FastMap: Revisiting Structure from Motion through First-Order Optimization

Jiahao Li[1*] Haochen Wang[1*] Muhammad Zubair Irshad[2] Igor Vasiljevic[2]
Matthew R. Walter[1] Vitor Campagnolo Guizilini[2†] Greg Shakhnarovich[1†]
[1]TTI-Chicago [2]Toyota Research Institute
{jiahao,whc,mwalter,greg}@ttic.edu
{zubair.irshad,igor.vasiljevic,vitor.guizilini}@tri.global

Abstract

We propose FastMap*, a new global structure from motion method focused on speed and simplicity. Previous methods like COLMAP and GLOMAP are able to estimate high-precision camera poses, but suffer from poor scalability when the number of matched keypoint pairs becomes large, mainly due to the time-consuming process of second-order Gauss-Newton optimization. Instead, we design our method solely based on first-order optimizers. To obtain maximal speedup, we identify and eliminate two key performance bottlenecks: computational complexity and the kernel implementation of each optimization step. Through extensive experiments, we show that* FastMap *is up to* 10× *faster than COLMAP and GLOMAP with GPU acceleration and achieves comparable pose accuracy. Project webpage:* https://jiahao.ai/fastmap.

1. Introduction

Data is the fuel for state-of-the-art computer vision systems. Recently, synthetic 3D datasets [9, 15, 17, 39, 65, 71] have been scaled up to provide supervision for diverse tasks such as Visual-SLAM [59], 3D point tracking [18, 25], 3D asset generation [33, 55, 68], etc. However, scaling up real-world 3D data remains difficult due to the lack of ground-truth camera poses. Many applications such as monocular depth estimation [6, 26, 48] and learning-based 3D reconstruction [11, 63, 64] still rely on pseudo-ground-truth produced by pure geometry-based structure from motion systems (SfM) such as COLMAP [53]. However, COLMAP is slow—processing a scene consisting of thousands of images can take multiple days. Global SfM methods such as GLOMAP [44] improve upon COLMAP's speed, but still take many hours to converge on large scenes. Efficiently scaling learning-based systems to more training data requires a fast and high-quality ground-truth annotator.

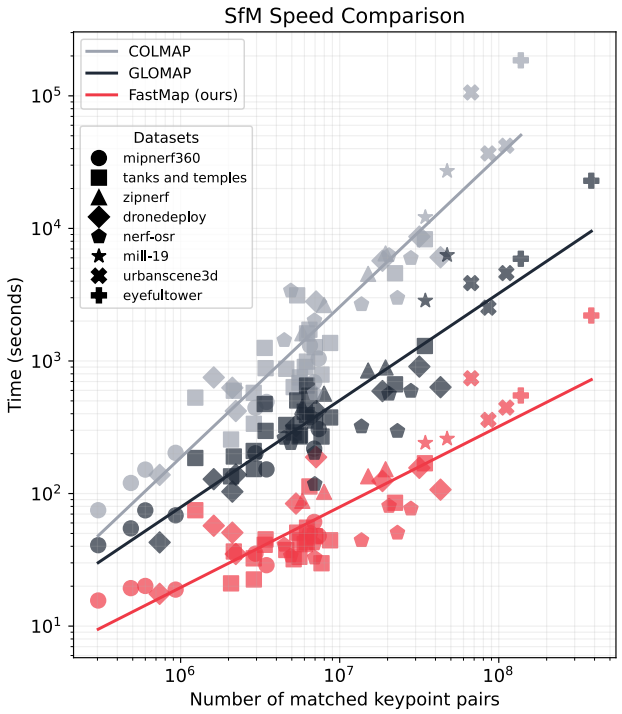

Figure 1. Timing of FastMap compared to COLMAP and GLOMAP (all with GPU acceleration on a single A6000) on scenes from eight datasets, excluding the matching stage for all methods. Note the **logarithmic** time scale. Lines represent a least squares power function fit to timing across multiple datasets, as a function of the number of matched keypoint pairs.

A typical SfM pipeline spends most of the time doing optimization. For example, COLMAP [53] registers one image at a time, and performs a round of global bundle adjustment optimization every few images to avoid drifting. GLOMAP [44] estimates global translation by optimizing the camera centers and 3D points from random initializations, and uses a final round of bundle adjustment at the end for refinement. These optimization problems are nonlinear [60] and require iterative methods to solve. Specifically,

*core contributors
†joint supervision & equal contributions

quasi second-order methods such as Levenberg-Marquardt (LM) [1] are the standard choice. LM is a trust-region variant of the Gauss-Newton method, which estimates the Hessian with the residual Jacobian. While several techniques have been adopted to speed up LM optimization, such as the Schur complement trick and sparse Cholesky decomposition, each iteration still takes long wall-clock time to compute. In contrast, adaptive first-order methods [28] can potentially eliminate the scalability bottleneck and simplify algorithm design, but are not well explored in the current SfM literature.

To bridge this gap, we propose FASTMAP, a global SfM method that relies only on first-order optimization. We identify and tackle two main speed bottlenecks when switching from second-order to first-order optimizers. First, many sub-problems in SfM, such as bundle adjustment [60] and global positioning [44], jointly optimize camera poses and 3D points. Usually, the number of 3D points is orders-of-magnitude larger than the number of image pairs. This is not the main issue for second-order methods because they spend most of the time solving the reduced linear system from the Schur complement, but can be a significant bottleneck for first-order methods that only require computing gradients. To address this, we design our method such that all the optimization problems involved have a per-step computational complexity independent of the number of 3D points.

The second bottleneck comes from the implementation. While it is straightforward to implement gradient descent with modern deep learning Autograd engines [46], we find that it leads to sub-optimal utilization of GPU resources. This is mainly due to the large overhead of kernel launching, unnecessary data movement between global and shared memory, and improper kernel choice by the library. Instead, we use kernel fusion to perform forward and backward steps in one CUDA kernel, which significantly speeds up each optimization step.

Extensive experiments on 8 different datasets demonstrate that our method can be up to $10\times$ faster than both COLMAP and GLOMAP with GPU-accelerated Ceres solver [1] (Fig. 1). It also achieves comparable accuracy to these two state-of-the-art methods in terms of both pose accuracy and novel view synthesis quality.

In summary, we introduce FASTMAP, a new SfM framework with the following contributions:

- We show that first-order optimization can be used to make a scalable and accurate SfM system.
- We design a fully 3D point-free pipeline that is friendly to first-order optimizers.
- We show that kernel fusion can significantly speed up gradient computation by eliminating overhead.

2. Related Work

Global SfM Systems *Incremental* SfM methods like COLMAP [53] are state-of-the-art in accuracy and robustness, but *global* SfM systems are catching up [44]. These methods solve for all camera poses at once to avoid registering images sequentially, dramatically improving run-time. OpenMVG [41] and Theia [57] are two popular global SfM systems which are fast, but trail COLMAP in accuracy and robustness [44]. HSfM [8] is a hybrid approach that combines incremental and global approaches, estimating rotations globally and translations incrementally. Unlike prior global approaches that first perform translation averaging and then triangulation, GLOMAP [44] combines both steps, solving for camera translations and 3D points in one global step. They report results on par with COLMAP, but with large speed improvements both on small- [54] and large-scale [52] datasets.

Global Rotation and Translation In a typical global SfM system, global rotation and translation are estimated directly from pairwise relative motions. Hartley et al. [21] provides a good tutorial on rotation averaging. Govindu [16] frames motion estimation as a global optimization problem, and Martinec and Pajdla [38] first solves for camera rotations using pairwise constraints and then obtains translations from a linear system using epipolar constraints. Wilson and Bindel [66] propose a more stable optimizer for rotation averaging. Many existing approaches struggle when baseline lengths differ significantly [44]. LUD [43] and Zhuang et al. [73] focus on improving stability and robustness in ill-conditioned scenarios. Jiang et al. [23] introduces a linearized approach that enforces constraints across camera triplets, ensuring consistency in multi-view configurations. Wilson and Snavely [67] propose improving translation estimation by combining outlier removal with a simplified solver.

Learning-based SfM Learning-based SfM methods vary in the degree of their departure from the traditional SfM pipeline and in their tradeoff between speed and accuracy. VGGSfM [62] hews closely to the traditional SfM methodology, building on point-tracking methods to propose a fully differentiable SfM method that includes bundle adjustment. Flowmap [56] uses pretrained optical flow and point tracking networks and a depth CNN to optimize per-scene global poses, calibration, and depth maps using gradient descent. Ace-Zero [7] uses a trained dense 3D scene coordinate regressor as an alternative to triangulation and registration in incremental SfM, instead incrementally relocalizing with the learned coordinate regressor. The DUSt3R [64] architecture, which maps image pairs to *point maps*, initiated a new paradigm in learned SfM. DUSt3R pointmap estimates from image pairs can be used for camera calibration, depth estimation, correspondence, pose estimation and dense re-

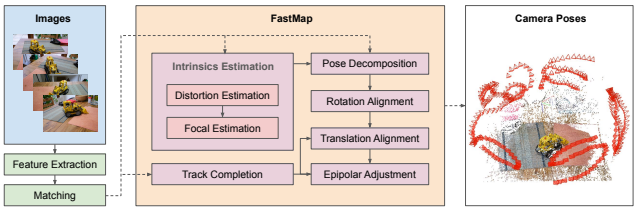

Figure 2. An overview of FASTMAP. Input images are processed using feature extraction and matching. Given the matching results, FASTMAP estimates the intrinsics and extrinsics of the cameras. Finally a sparse point cloud is generated by triangulation.

construction. Many recent works [36, 63, 70] improve upon DUSt3R in various ways, such as efficiently processing more input images [70], using diffusion models [36], and predicting more 3D geometry attributes [63]. In particular, MASt3R [32] upgrades DUSt3R using dense correspondences from the predicted DUSt3R pointmap pairs, and MASt3R-SfM [11] incorporates a global alignment stage, offering a full-fledged SfM system based on MASt3R.

3. Method

Overview Our proposed FASTMAP framework (Fig. 2) consists of multiple stages roughly in sequential order. In this section we introduce the algorithmic details. Sec. 3.1 describes how we estimate the distortion parameters and focal lengths for each camera after extracting and matching keypoints. Then, Sec. 3.2 analyzes the pros and cons of first-order vs second-order optimization, and Sec. 3.3 describes all the optimization problems for global pose estimation and refinement. Finally, we discuss kernel fusion (Sec. 3.4) for further speed-up.

Matching FASTMAP's matching stage is identical to that of both COLMAP and GLOMAP: it involves first extracting and matching keypoints from the input images, followed by geometric verification of the resulting image pairs [53]. The output of the matching stage consists of the set of inlier keypoint pairs and either an estimated fundamental matrix $\mathbf{F}_{ij}$ or a homography matrix $\mathbf{H}_{ij}$ (the latter if it is consistent with sufficiently many inlier matches) for each image pair (i, j) with enough correspondences.

3.1. Intrinsics Estimation

Accurate intrinsics estimation has a direct impact on the precision of relative pose decomposition, which is critical for later stages. In this section, we describe the algorithms that FASTMAP employs to estimate the focal lengths and distortion from the matching results.

Camera Assumptions We use a one-parameter radial distortion model. The principal point is fixed to be the center, therefore the only intrinsics parameters to estimate are the distortion parameter and the focal length. We also assume that all the images are taken with a small number of distinct cameras, and that we know which images are from the same camera. In practice, this can be inferred from image resolutions, EXIF tags, file and directory names, etc.

Distortion Estimation We formulate distortion estimation as the problem of finding the distortion parameters that result in the most consistent two-view geometric model for each image pair (e.g., the fundamental matrix estimated from undistorted keypoints has the lowest epipolar error). We do so using the one-parameter division undistortion model [3, 12, 14, 22]

$$x_u = \frac{x_d}{1 + \alpha r_d^2} \qquad y_u = \frac{y_d}{1 + \alpha r_d^2}, \tag{1}$$

where (x_d, y_d) and (x_u, y_u) are the distorted and undistorted coordinates, respectively, $r_d = \sqrt{x_d^2 + y_d^2}$ and $r_u = \sqrt{x_u^2 + y_u^2}$, and α is the distortion model parameter. The model can be inverted in closed form to apply distortion to keypoints [12]. We found this model to be more convenient than the commonly used, but difficult to invert Brown-Conrady distortion model [10].

We use brute-force interval search to estimate the distortion parameter α. Given a set of image pairs that share the same α, we sample set of candidate values from an interval $[\alpha_{\text{min}}, \alpha_{\text{max}}]$, and evaluate the average epipolar errors for each candidate after undistorting and re-estimating the fundamental matrices based on the sampled α (we ignore all the homography matrices at this stage). This method directly minimizes our objective (epipolar error) and takes into account information from multiple different image pairs, improving robustness to noise. Moreover, each candidate can be evaluated independently, making it highly parallelizable on a GPU. In the supplementary material (Sec. B.1), we provide more details regarding how to accelerate the above method with hierarchical sampling, and discuss generalizations to images with different distortion parameters.

Focal Length Estimation We use the estimated distortion model to undistort all the keypoints, and re-estimate the fundamental and homography matrices. At this point, the only remaining unknown intrinsic parameter for each camera is the focal length. We estimate the focal length based on the re-estimated fundamental matrices from undistorted keypoints. While this is a well studied problem [2, 19, 24, 30, 58], existing methods are susceptible to noise or require nonlinear optimization. Instead, we employ an interval search strategy similar to that used for distortion estimation, but with a different objective.

Our method is based on the well-known property that a 3×3 matrix is an essential matrix if and only if its singular values are such that $\lambda_1 = \lambda_2$ and $\lambda_3 = 0$ (where $\lambda_1 \geq \lambda_2 \geq \lambda_3$) [13, 20]. Given the correct fundamental matrix $\mathbf{F}$ and intrinsics matrix $\mathbf{K}$, the essential matrix $\mathbf{E} = \mathbf{K}^\top \mathbf{F} \mathbf{K}$ should satisfy $\frac{\lambda_1}{\lambda_2} = 1$. If all images

share the same intrinsics, with a set of fundamental matrices $\{\mathbf{F}_i\}$, we can evaluate the accuracy of a candidate focal length f using the singular value ratio above. Letting $\lambda_1^{(i)} \geq \lambda_2^{(i)} \geq \lambda_3^{(i)}$ be the singular values of $\mathbf{K}^\top \mathbf{F}_i \mathbf{K}$, where $\mathbf{K}$ is a function of f, we can measure the validity of f as

$$v = \sum_i \exp\left(\frac{1 - \lambda_1^{(i)}/\lambda_2^{(i)}}{\tau}\right), \tag{2}$$

where τ is a temperature hyper-parameter. Intuitively, the above formula is close to one when $\lambda_1^{(i)}/\lambda_2^{(i)}$ is close to one, and decreases exponentially as $\lambda_1^{(i)}/\lambda_2^{(i)}$ increases.

We sample a set of candidate focal lengths and evaluate them using Eqn. 2. We choose the candidate with the highest value (2) as the final estimate. This method can be easily generalized to images with different focal lengths (see Sec. B.2 in the supplementary for details). After estimating the focal length, we transform the keypoints, fundamental matrices, and homography matrices using the estimated intrinsic matrices so all components are calibrated.

3.2. First-order Optimization

Levenberg-Marquardt Most of the previous SfM methods use quasi second-order methods such as Levenberg-Marquardt (LM) for optimization. Almost all methods use LM for bundle adjustment [60], and GLOMAP [44] uses it for global translation alignment. Levenberg-Marquardt is a Gauss-Newton method that first approximates the Hessian with Jacobians of the residuals, and then solves a large linear system to compute the update direction. Techniques like the Schur's complement and sparse Cholesky decomposition are used to improve computational efficiency by exploiting the sparsity of the system [1, 60].

While second-order methods can converge quickly near the optimum, they suffer from poor scalability. Even with the Schur complement method, each step requires solving a reduced linear system whose size grows quadratically with the number of images. In practice, this results in a cubic-time cost in the number of images, which dominates the computation of the update direction. In very large and densely-connected problems, this becomes prohibitively expensive, even with GPU acceleration. To address this, many frameworks employ the preconditioned conjugate gradient (PCG) method, which approximately solves the reduced linear system at a per-iteration cost that scales only quadratically in the number of images. However, PCG slows convergence and introduces implementation complexity.

First-order Optimization On the other hand, first-order optimization methods are prevalent in other fields of computer vision, thanks to the success of deep learning. Optimizing a neural network with a large number of parameters is only tractable with first-order methods, and many adaptive gradient methods exist that accelerate the naive gradient descent. In this paper, we investigate the use of first-order optimization methods in SfM.

Efficiency Unfortunately, first-order methods usually converge much more slowly than Gauss-Newton methods in terms of the decrease in loss at each iteration (i.e., they require more iterations). The key to the success of using first-order optimization in SfM is to make the computation of each step as fast as possible. We identify the two most important speed bottlenecks:

1. *3D points*: One of the most important components of a typical SfM pipeline is bundle adjustment [60], which jointly optimizes camera poses and 3D points. In practice, the number of 3D points is usually orders-of-magnitude larger than the number of image pairs. To make Gauss-Newton tractable in this setting, methods employ the Schur complement method [60] to first eliminate the 3D point variables to form a reduced system independent of the number of points, and then reciver the 3D points via back-substitution. In this stage, solving the reduced system usually dominates the compute time. However, if we switch to gradient descent, the main bottleneck becomes computing the forward and backward passes for the 3D point parameters. To address this, we design all the optimization problems in our method (Sec. 3.3) so that at each iteration, the computation complexity is independent of the number of 3D points.
2. *Kernel implementation*: The optimization problems that FASTMAP solves can be easily implemented using modern Autograd frameworks such as PyTorch. These libraries are highly optimized for large-scale deep learning applications that involve a lot of linear operations on large tensors. In our case, most of the operations are relatively small (e.g., 3×3 matrix multiplication), and naively implementing everything with high-level PyTorch optimizations induces significant kernel launching and data movement overhead. We solve this problem through kernel fusion (Sec. 3.4), which eliminates most of the overhead and increases GPU utilization.

Section 3.3 introduces all the optimization problems present in our method. They are chosen such that the computational complexity of each step is independent of the number of 3D points. In Sec. 3.4, we describe the kernel fusion technique to fully exploit the power of GPUs for even more speedup.

3.3. Optimization Formulations

Here, we introduce the optimization-based formulations of estimating and finetuning the global poses.

Global Rotation With the estimated intrinsics, we can decompose the fundamental and homography matrices into relative rotations and translations [20, 37]. Given the set of image pairs $\mathcal{P} = \{(i, j)\}$ and the corresponding relative rotation matrices $\{\mathbf{R}^{i \to j}\}_{(i,j) \in \mathcal{P}}$, FASTMAP next estimates

the world-to-camera global rotation $\mathbf{R}^{(i)}$ matrix for each image i. We formulate this as an optimization of a loss defined over all image pairs $\mathcal{P} = \{(i, j)\}$

$$\mathcal{L}_{\text{R}} = \frac{1}{|\mathcal{P}|} \sum_{(i,j)\in\mathcal{P}} d(\mathbf{R}^{(j)}, \mathbf{R}^{i\to j}\mathbf{R}^{(i)}), \tag{3}$$

where $d(\cdot, \cdot)$ is the geodesic distance between rotations

$$d(\mathbf{R}, \mathbf{R}') = \cos^{-1}\left(\frac{\text{Tr}(\mathbf{R}^T\mathbf{R}') - 1}{2}\right). \tag{4}$$

For simplicity, we parameterize the global rotation matrices $\mathbf{R}_i$ using a differentiable 6D representation [72].

Unfortunately, directly optimizing the above objective from random initialization of $\mathbf{R}_i$ is prone to local minima. We use a slightly modified version of the method proposed by Martinec and Pajdla [38] to obtain a good initialization. The basic idea of the method is that although the column vectors in a rotation matrix are constrained by orthogonality, each column vector alone is only subject to a unit length constraint. If we consider one column at a time, we can formulate the optimization as as a least squares problem and solve it using SVD. See the supplementary material (Sec. B.3.1) for details of this initialization scheme.

Global Translation After global rotation alignment, we re-estimate the relative translations between image pairs (Sec. B.5). The next step is to utilize these relative translations to estimate the 3D coordinates of the camera centers in a common (world) coordinate frame, up to a similarity transformation. This step is usually called *translation averaging*. It is notoriously susceptible to noise, and making it robust to all kinds of scenarios is the focus of many global SfM papers [23, 43, 44, 67, 73]. However, this is not the focus of our paper and so we choose a relatively simple method to tackle this problem, which we find to be sufficient for most of the scenes we evaluate on. A more robust design for this stage is left for future work.

Given world-to-camera rotations $\{\mathbf{R}_i\}_{1\le i\le N}$ for N images and unit-length relative translations $\{\boldsymbol{t}^{i\to j}\}_{(i,j)\in\mathcal{P}}$ for a set of image pairs $\mathcal{P}$, we compute the normalized vector from the camera centers of image i to j in world coordinates

$$\boldsymbol{o}^{i\to j} = -\mathbf{R}_j^\top \boldsymbol{t}^{i\to j}. \tag{5}$$

We estimate the camera locations $\{\boldsymbol{o}_i\}_{1\le i\le N}$ in the world frame by minimizing the error between the normalized relative translation $\frac{\boldsymbol{o}_j - \boldsymbol{o}_i}{\|\boldsymbol{o}_j - \boldsymbol{o}_i\|_2}$ and the target $\boldsymbol{o}^{i\to j}$ above with gradient descent

$$\mathcal{L}_{\text{t}} = \frac{1}{|\mathcal{P}|} \sum_{(i,j)\in\mathcal{P}} \left\| \frac{\boldsymbol{o}_j - \boldsymbol{o}_i}{\|\boldsymbol{o}_j - \boldsymbol{o}_i\|_2} - \boldsymbol{o}^{i\to j} \right\|_1. \tag{6}$$

Unlike global rotation optimization, this objective can often be effectively optimized from a random initialization. GLOMAP [44] makes a similar observation, but it optimizes poses and 3D points jointly, and is much more computationally expensive.

Although random initialization works surprisingly well for the objective in Eqn. 6, it occasionally produces a small number of outliers. To deal with this, we perform multiple independent runs from different random initializations and merge the solutions as the initialization for the final optimization loop. Please see Sec. B.5.2 for details.

Epipolar Adjustment A typical SfM pipeline relies on *bundle adjustment* (BA) [60] to jointly refine the camera poses and inferred 3D points. Directly implementing BA with first-order optimizers is computationally expensive when the number of points is large. Instead, we refine the poses from previous stages using *re-weighting epipolar adjustment*, an optimization method [50] for which the computational complexity in each iteration is linear only in the number of image pairs, not in the number of points.

In relative translation re-estimation, we obtain a set of image pairs with number of inliers above some threshold. We denote the set of such image pairs as $\mathcal{P} = \{(i_n, j_n)\}_{1\le n\le|\mathcal{P}|}$ (abusing the notation for the original set of images above), where i_n and j_n are the indices of the first and second images in the pair. For an image pair $(i_n, j_n) \in \mathcal{P}$, we represent the set of point pairs as $\mathcal{Q}_n = \{(\boldsymbol{x}^{(1)}_{nm}, \boldsymbol{x}^{(2)}_{nm}) \in \mathbb{R}^2 \times \mathbb{R}^2\}_{1\le m\le|\mathcal{Q}_n|}$, and let $\tilde{\mathcal{Q}}_n = \{(\tilde{\boldsymbol{x}}^{(1)}_{nm}, \tilde{\boldsymbol{x}}^{(2)}_{nm}) \in \mathbb{P}^2 \times \mathbb{P}^2\}_{1\le m\le|\tilde{\mathcal{Q}}_n|}$ be the set of point pairs in normalized homogeneous coordinates.

Using estimated initializations from the previous stages, we optimize over the world-to-camera global rotations and translations to minimize the absolute epipolar error

$$\mathcal{L}_{\text{e}} = \frac{1}{Z} \sum_{n=1}^{|\mathcal{P}|} \sum_{m=1}^{|\tilde{\mathcal{Q}}_n|} |\tilde{\boldsymbol{x}}^{(2)\top}_{nm} \mathbf{E}_n \tilde{\boldsymbol{x}}^{(1)}_{nm}|, \tag{7}$$

where $Z = \sum_{n=1}^{|\mathcal{P}|} |\tilde{\mathcal{Q}}_n|$ is the total number of point pairs, and $\mathbf{E}_n$ is the essential matrix computed from the global rotations and translations for images i_n and j_n.

Evaluating Eqn. 7 for every iteration is expensive because it involves every point pair. However, if we replace the cost terms with the L2 loss (as in Rodriguez et al. [49]), the overall objective can be re-organized to aggregate terms that involve point pairs shared by the same image pair into a compact quadratic form (see Sec. B.6)

$$\mathcal{L}_{\text{e}} = \frac{1}{Z} \sum_{n=1}^{|\mathcal{P}|} \sum_{m=1}^{|\tilde{\mathcal{Q}}_n|} (\tilde{\boldsymbol{x}}^{(2)\top}_{nm} \mathbf{E}_n \tilde{\boldsymbol{x}}^{(1)}_{nm})^2 = \frac{2}{Z} \sum_{n=1}^{|\mathcal{P}|} \boldsymbol{e}_n^\top \mathbf{W}_n \boldsymbol{e}_n, \tag{8}$$

where $\boldsymbol{e}_n = \text{flatten}(\mathbf{E}_n) \in \mathbb{R}^9$, and $\mathbf{W}_n \in \mathbb{R}^{9\times 9}$ is a matrix computed from all the point pairs in $\tilde{\mathcal{Q}}_n$. Note that only $\boldsymbol{e}_n$ is a function of the parameters to be optimized. The matrices $\mathbf{W}_n$ can be pre-computed for each image pair before

optimization. With the precomputed $\mathbf{W}_n$, the cost of each optimization step is linear in the number of image pairs.

The expedience of Eqn. 8 comes with the side effect that the L2 loss is sensitive to outliers. We propose to robustify the loss function but still preserve the benefit of pre-computation with *iterative re-weighted least squares (IRLS)*. The intuition is that if we have an initialization close to the optimum, the L1 loss, which is more robust to outliers, can be approximated by a weighted L2 loss. In other words, for some differentiable computed scalar value z, we have $z^2 \approx z^2/|\hat{z}|$, where $\hat{z}$ is the value of z at initialization. In our case, after global translation alignment, we already have a good initialization of global poses, so we can compute the absolute epipolar error $|\hat{\epsilon}_{nm}|$ for each point pair, and use it to weight the L2 loss used above to get an approximate robust L1 loss

$$\hat{\mathcal{L}}_{\text{e}} = \frac{1}{Z} \sum_{n=1}^{|\mathcal{P}|} \sum_{m=1}^{|\tilde{\mathcal{Q}}_n|} \frac{(\tilde{\boldsymbol{x}}_{nm}^{(2)\top} \mathbf{E}_n \tilde{\boldsymbol{x}}_{nm}^{(1)})^2}{|\hat{\epsilon}_{nm}|} = \frac{2}{Z} \sum_{n=1}^{|\mathcal{P}|} \boldsymbol{e}_n^\top \hat{\mathbf{W}}_n \boldsymbol{e}_n, \tag{9}$$

where $\hat{\mathbf{W}}_n$ is similar to $\mathbf{W}_n$ in Eqn. 8, but computed from $\tilde{\mathcal{Q}}_n$ weighted by $|\hat{\epsilon}_{nm}|$.

After a round of optimization, we can better approximate the L1 loss by re-computing the weights and then doing another optimization loop. To further reduce the impact of outliers, we periodically filter out point pairs with large epipolar error. We start from a relatively large initial threshold and gradually decrease it to a pre-determined minimum.

The above optimization problem only involves camera poses. We can also optimize the focal lengths by incorporating them into the computation of the essential matrices (in this case, the results are actually fundamental matrices).

3.4. Kernel Fusion

A standard way to implement gradient descent in the above algorithms is to use the Autograd feature in modern deep learning libraries such as PyTorch [46]. However, the tensors in our setting are mostly batches of small matrices and vectors of shapes $B \times 3 \times 3$ and $B \times 3$, where B is the number of image pairs, and a naive PyTorch implementation introduces some significant bottlenecks:

1. *Kernel Launching Overhead:* When the scene is relatively small (i.e., the number of image pairs is small), the kernel launching overhead dominates the running time and is limited by the CPU speed.
2. *Data Movement:* Computing the objective and gradients involves a series of small operations (e.g., 3×3 matrix multiplication and cross product). Each operation involves reading the data from the high-latency global memory to the fast on-chip shared memory and writing back when finished. his leads to substantial inefficiency and makes the computation predominantly memory-bound.
3. *Kernel Design:* PyTorch kernels are optimized for deep learning workload, which usually consists of linear operations on large tensors. These kernels can lead to suboptimal performance when applied to tensors without the assumed shapes.

To address this problem, we fuse all the operations for computing the gradients, including forward and backward passes, into one single custom CUDA kernel. This introduces challenges in shared-memory management when the computation involves many small operations. However, since almost all input and intermediate tensors have one of the three shapes ($B \times 3 \times 3$, $B \times 3$, or simply B), we can efficiently reuse shared-memory slots to limit the reduction in thread occupancy. In Tab. 4 we show that the fused kernel can be more than two orders-of-magnitude faster than a naive PyTorch implementation under different scene sizes and hardware settings. Please refer to the ablation study (Sec. 4.3) for a more detailed analysis.

4. Experiments

4.1. Setup

We compare FASTMAP with two state-of-the-art methods: COLMAP [53] (commit `c4a3b30`) and GLOMAP [44] (commit `01060b4`), both with GPU-accelerated Ceres [1] solver enabled. For all three methods, we use the COLMAP image matching system. We use shared intrinsics if all images in a scene are from the same camera, and leave all other hyper-parameters at their default values in COLMAP and GLOMAP. We run all three methods on a machine with a single A6000 (Ampere) GPU and an AMD EPYC 9274F CPU (Zen 4) with 24 cores / 48 threads. By default, FASTMAP uses 2 CPU threads, whereas COLMAP and GLOMAP use all 48 threads. We report more detailed speed comparisons with different hardware configurations in Tab. 6 in the appendix.

Datasets We focus on the case of high-overlap images densely connected by feature matching, and evaluate the three methods on eight datasets: MipNeRF360 [4], Tanks and Temples [29], ZipNeRF [5], NeRF-OSR [51], DroneDeploy [47], Mill-19 [61], Urbanscene3D [34], and Eyeful Tower [69]. They cover a wide range of real-world scenarios and camera trajectory patterns. The number of images per scene ranges from around 200 to 6000. Sec. B.8 provides a more detailed discussion of the GT poses provided along with these datasets.

Metrics We report wall-clock time in seconds, excluding the time required for feature extraction and matching (identical for all three methods and dominated by the SfM backend time for large scenes). We evaluate pose accuracy using the standard metrics [11, 44, 53]: ATE, RRA@δ, RTA@δ, and AUC@δ. For some of the scenes, we also evaluate the

		time (sec)			ATE↓			RTA@3↑			AUC-R&T @ 3 ↑			RTA@1↑			AUC-R&T @ 1 ↑		
	n_imgs	FastMap	GLOMAP	COLMAP	FastMap	GLOMAP	COLMAP	FastMap	GLOMAP	COLMAP	FastMap	GLOMAP	COLMAP	FastMap	GLOMAP	COLMAP	FastMap	GLOMAP	COLMAP
mipnerf360 (9)	215.6	33	165	503	4.2e-4	3.3e-5	5.8e-5	99.9	100.0	100.0	97.4	98.2	97.2	99.8	100.0	99.7	92.4	94.6	91.9
tnt_advanced (6)	337.8	61	357	1016	6.4e-3	1.2e-2	1.2e-3	71.4	79.1	98.5	42.6	75.3	94.8	42.3	77.5	97.0	16.7	69.8	90.0
tnt_intermediate (8)	268.6	35	314	833	7.8e-5	1.9e-5	2.6e-4	99.9	100.0	99.8	94.1	99.0	98.9	99.3	99.9	99.5	83.1	96.9	97.3
tnt_training (7)	470.1	63	515	2751	3.0e-3	1.1e-2	3.0e-4	87.8	88.7	99.9	77.2	87.9	99.5	82.1	88.6	99.9	60.5	86.3	98.7
nerf_osr (8)	402.8	50	324	3163	1.6e-3	1.1e-3	1.3e-3	91.7	92.0	92.1	70.9	71.9	71.7	71.1	71.9	71.7	43.2	45.2	44.7
drone_deploy (9)	524.7	91	365	3352	4.9e-3	4.3e-3	2.0e-3	97.9	98.2	91.3	79.2	81.1	65.2	89.6	91.5	73.5	50.4	53.5	40.2
zipnerf (4)	1527.2	119	690	3820	3.0e-3	7.1e-3	3.4e-4	99.0	98.1	99.7	92.6	96.6	98.1	97.4	98.0	99.6	81.4	93.6	95.2
urban_scene (3)	3824	515	3664	61622	1.7e-5	1.4e-5	1.4e-5	99.9	99.9	100.0	95.3	97.0	97.0	99.5	99.6	99.6	86.3	91.2	91.3
mill19_building	1920	258	6289	27080	3.0e-4	1.3e-2	1.9e-5	99.9	0.1	99.9	95.5	0.0	95.6	99.3	0.0	99.3	87.0	0.0	87.4
mill19_rubble	1657	240	2849	12153	3.6e-5	6.4e-5	3.4e-5	99.9	99.8	99.9	93.6	94.5	94.6	98.6	98.6	98.7	81.6	84.7	84.8
eyeful_apartment	3804	549	5905	185361	2.8e-3	9.4e-3	2.2e-3	86.8	75.0	90.2	45.5	50.5	62.0	51.1	61.3	71.7	6.4	18.2	21.9
eyeful_kitchen	6042	2202	22884	timeout	3.1e-3	7.4e-3	-	85.0	59.9	-	38.1	41.2	-	46.7	51.7	-	4.6	14.4	-

Table 1. Speed and pose accuracy of FastMap, GLOMAP, and COLMAP. All three methods are accelerated by GPU. For datasets with more than two scenes, we denote the average metrics as `dataset-name(#scenes)`. In particular, Tanks and Temples [29] has three official splits, and we do the averaging separately for them. Mill-19 [61] and Eyeful Tower [69] scenes are listed separately. Metrics are color-coded in green, with color changes if the percentage gap >2% or ATE ratio >1.5. Red denotes complete failures and gray means the method did not finish in a week. Note the significant speedup of FastMap vs. previous work, especially on larger scenes.

		Absolute PSNR ↑			Relative to COLMAP	
		FastMap	GLOMAP	COLMAP	FastMap	GLOMAP
m360_bicycle	Zip-NeRF	25.60	25.78	25.86	-0.26	-0.08
	+ CamP	26.21	26.36	26.41	-0.21	-0.05
	GSplat	25.51	25.59	25.62	-0.11	-0.03
m360_bonsai	Zip-NeRF	34.78	34.91	34.47	0.31	0.44
	+ CamP	35.26	35.32	35.37	-0.12	-0.05
	GSplat	32.32	32.29	31.49	0.84	0.81
m360_counter	Zip-NeRF	28.97	28.95	29.18	-0.21	-0.23
	+ CamP	29.09	29.18	29.29	-0.20	-0.12
	GSplat	28.99	29.06	29.02	-0.02	0.04
m360_flowers	Zip-NeRF	22.05	22.29	21.89	0.15	0.40
	+ CamP	23.53	23.47	23.27	0.25	0.20
	GSplat	21.74	21.79	21.59	0.15	0.20
m360_garden	Zip-NeRF	28.10	28.20	28.20	-0.11	0.00
	+ CamP	28.54	28.49	28.54	0.00	-0.05
	GSplat	27.61	27.67	27.72	-0.11	-0.05
m360_kitchen	Zip-NeRF	32.29	32.43	32.31	-0.02	0.12
	+ CamP	32.47	32.19	32.21	0.27	-0.02
	GSplat	31.36	31.62	31.58	-0.22	0.05
m360_room	Zip-NeRF	32.81	32.94	32.93	-0.12	0.01
	+ CamP	32.51	32.48	32.44	0.07	0.04
	GSplat	31.77	31.71	31.67	0.11	0.04
m360_stump	Zip-NeRF	27.34	27.41	27.43	-0.09	-0.02
	+ CamP	28.10	28.03	28.03	0.07	0.00
	GSplat	26.97	26.89	26.84	0.13	0.05
m360_treehill	Zip-NeRF	23.73	24.05	24.04	-0.31	0.01
	+ CamP	25.73	25.74	25.99	-0.26	-0.25
	GSplat	22.71	22.88	22.80	-0.08	0.08
tnt_training (7)	InstantNGP	20.73	19.37	21.05	-0.32	-1.68
	GSplat	23.22	21.54	24.19	-0.97	-2.65
tnt_intermediate (8)	InstantNGP	22.29	22.51	22.38	-0.09	0.13
	GSplat	24.24	25.28	25.24	-1.00	0.03
tnt_advanced (6)	InstantNGP	16.59	16.94	17.55	-0.95	-0.60
	GSplat	18.82	18.74	22.06	-3.24	-3.32

Table 2. Novel view synthesis evaluation on MipNeRF360 [4] and Tanks and Temples [29]. Results for MipNeRF360 are listed separately for each scene, and those for Tanks and Temples are averaged over all scenes in each of the three splits. The color changes only if the PSNR difference >0.25. We report results for Zip-NeRF, Zip-NeRF + CamP optimization, and Gaussian Splatting.

novel view synthesis quality of NeRF [40] and Gaussian Splatting [27] trained on the output poses, intrinsics, and triangulated point clouds.

	tnt_training (7)			tnt_intermediate (8)			tnt_advanced (6)		
	ATE	RTA@5	RRA@5	ATE	RTA@5	RRA@5	ATE	RTA@5	RRA@5
ACE-Zero [7]	1.2e-2	72.9	73.9	8.0e-3	74.0	67.5	2.8e-2	19.1	22.9
MAST3R-SfM [11]	6.2e-3	64.9	56.2	7.2e-3	57.5	50.8	2.0e-2	36.5	38.8
GLOMAP	1.1e-2	88.8	89.3	1.9e-5	100.0	100.0	1.2e-2	79.3	80.5
FastMap	3.2e-3	88.8	95.8	9.2e-5	100.0	100.0	6.8e-3	70.5	82.1

Table 3. Comparison to learning-based SfM on Tanks and Temples [29]. We use COLMAP poses from Kulhanek and Sattler [31] as reference. We average numbers for scenes in each split.

4.2. Analysis

Pose accuracy Table 1 compares the three methods in average camera pose metrics on all the datasets (we include per-scene evaluation results in the supplementary material). In general, our method is much faster than both GLOMAP and COLMAP. The speedup over GLOMAP is less dramatic when there are only a few hundred images (e.g., MipNeRF360), but it can be about 10× faster on scenes with several thousand images (e.g., Urbanscene3D, Mill-19, and Eyeful Tower). On most datasets, FastMap is on par with GLOMAP and COLMAP in terms of RTA@3. There is a more prominent difference for stricter metrics (RTA@1, AUC@3, AUC@1). This shows that while FastMap succeeds in recovering the overall structures of camera trajectories, it does achieve the highest level of precision when the error is reduced to one or two degrees.

None of the methods are perfect. FastMap performs particularly bad on the Advanced split of Tanks and Temples, probably because there are many erroneous matches due to repetitive patterns and symmetric structures in the scenes. This is a well-known problem of global SfM (i.e., GLOMAP also suffers a significant drop in performance compared to COLMAP), and incremental SfM methods like COLMAP are more robust in these settings. On the building scene of the Mill-19 dataset, GLOMAP fail catastrophically, however FastMap and COLMAP remain highly accurate. On DroneDeploy, none of the three methods is very good in terms of AUC@1 and RTA@1.

In Tab. 3, we compare FASTMAP to two representative learning-based methods, ACE-Zero [7] and MAST3R-SfM [11], where we include the results from Duisterhof et al. [11, Tab. 10]. Both methods perform significantly worse than FASTMAP and GLOMAP. This indicates that while learning-based methods are promising, they still lag far behind traditional methods in terms of pose accuracy.

Novel view synthesis Table 2 evaluates the quality of novel view synthesis on MipNeRF360 and Tanks and Temples when using FASTMAP, COLMAP, and GLOMAP to estimate the camera poses. We use ZipNeRF [5], a very high-quality NeRF method, for MipNeRF360, and use Instant-NGP [42] for Tank and Temples, which offers a better trade-off between quality and speed. We also evaluate the performance of Gaussian Splatting [27] on both datasets.

While FASTMAP lags behind GLOMAP and COLMAP on most MipNeRF360 scenes, the PSNR difference is within 0.5. On Tanks and Temples, FASTMAP performs on par with GLOMAP, but both are worse than COLMAP. Here, again, the lower pose accuracy of FASTMAP under the strictest metrics does not prevent FASTMAP poses from yielding competitive PSNR. These results suggest that pose accuracy under a strict metric could be a misleading proxy for downstream view synthesis quality, and vice versa.

We also investigate the impact of different SfM poses on rendering with CamP [45], which simultaneously optimizes the radiance field and refines the camera poses. We include the results in Tab. 2 for comparison. In general, CamP improves the PSNR for all the three methods, and for some scenes (e.g., flowers, garden, kitchen, etc.) the gap in rendering quality is closed and sometimes even reversed.

4.3. Ablations

Kernel fusion In Tab. 4, we show the timing comparison of naive PyTorch and kernel fusion approaches to implementing the first-order optimization of epipolar adjustment. Profiling and comparing the CPU and GPU times for these two approaches is challenging due to the various forms of execution overlap. Instead, we directly compare the wall-clock time on different hardware setups. On small-to-medium scale scenes (i.e., 5k and 50k image pairs), the running time is severely bottlenecked by CPU overhead, and using a slower CPU can significantly impact the speed. Interestingly, the PyTorch version is faster on the less-powerful 2080 Ti than A6000, reflecting that its kernel implementation cannot fully utilize the power of high-end GPUs and is not suitable in our case. Across all the three hardware settings and scene sizes, our fused kernel implementation is around $20\times$ to $90\times$ faster than the naive PyTorch version.

Distortion estimation is one of the first steps of FASTMAP, and its accuracy is critical to the final performance. Table 5 presents the performance of FASTMAP with and without distortion estimation on the Intermediate split of Tanks and Temples. Without distortion estimation, results drop, sometimes catastrophically. We provide in Sec. B.1 an additional insight into the effect of distortion estimate on the immediately following step of focal length estimation.

# pairs	CPU	GPU	torch (ms)	fused (ms)	speedup
5k	4.05GHz	A6000	2.83	**0.05**	56×
	2.2GHz	A6000	9.82	**0.11**	89×
	2.2GHz	2080 Ti	9.41	**0.11**	85×
50k	4.05GHz	A6000	8.20	**0.14**	58×
	2.2GHz	A6000	12.47	**0.20**	62×
	2.2GHz	2080 Ti	11.93	**0.27**	44×
500k	4.05GHz	A6000	65.94	**1.16**	56×
	2.2GHz	A6000	69.31	**1.21**	53×
	2.2GHz	2080 Ti	44.32	**1.92**	23×

Table 4. Effect of kernel fusion for epipolar adjustment under different hardware settings and scene sizes (#pairs refers to the number of image pairs). Interestingly, the naive PyTorch implementation is faster on 2080 Ti than A6000 with 500k image pairs, showing that the native PyTorch kernel implementation cannot fully utilize the GPU for our problems. Note that the performance of the same CPU or GPU can be slightly different on different machines.

	AUC@3		AUC@10		RTA@3		RTA@10	
	w/	w/o	w/	w/o	w/	w/o	w/	w/o
Family	**95.1**	72.8	**98.5**	91.8	**100.0**	99.9	**100.0**	100.0
Francis	**95.5**	71.1	**98.6**	91.2	**99.9**	99.6	**100.0**	99.9
Horse	**96.8**	76.8	**99.0**	93.0	**100.0**	100.0	**100.0**	100.0
Lighthouse	**90.7**	4.6	**97.1**	42.2	**99.6**	46.5	**100.0**	98.5
M60	**95.6**	28.3	**98.7**	72.9	**99.9**	85.9	**100.0**	99.7
Panther	**93.0**	12.1	**97.9**	64.2	**99.9**	78.3	**100.0**	99.9
Playground	**84.6**	2.2	**95.4**	14.4	**100.0**	15.2	**100.0**	51.9
Train	**92.4**	54.2	**97.7**	86.0	**99.9**	99.6	**100.0**	99.9

Table 5. Effect of camera distortion estimation on pose accuracy.

Others Due to page limit, we put some other ablation results in Sec. B.9, including those for track completion, multiple initialization, and epipolar adjustment.

5. Limitations and Conclusions

We introduce FASTMAP, a new structure from motion method focused on simplicity and speed. Contrary to the common practice in other SfM systems, FASTMAP uses first-order optimization extensively and is much faster than state-of-the-art methods (COLMAP and GLOMAP), while achieving comparable performance on pose accuracy and novel view synthesis quality. These improvements do come with a few drawbacks. For example, FASTMAP might fail on scenes where there are a lot of degenerate motions, and is more sensitive to incorrect matching induced by repetitive patterns and symmetric structures when compared to GLOMAP (please refer to the appendix for a more detailed discussion of limitations). Nevertheless, we believe it is an important step towards highly efficient camera pose estimation for real-world 3D data acquisition at scale.

References

[1] Sameer Agarwal, Keir Mierle, and The Ceres Solver Team. Ceres Solver, 2023. 2, 4, 6

[2] Daniel Barath, Tekla Toth, and Levente Hajder. A minimal solution for two-view focal-length estimation using two affine correspondences. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 3

[3] João Pedro Barreto and Kostas Daniilidis. Fundamental matrix for cameras with radial distortion. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2005. 3

[4] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 6, 7, 16

[5] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Zip-NeRF: Anti-aliased grid-based neural radiance fields. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2023. 6, 8, 16

[6] Shariq Farooq Bhat, Reiner Birkl, Diana Wofk, Peter Wonka, and Matthias Müller. Zoedepth: Zero-shot transfer by combining relative and metric depth. *arXiv preprint arXiv:2302.12288*, 2023. 1

[7] Eric Brachmann, Jamie Wynn, Shuai Chen, Tommaso Cavallari, Áron Monszpart, Daniyar Turmukhambetov, and Victor Adrian Prisacariu. Scene coordinate reconstruction: Posing of image collections via incremental learning of a relocalizer. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024. 2, 7, 8

[8] Hainan Cui, Xiang Gao, Shuhan Shen, and Zhanyi Hu. HSfM: Hybrid structure-from-motion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[9] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13142–13153, 2023. 1

[10] C Brown Duane. Close-range camera calibration. *Photogramm. Eng*, 37(8), 1971. 3

[11] Bardienus Duisterhof, Lojze Zust, Philippe Weinzaepfel, Vincent Leroy, Yohann Cabon, and Jerome Revaud. MASt3R-SfM: A fully-integrated solution for unconstrained structure-from-motion. *arXiv preprint arXiv:2409.19152*, 2024. 1, 3, 6, 7, 8

[12] Bastian Erdnüß. A review of the one-parameter division undistortion model. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 1, 2021. 3

[13] Olivier Faugeras. *Three-dimensional computer vision: A geometric viewpoint*. MIT Press, 1993. 3

[14] Andrew W Fitzgibbon. Simultaneous linear estimation of multiple view geometry and lens distortion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001. 3

[15] Jose L Gómez, Manuel Silva, Antonio Seoane, Agnès Borrás, Mario Noriega, Germán Ros, Jose A Iglesias-Guitian, and Antonio M López. All for one, and one for all: Urbansyn dataset, the third musketeer of synthetic driving scenes. *Neurocomputing*, 637:130038, 2025. 1

[16] Venu Madhav Govindu. Combining two-view constraints for motion estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001. 2

[17] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, et al. Kubric: A scalable dataset generator. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3749–3761, 2022. 1

[18] Adam W Harley, Yang You, Xinglong Sun, Yang Zheng, Nikhil Raghuraman, Yunqi Gu, Sheldon Liang, Wen-Hsuan Chu, Achal Dave, Pavel Tokmakov, et al. Alltracker: Efficient dense point tracking at high resolution. *arXiv preprint arXiv:2506.07310*, 2025. 1

[19] Richard Hartley. Extraction of focal lengths from the fundamental matrix. *Unpublished manuscript*, 2, 1993. 3

[20] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, 2003. 3, 4

[21] Richard Hartley, Jochen Trumpf, Yuchao Dai, and Hongdong Li. Rotation averaging. *International Journal on Computer Vision*, 103, 2013. 2

[22] Jose Henrique Brito, Roland Angst, Kevin Koser, and Marc Pollefeys. Radial distortion self-calibration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. 3

[23] Nianjuan Jiang, Zhaopeng Cui, and Ping Tan. A global linear method for camera pose registration. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2013. 2, 5

[24] Kenichi Kanatani and Chikara Matsunaga. Closed-form expression for focal lengths from the fundamental matrix. In *Proceedings of the Asian Conference on Computer Vision*, 2000. 3

[25] Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker: It is better to track together. In *European conference on computer vision*, pages 18–35. Springer, 2024. 1

[26] Bingxin Ke, Anton Obukhov, Shengyu Huang, Nando Metzger, Rodrigo Caye Daudt, and Konrad Schindler. Repurposing diffusion-based image generators for monocular depth estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9492–9502, 2024. 1

[27] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4), 2023. 7, 8

[28] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2

[29] 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), 2017. 6, 7

[30] Viktor Kocur, Daniel Kyselica, and Zuzana Kukelova. Robust self-calibration of focal lengths from the fundamental matrix. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 3

[31] Jonas Kulhanek and Torsten Sattler. NeRFBaselines: Consistent and reproducible evaluation of novel view synthesis methods. *arXiv preprint arXiv:2406.17345*, 2024. 7, 16

[32] Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Grounding image matching in 3d with MASt3R. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024. 3

[33] Jiahao Li, Hao Tan, Kai Zhang, Zexiang Xu, Fujun Luan, Yinghao Xu, Yicong Hong, Kalyan Sunkavalli, Greg Shakhnarovich, and Sai Bi. Instant3d: Fast text-to-3d with sparse-view generation and large reconstruction model. *arXiv preprint arXiv:2311.06214*, 2023. 1

[34] Liqiang Lin, Yilin Liu, Yue Hu, Xingguang Yan, Ke Xie, and Hui Huang. Capturing, reconstructing, and simulating: the UrbanScene3D dataset. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022. 6

[35] Philipp Lindenberger, Paul-Edouard Sarlin, Viktor Larsson, and Marc Pollefeys. Pixel-perfect structure-from-motion with featuremetric refinement. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021. 16

[36] Yuanxun Lu, Jingyang Zhang, Tian Fang, Jean-Daniel Nahmias, Yanghai Tsin, Long Quan, Xun Cao, Yao Yao, and Shiwei Li. Matrix3d: Large photogrammetry model all-in-one. *arXiv preprint arXiv:2502.07685*, 2025. 3

[37] Ezio Malis and Manuel Vargas. *Deeper understanding of the homography decomposition for vision-based control*. PhD thesis, Inria, 2007. 4

[38] Daniel Martinec and Tomas Pajdla. Robust rotation and translation estimation in multiview reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007. 2, 5, 12

[39] Lukas Mehl, Jenny Schmalfuss, Azin Jahedi, Yaroslava Nalivayko, and Andrés Bruhn. Spring: A high-resolution high-detail dataset and benchmark for scene flow, optical flow and stereo. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4981–4991, 2023. 1

[40] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1), 2021. 7

[41] Pierre Moulon, Pascal Monasse, Romuald Perrot, and Renaud Marlet. OpenMVG: Open multiple view geometry. In *Proceedings of the International Workshop on Reproducible Research in Pattern Recognition*, 2017. 2

[42] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (TOG)*, 41(4), 2022. 8

[43] Onur Ozyesil and Amit Singer. Robust camera location estimation by convex programming. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2, 5

[44] Linfei Pan, Daniel Barath, Marc Pollefeys, and Johannes Lutz Schönberger. Global structure-from-motion revisited. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024. 1, 2, 4, 5, 6, 14

[45] Keunhong Park, Philipp Henzler, Ben Mildenhall, Jonathan T. Barron, and Ricardo Martin-Brualla. CamP: Camera preconditioning for neural radiance fields. *ACM Transactions on Graphics*, 2023. 8

[46] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 2, 6

[47] Nicholas Pilkington. DroneDeploy NeRF dataset. https://github.com/nickponline/dd-nerf-dataset, 2022. 6

[48] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE transactions on pattern analysis and machine intelligence*, 44(3):1623–1637, 2020. 1

[49] Antonio L Rodriguez, Pedro E López-de Teruel, and Alberto Ruiz. GEA optimization for live structureless motion estimation. In *Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, 2011. 5

[50] Antonio L Rodríguez, Pedro E López-de Teruel, and Alberto Ruiz. Reduced epipolar cost for accelerated incremental sfm. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011. 5

[51] Viktor Rudnev, Mohamed Elgharib, William Smith, Lingjie Liu, Vladislav Golyanik, and Christian Theobalt. Nerf for outdoor scene relighting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022. 6

[52] Paul-Edouard Sarlin, Mihai Dusmanu, Johannes L Schönberger, Pablo Speciale, Lukas Gruber, Viktor Larsson, Ondrej Miksik, and Marc Pollefeys. LaMAR: Benchmarking localization and mapping for augmented reality. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022. 2, 17

[53] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 2, 3, 6, 14

[54] Thomas Schops, Johannes L Schonberger, Silvano Galliani, Torsten Sattler, Konrad Schindler, Marc Pollefeys, and Andreas Geiger. A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2, 16

[55] Yichun Shi, Peng Wang, Jianglong Ye, Mai Long, Kejie Li, and Xiao Yang. Mvdream: Multi-view diffusion for 3d generation. *arXiv preprint arXiv:2308.16512*, 2023. 1

[56] 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. 2

[57] Christopher Sweeney, Tobias Hollerer, and Matthew Turk. Theia: A fast and scalable structure-from-motion library. In *Proceedings of the ACM International Conference on Multimedia MM*, 2015. 2

[58] Chris Sweeney, Torsten Sattler, Tobias Hollerer, Matthew Turk, and Marc Pollefeys. Optimizing the viewing graph for structure-from-motion. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2015. 3

[59] Zachary Teed and Jia Deng. Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras. *Advances in neural information processing systems*, 34:16558–16569, 2021. 1

[60] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustment—a modern synthesis. In *Vision Algorithms: Theory and Practice: International Workshop on Vision Algorithms Corfu, Greece, September 21–22, 1999 Proceedings*, 2000. 1, 2, 4, 5, 14

[61] Haithem Turki, Deva Ramanan, and Mahadev Satyanarayanan. Mega-NeRF: Scalable construction of large-scale NeRFs for virtual fly-throughs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 6, 7, 16

[62] Jianyuan Wang, Nikita Karaev, Christian Rupprecht, and David Novotny. VGGSfM: Visual geometry grounded deep structure from motion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 2

[63] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5294–5306, 2025. 1, 3

[64] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. DUSt3R: Geometric 3D vision made easy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 1, 2

[65] Wenshan Wang, Delong Zhu, Xiangwei Wang, Yaoyu Hu, Yuheng Qiu, Chen Wang, Yafei Hu, Ashish Kapoor, and Sebastian Scherer. Tartanair: A dataset to push the limits of visual slam. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4909–4916. IEEE, 2020. 1

[66] Kyle Wilson and David Bindel. On the distribution of minima in intrinsic-metric rotation averaging. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[67] Kyle Wilson and Noah Snavely. Robust global translations with 1DSfM. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014. 2, 5

[68] Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 21469–21480, 2025. 1

[69] Linning Xu, Vasu Agrawal, William Laney, Tony Garcia, Aayush Bansal, Changil Kim, Samuel Rota Bulò, Lorenzo Porzi, Peter Kontschieder, Aljaž Božič, Dahua Lin, Michael Zollhöfer, and Christian Richardt. VR-NeRF: High-fidelity virtualized walkable spaces. In *SIGGRAPH Asia Conference Proceedings*, 2023. 6, 7

[70] Jianing Yang, Alexander Sax, Kevin J Liang, Mikael Henaff, Hao Tang, Ang Cao, Joyce Chai, Franziska Meier, and Matt Feiszli. Fast3r: Towards 3d reconstruction of 1000+ images in one forward pass. *arXiv preprint arXiv:2501.13928*, 2025. 3

[71] Yang Zheng, Adam W Harley, Bokui Shen, Gordon Wetzstein, and Leonidas J Guibas. Pointodyssey: A large-scale synthetic dataset for long-term point tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19855–19865, 2023. 1

[72] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 5

[73] Bingbing Zhuang, Loong-Fah Cheong, and Gim Hee Lee. Baseline desensitizing in translation averaging. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2, 5