and poor model performance.

By adopting the range parameterization, our model ensures that the range r remains well-defined and finite for all values of ψ and ϕ, including at $\psi = \frac{\pi}{2}$. This allows the model to accurately represent points in all directions, supporting cameras with FOVs up to $180°$ and beyond. By eliminating the singularities present in the traditional parameterization, we enhance the stability of the optimization process. This leads to more reliable convergence during gradient-based optimization and ultimately improves the model's performance.

C. Parametric Estimation of Focal Length and Distortion Coefficients

In AnyMap, we do not explicitly estimate the camera's focal length f or distortion coefficients $\mathbf{k} = (k_1, \ldots, k_n)$. Instead, we implicitly learn an unprojection function π^{-1} for each pixel, which maps an image point x to the direction of the corresponding incoming light ray, as specified by the general camera model in Sec. 3.1.

Since most existing implementations for tasks like novel view synthesis in Neural Radiance Fields and 3D Gaussian Splatting lack support for the proposed general camera model, and given that practical camera calibration often requires parameters according to a standardized mathematical model, we propose a procedure to robustly estimate the focal length f and distortion coefficients $\mathbf{k}$ from the learned unprojection π^{-1}.

To achieve this, we fit π^{-1} to an optional mathematical camera model $\mathbf{m}$ to estimate the focal length $\hat{f}$ and distortion coefficients $\hat{\mathbf{k}}$ that best approximate the learned unprojection π^{-1}. This fit is based on the unprojection equation Ψ^{-1} defined by the model $\mathbf{m}$.

We obtain the estimates $\hat{f}$ and $\hat{\mathbf{k}}$ by minimizing the following objective function:

$$\hat{f}, \hat{\mathbf{k}} = \underset{f,\mathbf{k}}{\operatorname{argmin}} \sum_{x \in \Omega} \alpha \left\| \pi^{-1}(x) - \Psi^{-1}(x, f, \mathbf{k}) \right\|_2 , \quad (14)$$

where α is the robust Cauchy loss function, and $\Psi^{-1}(x, f, \mathbf{k})$ computes the direction of the incoming light ray for each image point x based on the optimized parameters f and $\mathbf{k}$.

To optimize Equation (14), we apply the Trust Region Reflective (TRF) algorithm, which operates within a constrained parameter space informed by the input camera model $\mathbf{m}$. For example, in the Unified Camera Model (UCM) [9], we constrain $\alpha \in [0, 1]$. The TRF algorithm requires an initial estimate, f_0 and $\mathbf{k}_0$, which we obtain following the method in [3] using the differential evolution algorithm [15]. Although differential evolution can involve more function evaluations than conventional gradient-based techniques, the restricted search space, constrained by the camera model, allows us to find initial estimates efficiently, with an average computation time of 98 ms for the Extended Unified Camera Model (EUCM) [6] and 185 ms for the Fisheye624 model [12]. In our implementation, most computations are GPU-accelerated for fast runtimes.

D. Dataset Details

For the Aria Everyday Activities dataset [8], we selected three sequences, each consisting of 100 frames, extracted from the following recordings:

- loc1_script1_seq1_rec1
- loc1_script1_seq3_rec1
- loc1_script1_seq5_rec1

To illustrate the levels of image distortion used in our synthetic evaluation, Fig. 1 presents example frames from sequences in the LLFF [10] and Tanks & Temples [7] datasets.

E. Additional Implementation Details

We implement the MLP $\mathcal{G}$ that predicts the weights for each basis trajectory in our motion parameterization following the

Figure 1. Examples of image distortion levels applied in our synthetic evaluation, illustrated using sequences from LLFF [10] (top) and Tanks & Temples [7] (bottom).

	MipNeRF-360 [1]						LLFF [10]						Tanks & Temples [7]						Aria Everyday Activities [8]					
Method	PSNR ↑	SSIM ↑	LPIPS ↓	Re ↓	Δf ↓	ATE ↓	PSNR ↑	SSIM ↑	LPIPS ↓	Re ↓	Δf ↓	ATE ↓	PSNR ↑	SSIM ↑	LPIPS ↓	Re ↓	Δf ↓	ATE ↓	PSNR ↑	SSIM ↑	LPIPS ↓	Re ↓	Δf ↓	ATE ↓
AnyMap (w/o $\mathcal{L}^u$)	26.62	0.841	0.035	1.22	2.80	0.0024	20.04	0.727	0.080	2.69	1.12	0.0042	21.77	0.723	0.059	2.39	0.95	0.0138	19.24	0.648	0.087	3.69	1.05	0.0112
AnyMap (dynamic)	27.97	0.910	0.023	1.04	1.04	0.0024	21.20	0.768	0.073	2.42	0.91	0.0040	22.79	0.778	0.057	2.03	0.75	0.0110	20.31	0.690	0.080	3.05	0.96	0.0109
AnyMap (parameteric)	20.38	0.650	0.078	2.21	3.45	0.0032	16.23	0.512	0.104	3.15	2.10	0.0061	20.89	0.695	0.085	3.06	1.62	0.0196	14.06	0.528	0.130	5.20	2.05	0.0138
AnyMap	**28.15**	**0.915**	**0.021**	**0.65**	**1.02**	**0.0022**	**21.30**	**0.777**	**0.069**	**2.35**	**0.89**	**0.0034**	**22.73**	**0.780**	**0.056**	**2.01**	**0.73**	**0.0092**	**20.38**	**0.692**	**0.076**	**3.01**	**0.94**	**0.0101**

Table 3. **AnyMap variants** show the impact of design choices in our method.

approach of [16]. To enhance convergence speed, we apply positional encoding to the input 3D points and time frames.

For initialization, we determine the initial parameters f and α by evaluating all combinations of 10 possible values for $\alpha \in \{0.0, 0.1, 0.2, \ldots, 1.0\}$ and 60 equally spaced values for f in the range $[0.5, 2.0]$, resulting in 600 candidate pairs.

We optimize AnyMap using the AdamW optimizer with a learning rate of 3×10^{-5}.

E.1. Comparing Invertible Transformations

In AnyMap, we employ an invertible neural network (INN) based on ResFlow [2] to implement the camera network, thereby implicitly learning a non-linear invertible unprojection function. While faster alternatives like NICE [4] and RealNVP [5] are available, we found that their representational power is insufficient to achieve reprojection errors comparable to ResFlow or even to traditional multi-view Structure-from-Motion methods such as COLMAP and GLOMAP. Tab. 2 presents the reprojection error and LPIPS scores obtained using ResFlow in our AnyMap implementation and compares them to variants where the INN is implemented using NICE and RealNVP. The results indicate that although convergence times are faster when using NICE and RealNVP, the reprojection error (Re) and LPIPS scores are significantly worse compared to ResFlow, leading to results inferior to those of COLMAP.

E.2. Memory and Time Requirements

The computational complexity of AnyMap scales linearly with the number of frames V. Optimizing a 150-frame video takes approximately 18 minutes on an NVIDIA A100 GPU and requires up to 38 GB of peak memory consumption. The time and memory usage are comparable to the state-of-the-art FlowMap [14], which employs a similar parameterization of depth and extrinsics but does not incorporate a learnable camera model. The techniques discussed in [14] for significantly reducing peak memory usage—such as backpropagating through only a subset of videos in each epoch—are also applicable to our method, as is the application of early stopping to reduce training times.

E.3. Priors in unprojection loss

We intentionally avoid relying on strong camera-model priors by generating images with one distortion model (KB6) while using a different model (EUCM) in our unprojection loss $\mathcal{L}^u$. To further test this setup, we explore the challenging scenario of applying $\mathcal{L}^u$ with the radial-only EUCM on the Aria Everyday dataset, which actually contains both radial and tangential distortions. Under these conditions, AnyMap achieves an LPIPS score of 0.078 (2.6% worse), a reprojection error Re of 3.10 (2.9% worse), and the same $\Delta f = 0.94$ as the correct Fisheye624 model. Despite these slight performance drops, our results still outperform the variant of AnyMap without $\mathcal{L}^u$ (see Tab. 4), confirming that $\mathcal{L}^u$ remains crucial even without prior knowledge of the actual camera model.

Method	MipNeRF-360			LLFF			Tanks&Temples			Aria Everyday		
	ATE ↓	LPIPS ↓	t (min.) ↓	ATE ↓	LPIPS ↓	t (min.) ↓	ATE ↓	LPIPS ↓	t (min.) ↓	ATE ↓	LPIPS ↓	t (min.) ↓
FlowMap	0.0891	–	**9.2**	0.0623	–	**5.9**	0.1129	–	**17.2**	0.0854	–	**8.9**
FlowMap-UCM	0.0704	0.250	12.5	0.0501	0.295	9.4	0.0655	0.246	22.4	0.0490	0.301	13.2
FlowMap-EUCM	0.0640	0.197	15.4	0.0469	0.180	12.6	0.0582	0.201	28.3	0.0447	0.248	17.5
AnyMap	**0.0022**	**0.021**	10.3	**0.0034**	**0.069**	6.4	**0.0092**	**0.056**	18.4	**0.0101**	**0.076**	9.5

Table 4. Quantitative comparison of FlowMap-based methods (pinhole, UCM, and EUCM) against our proposed AnyMap on four datasets: MipNeRF-360, LLFF, Tanks&Temples, and Aria Everyday. We report Average Trajectory Error (ATE), LPIPS, and training time t (in minutes). Our AnyMap achieves consistently lower errors while remaining memory- and runtime-efficient, highlighting the benefits of an implicit, compact MLP over multi-parameter camera models.

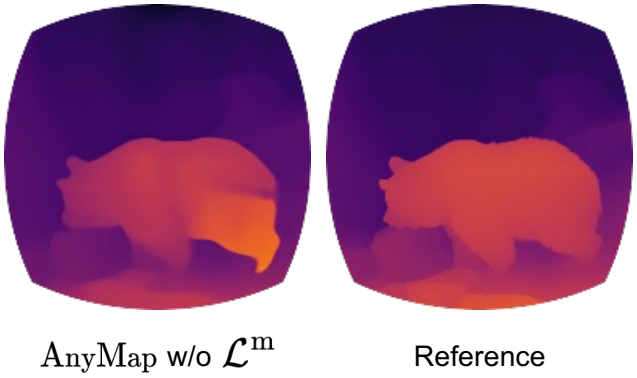

Figure 2. Effect of disabling the monocular depth loss $\mathcal{L}^{\mathrm{m}}$ on scale consistency. Without the monocular depth constraint ($\lambda_{\mathrm{m}} = 0$), dynamic regions such as the moving paw of the bear exhibit scale drift, leading to depth predictions that are inconsistent with the static parts of the object.

F. Additional Experiments

We present experimental results that were omitted from the main manuscript due to space constraints. Tab. 3 provides the complete metrics from our ablation study. Fig. 2 offers a qualitative example of scale drift in image regions corresponding to dynamic objects, which occurs after multi-view optimization of 3D geometry when the monocular depth loss $\mathcal{L}^{\mathrm{m}}$ is disabled ($\lambda_{\mathrm{m}} = 0$). In this example, the paw of the bear, which moves significantly between frames, is predicted at a depth that is inconsistent with the rest of the bear.

Finally, in Fig. 3, we provide some visual examples of the 3D geometry obtained using AnyMap on select video sequences from Tanks & Temples.

F.1. Extended comparison to learning-based SfM

Learning-based SfM approaches typically assume pinhole (un)projection with only a few parameters (e.g., focal length). However, handling camera distortions that involve two or more parameters proves challenging when learning them explicitly. As shown in our ablation study in Tab. 4, the parametric version of AnyMap underperforms due to ambiguities in these multi-parameter models, leading to unstable training, as also demonstrated in [3].

Moreover, the runtime and memory usage of state-of-the-art learning-based Structure-from-Motion methods (*e.g.*, FlowMap [14]) grow rapidly with increased distortion complexity. To circumvent these scalability issues, we learn (un)projection implicitly as a point-to-ray mapping using a compact MLP. This approach remains geometrically consistent through end-to-end supervision from optical flow matches, without requiring explicit parameters or prior camera-model knowledge.

To evaluate our design, we replace FlowMap's pinhole model with two alternative distortion models, UCM and EUCM, maintaining FlowMap's soft selection of 60 candidates per parameter. We report results in Tab. 4. Both variants yield worse performance than our AnyMap approach, consume more than twice the VRAM (70 GB vs. 36 GB), and incur higher runtimes due to the expanded parameter search. Extending FlowMap to Fisheye624 (which requires over 10 parameters) was infeasible even with 80 GB of VRAM, underscoring the benefits of our implicit, compact MLP strategy.

F.2. Comparison to conventional SLAM

In Tab. 5, we compare our method with three popular SLAM systems: ORB-SLAM, LSD-SLAM, and DROID-SLAM. All SLAM approaches use ground-truth intrinsics, whereas AnyMap is fully uncalibrated. Although DROID-SLAM outperforms ORB- and LSD-SLAM, AnyMap provides further accuracy gains. This behavior is expected: AnyMap acts as an SfM pipeline that jointly optimizes all frames for improved accuracy, unlike real-time SLAM methods designed for faster operation.

We also evaluate AnyMap on a 500-frame segment from the KITTI Depth benchmark. Consistent with other datasets, AnyMap achieves a lower ATE than DROID-SLAM while requiring longer runtime. Its final accuracy is comparable to COLMAP (ATE 0.0089 for AnyMap vs. 0.0090 for COLMAP), underscoring the benefits of our joint optimization approach even in large-scale, real-world scenarios.

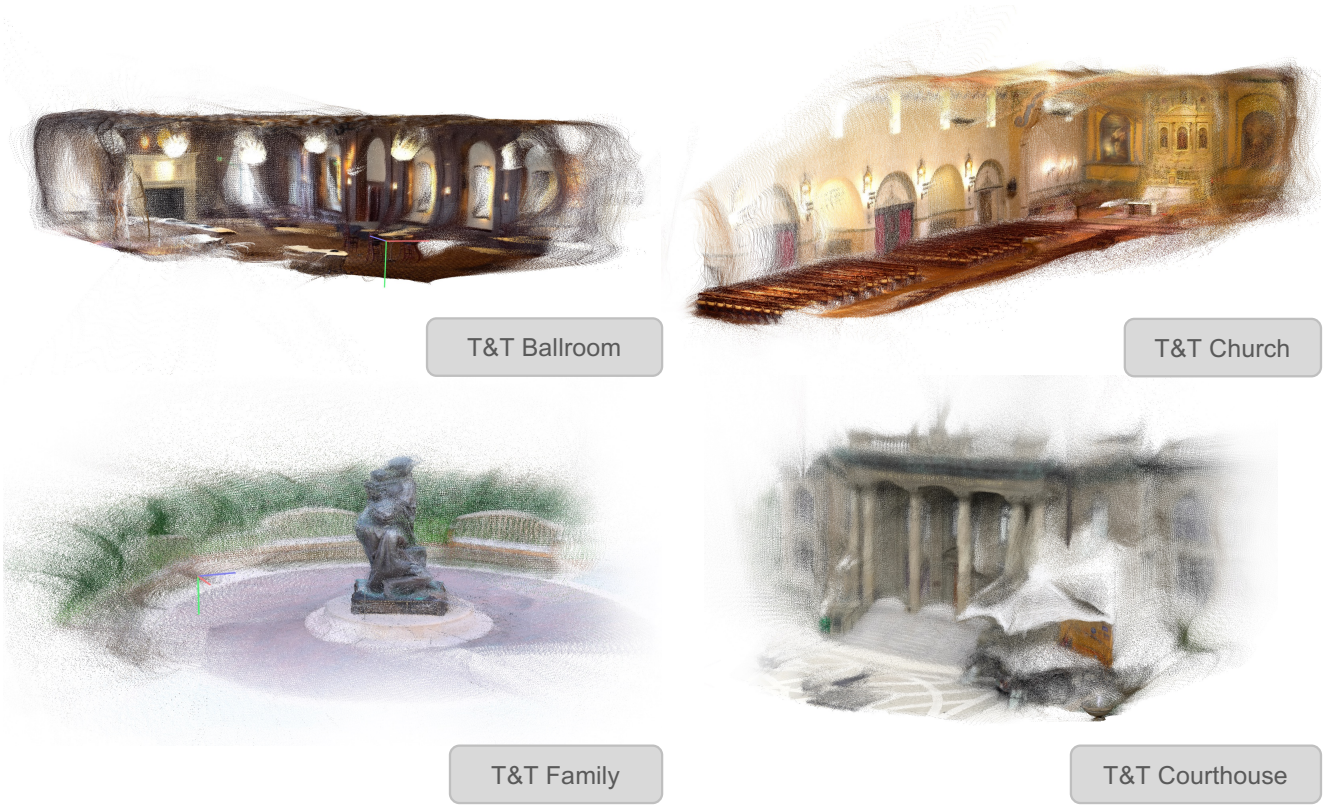

Figure 3. Visualization of dense point clouds obtained by running `AnyMap` on synthetically *distorted* video sequences from Tanks & Temples [7].

	MipNeRF-360		LLFF		Tanks&Temples		Aria Everyday		KITTI Depth	
Method	ATE ↓	t (min.) ↓	ATE ↓	t (min.) ↓	ATE ↓	t (min.) ↓	ATE ↓	t (min.) ↓	ATE ↓	t (min.) ↓
ORB-SLAM	0.0589	**0.4**	0.0490	0.8	0.1065	**0.7**	0.1076	**0.8**	0.603	**0.9**
LSD-SLAM	0.0634	**0.4**	0.0561	**0.2**	0.1120	0.8	0.1204	**0.8**	0.620	**0.9**
DROID-SLAM	0.0428	0.6	0.0376	0.3	0.0943	0.8	0.0827	**0.8**	0.0459	1.0
`AnyMap`	**0.0022**	10.3	**0.0034**	6.4	**0.0092**	18.4	**0.0101**	9.5	**0.0089**	17.9

Table 5. Comparison of ORB-SLAM, LSD-SLAM, and DROID-SLAM against our uncalibrated `AnyMap` across five datasets. We measure Average Trajectory Error (ATE) and runtime t (in minutes). Although these SLAM methods leverage ground-truth intrinsics and can operate in real time, `AnyMap` jointly optimizes all frames and consistently achieves lower ATE, demonstrating the accuracy benefits of a full SfM pipeline at the cost of longer computation.

References

[1] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5855–5864, 2021. 3, 4

[2] Ricky TQ Chen, Jens Behrmann, David K Duvenaud, and Jörn-Henrik Jacobsen. Residual flows for invertible generative modeling. *Advances in Neural Information Processing Systems*, 32, 2019. 3, 4

[3] Andrea Porfiri Dal Cin, Francesco Azzoni, Giacomo Boracchi, and Luca Magri. Revisiting calibration of wide-angle radially symmetric cameras. In *European Conference on Computer Vision*, pages 214–230. Springer, 2025. 1, 2, 3, 5

[4] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Nonlinear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014. 3, 4

[5] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016. 3, 4

[6] Bogdan Khomutenko, Gaëtan Garcia, and Philippe Martinet. An enhanced unified camera model. *IEEE Robotics and Automation Letters*, 1(1):137–144, 2015. 3

[7] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36(4): 1–13, 2017. 3, 4, 6

[8] Zhaoyang Lv, Nickolas Charron, Pierre Moulon, Alexander Gamino, Cheng Peng, Chris Sweeney, Edward Miller, Huixuan Tang, Jeff Meissner, Jing Dong, et al. Aria everyday activities dataset. *arXiv preprint arXiv:2402.13349*, 2024. 2, 3, 4

[9] Christopher Mei and Patrick Rives. Single view point omnidirectional camera calibration from planar grids. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3945–3950. IEEE, 2007. 3

[10] Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (ToG)*, 38(4):1–14, 2019. 3, 4

[11] Linfei Pan, Dániel Baráth, Marc Pollefeys, and Johannes Lutz Schönberger. Global structure-from-motion revisited. In *European Conference on Computer Vision (ECCV)*, 2024. 3

[12] Davide Scaramuzza, Agostino Martinelli, and Roland Siegwart. A toolbox for easily calibrating omnidirectional cameras. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5695–5701. IEEE, 2006. 3

[13] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 3

[14] Cameron Smith, David Charatan, Ayush Tewari, and Vincent Sitzmann. Flowmap: High-quality camera poses, intrinsics, and depth via gradient descent. *arXiv preprint arXiv:2404.15259*, 2024. 4, 5

[15] Rainer Storn and Kenneth Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11:341–359, 1997. 3

[16] Zhoutong Zhang, Forrester Cole, Richard Tucker, William T Freeman, and Tali Dekel. Consistent depth of moving objects in video. *ACM Transactions on Graphics (ToG)*, 40(4):1–12, 2021. 4