VGGSfM: Visual Geometry Grounded Deep Structure From Motion
– *Supplementary Material* –

Jianyuan Wang[1,2] Nikita Karaev [1,2] Christian Rupprecht[1] David Novotny[2]

[1]Visual Geometry Group, University of Oxford [2]Meta AI

The supplementary material includes: (1) additional implementation details for network architectures and training hyperparameters; and (2) more ablation studies along with discussions.

1. Implementation Details

Training As discussed in the main manuscript, the training process involves multiple stages. We first train the tracker $\mathtt{T}$ on the synthetic Kubric dataset, then separately train the tracker $\mathtt{T}$, camera initializer $\mathfrak{T}_{\mathcal{P}}$, and triangulator $\mathfrak{T}_X$ on Co3D or MegaDepth, and finally jointly train the whole framework on Co3D or MegaDepth. We use the AdamW [13] optimizer with a cyclic learning rate scheduler [20] where each cycle spans 30 epochs. The learning rate is 0.0001 for the joint training phase and 0.0005 for all prior stages. We train the model on 32 NVIDIA A100 (80GB) GPUs until convergence. The batch size varies for each iteration because we randomly sample 3 to 30 frames for each scene (batch) as in [23]. The training on the synthetic Kubric dataset takes about one day. The separate training of the tracker $\mathtt{T}$, camera initializer $\mathfrak{T}_{\mathcal{P}}$, and triangulator $\mathfrak{T}_X$ takes two days, two days, and one day respectively. The final joint training takes one day. For training, we track 256 query points and run bundle adjustment for 5 optimization steps. We use gradient clipping to ensure stable training, which constrains the gradients' norm to a maximum value of 1. Additionally, we normalize the ground-truth cameras in the same way as in [23], and the point cloud correspondingly.

Moreover, we augment the samples using a combination of augmentation transformations. This includes color jittering (brightness, contrast, saturation, and hue) with a 65% probability, Gaussian Blur with a 50% probability, and a 15% chance of converting images to grayscale. Please note that different frames from a single scene will receive different augmentations. Images are resized to 512×512 with zero padding. Ground truth tracks that remain invisible in over 50% of the frames are excluded from the training for the tracker $\mathtt{T}$. For the MegaDepth dataset, similar to [12, 18], we construct the training batches by only sampling frames with an overlap score with the query frame exceeding 0.1. Here, overlap scores are derived from the pre-processing steps outlined in [7].

Inference Time On a single NVIDIA A100 80GB GPU, given 25 frames and 4096 query points, the inference of the tracker, camera initializer, and triangulator takes around 4.3, 0.9, and 0.2 seconds respectively. In comparison, the popular pairwise matching variant SuperPoint + SuperGlue usually takes around 20 seconds. In the bundle adjustment process, each optimization step requires approximately 0.7 seconds. For each run of the whole reconstruction function f_θ (as discussed in the main manuscript, f_θ is run multiple times until reaching sub-pixel BA reprojection error), bundle adjustment is executed for 30 steps, unless early convergence is achieved.

Tracker We use the 2D convolutional architecture from [8, 11] as the backbone of our tracker. Specifically, for the coarse tracker, this structure consists of an initial convolutional layer with a 7×7 kernel and stride of 2, followed by eight residual blocks with 3×3 kernels and instance normalization. Finally, the architecture concludes with a pair of convolutional layers, one using a 3×3 kernel and the other a 1×1 kernel. This backbone outputs a 128-dimensional feature map reducing the spatial resolution by a factor of 8. We use 5 levels of correlation pyramids where each level uses a correlation radius of 4. Therefore, the tokens (flattened cost volume) $V \in \mathbb{R}^{N_T \times N_I \times C}$ have a feature dimension of $C = 5 \times (2 \times 4 + 1)^2 = 405$. The tokens are subsequently processed by a transformer with eight self-attention layers with a hidden dimension of 512 and 8 heads. Finally, a multilayer perceptron (MLP) is applied to predict the point location y, visibility v, and inverse confidence σ. The architecture uses GELU activation functions.

The architecture of the fine tracker is similar to the coarse tracker but shallower. The backbone of the fine tracker consists of one 3×3 convolution layer, two residual blocks with 3×3 kernels and instance normalization, and one 1×1 convolution layer. The correlation pyramid of the fine tracker

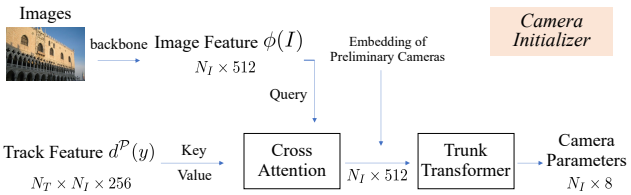

Figure 1. **Architecture of Camera Initializer.** Generally, we use the image features, track features, and the harmonic embedding of preliminary cameras to predict camera parameters. These parameters, represented in an $N_I \times 8$ matrix, comprise a quaternion (4 dimensions), a translation vector (3 dimensions), and a focal length (1 dimension).

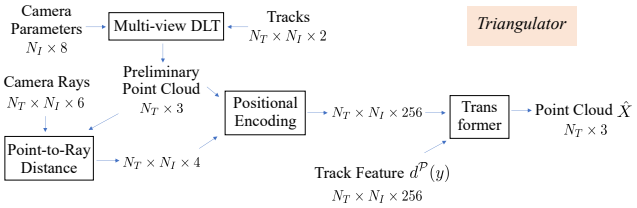

Figure 2. **Architecture of Triangulator.** We first estimate a preliminary point cloud using the camera parameters and tracks. Subsequently, we calculate the distance from this preliminary point cloud to all camera rays, as well as identify the nearest points on these rays. This information (along with the preliminary point cloud) is concatenated to the track features and fed into a transformer to predict the point cloud $\hat{X}$.

uses 3 levels and each level uses a radius of 3, which leads to tokens with a feature dimension of $3 \times (2 \times 3 + 1)^2 = 147$. The shallow transformer uses four self-attention layers, with a hidden dimension of 384 and 4 heads.

Following [8, 11], we train the tracker with 4 iterative updates and evaluate it with 6 iterative updates.

Camera Initializer The camera initializer (Fig. 1) takes frames I and track features $d^{\mathcal{P}}$ as input, and outputs initial cameras $\hat{\mathcal{P}}$. We extract features from the input images in a multi-scale manner as in [23]. However, we use ResNet [9] instead of DINO [2] as the camera initializer backbone, because we empirically found that DINO is harder to train jointly with other components. Each image is mapped to a 512-dimensional feature vector $\phi(I_i)$. Since the track features carry information about the image-to-image correspondence which provides grounding for camera-pose estimation, we fuse the stack of track features $d^{\mathcal{P}}(\mathbf{y})$, with shape $N_T \times N_I \times 256$, into the $N_I \times 512$ image features $\phi(I)$ with 4 cross-attention layers with 4 heads. This results in a $N_I \times 512$ global image descriptor.

Similar to the tracker, we adopt an iterative update mechanism inside the camera initializer. For each update, we obtain a set of 8-dimensionsal preliminary camera representations and map them to 128 dimensions with a positional harmonic embedding [14]. We then concatenate the global image descriptors and the embedding of the preliminary cameras, use an MLP to project the concatenated features to 512 dimensions, and feed the latter to a trunk transformer. The trunk transformer consists of 8 self-attention layers (transformer encoder) with 4 heads, whose hidden dimension is 512. The trunk transformer's output is further processed with another MLP layer, which predicts the camera parameters. This procedure is repeated four times. In the first run, the preliminary cameras are derived from each frame's relative camera pose to the query frame, which is computed from tracks using the 8-point algorithm. Following the approach of COLMAP [19], the focal lengths are initialized based on the longer side of the image size. In subsequent runs, the preliminary cameras (intrinsic and extrinsic) are the result of the previous prediction. In this process, the trunk transformer is run four times while the feature backbone is only run once.

It is noteworthy that the traditional 8-point algorithm is commonly used in conjunction with RANSAC to filter out noisy matches. In our approach, we employ a batched 8-point algorithm to approximate a similar effect to RANSAC while avoiding a time-consuming *for* loop. For each scene, we randomly select 20 sets, each comprising 50 point pairs. We then apply the 8-point algorithm to these sets in parallel, yielding 20 relative camera poses. Similar to RANSAC, we calculate the inlier count for each camera pose candidate using all available point pairs. A point pair is considered as an inlier if its Sampson epipolar error is less than 0.6 divided by the image width in pixels. Ultimately, the camera pose candidate with the highest number of inliers is selected.

Triangulator Given camera parameters and tracks, the triangulator (Fig. 2) $\mathfrak{T}_X$ initially estimates a preliminary point cloud $\bar{X}$ (of size $N_T \times 3$) using a closed-form multi-view Direct Linear Transform (DLT) for 3D triangulation. Furthermore, for each frame and the corresponding 2D point, a camera ray is computed. The distance from this camera ray to the associated 3D point in $\bar{X}$, along with the nearest point on the camera ray, are calculated. This results in the preliminary point cloud $\bar{X}$ with shape $N_T \times 3$, the ray distance with shape $N_T \times N_I \times 1$ and nearest points to camera rays of shape $N_T \times N_I \times 3$. These vectors are then concatenated (resulting in a tensor of shape $N_T \times N_I \times 7$) and embedded into a 256-dimensional space ($N_T \times N_I \times 256$) through positional encoding. The embedded vectors are further concatenated with the track feature $d^{\mathcal{P}}(\mathbf{y})$, leading to a shape of $N_T \times N_I \times 512$. Averaging over the N_I dimension yields a descriptor for the point cloud with dimensions $N_T \times 512$. This descriptor is input into a transformer comprising 4 self-attention layers, each with 4 heads and a hidden dimension of 384. The output of the transformer

	w/o Filtering	w/o BA	Ours
AUC@10°	2.31	18.34	73.92
RRE@5°	8.17	70.25	95.61
RTE@5°	5.42	39.42	81.03

Table 1. **Ablation Study for Bundle Adjustment.** We try the setting without using bundle adjustment, or using bundle adjustment but not filtering the correspondences.

is processed by a two-layer MLP (the hidden dimension is 256) to estimate $\hat{X}$.

Outlier Filtering It is important to filter out noisy correspondences in SfM, especially for BA optimization. For our framework, first, we drop 2D points with a visibility score $v < 0.6$ or variance $\sigma > 1$ (horizontally or vertically). Then, we use the preliminary cameras estimated by the 8-point algorithm and the initial cameras $\hat{\mathcal{P}}$ to remove correspondences with a Sampson epipolar error of more than 0.8 divided by the image width. Following Bundler [21] and COLMAP[19], we also require that at least one pair within each track has a triangulation angle of more than 3 degrees. Otherwise, the track (and the associated 3D point) is discarded. Moreover, for bundle adjustment, the 2D points with a reprojection error of more than 3 pixels are removed. Tracks with less than 3 points are discarded as well. It is worth mentioning that Homography verification [19] does not seem to be important for our framework, although it is common in incremental SfM.

2. Discussions and Ablation

Global SfM As discussed in the Related Work section of the main manuscript, there are two popular approaches for SfM: incremental and global. Global SfM approaches [1, 3–6, 10, 15–17, 22] usually predict the parameters for all the cameras at the same time and only perform bundle adjustment once. These methods often use rotation averaging and translation averaging to align pairwise relative camera poses into a consistent coordinate system. Our proposed method bears similarities to global SfM. However, it diverges in several key aspects: (1) unlike global SfM, which relies on pairwise matching (akin to incremental SfM), our method directly predicts tracks; (2) instead of rotation averaging and translation averaging in global SfM, we use a learnable network to predict camera parameters; (3) we iteratively apply the reconstruction function multiple times during testing, with bundle adjustment at each iteration. Besides these differences, our method is complementary to global SfM.

Bundle Adjustment Bundle adjustment is a key component for accurate SfM. As shown in Tab. 1, without bundle adjustment, we observe a clear performance drop, with AUC@10 from 73.92 to 18.34. BA is also known to be strongly susceptible to noisy inputs. Indeed, using bundle adjustment without track filtering (described in previous paragraphs) destroys the estimate and, as such, reduces the AUC@10 nearly to zero. Notably, even without bundle adjustment, our framework's estimation remains relatively robust; for instance, the rotation errors for over 70% of image pairs remain within 5 degrees (RRE@5° > 70%). However, executing bundle adjustment without track filtering results in incorrect optimization of camera parameters, whose RRE@5° is also just around 8%. At the same time, please note that all the methods in Table 2 of the main manuscript use bundle adjustment or its approximation. For example, PoseDiffusion [23] uses geometry-guided sampling and DeepSfM [24] adopts a special form of bundle adjustment. Without geometry-guided sampling, the AUC@10 of PoseDiffusion is around 11%.

Track Error Distribution We present a histogram in Fig. 3 to depict the distribution of tracking errors for the scene *British Museum 10 bag 000* within the IMC dataset, consisting of 10 images. As indicated, the distribution's peak, represented by the orange dashed line, approximately aligns with 0.4 pixels, while the median, depicted by the blue dash-dotted line, is around 0.6 pixels. Notably, most of the tracking predictions maintain an error margin of less than 3 pixels, highlighting the accuracy of our method. Some predictions even approach a near-zero error margin. Invisible points (*e.g.*, occluded or outside the view) are not included in this histogram.

Video Tracking To verify the effect of our proposed tracker, we also try to use the video tracking method PiPs inside our framework. The results, presented in Table 2, reveal a noticeable decline in performance when using PiPs

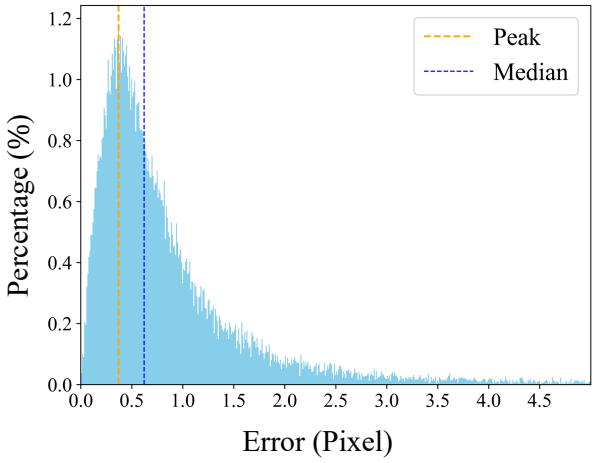

Figure 3. **Histogram of Tracking Errors** of the scene *British Museum 10 bag 000* on the IMC dataset. The horizontal axis denotes error in pixels, while the vertical axis shows the percentage (%) for each bin.

	AUC@10°	RRE@5°	RTE@5°
PiPs [8]	43.27	82.15	51.39
Our Trakcer	73.92	95.61	81.03

Table 2. **Ablation Study for Tracking.** We try the video tracking method PiPs [8] in our framework, which shows a clear performance drop .

as opposed to our tracker. This contrast underlines the effectiveness of our proposed tracking solution.

Failure Cases Our method seldom fails entirely for a scene, but it may struggle with specific images where classical SfM also fails. The latter includes symmetric objects (e.g., Co3D apples) or "doppelgangers" (e.g., IMC Piazza San Marco). We can follow classical SfM to detect failure cases, e.g., BA-reprojection or Sampson error. A low tracking confidence can also indicate failure.

Scalability Our model can currently reconstruct 80-120 frames on a single A100 GPU. The main bottleneck is the GPU memory of Transformer blocks. While Transformer can handle arbitrary numbers of input frames, its memory scales *quadratically* with input size. In practice, using $2\times$ frames will require $\sim 3.3\times$ GPU memory.

There are several directions to enhance scalability. For example, following recent advancements in NLP, we can leverage Transformer with linear memory complexity (such as Mamba) to reduce GPU consumption. Additionally, using mixed precision or reconstructing with sliding window can be considered. We remark that scalability is a crucial aspect of SfM, but is not the primary focus of our paper. Given the classical SfM pipeline also originated from dozens of pictures, we are humbly satisfied with processing $\sim$100 frames for now. We plan to extend the input size in our future work.

References

[1] Mica Arie-Nachimson, Shahar Z Kovalsky, Ira Kemelmacher-Shlizerman, Amit Singer, and Ronen Basri. Global motion estimation from point matches. In *2012 Second international conference on 3D imaging, modeling, processing, visualization & transmission*, pages 81–88. IEEE, 2012. 3

[2] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021. 2

[3] David J Crandall, Andrew Owens, Noah Snavely, and Daniel P Huttenlocher. Sfm with mrfs: Discrete-continuous optimization for large-scale structure from motion. *IEEE transactions on pattern analysis and machine intelligence*, 35(12):2841–2853, 2012. 3

[4] Hainan Cui, Xiang Gao, Shuhan Shen, and Zhanyi Hu. Hsfm: Hybrid structure-from-motion. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1212–1221, 2017.

[5] Zhaopeng Cui and Ping Tan. Global structure-from-motion by similarity averaging. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 864–872, 2015.

[6] Zhaopeng Cui, Nianjuan Jiang, Chengzhou Tang, and Ping Tan. Linear global translation estimation with feature tracks. *arXiv preprint arXiv:1503.01832*, 2015. 3

[7] Mihai Dusmanu, Ignacio Rocco, Tomas Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii, and Torsten Sattler. D2-net: A trainable cnn for joint description and detection of local features. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, pages 8092–8101, 2019. 1

[8] Adam W Harley, Zhaoyuan Fang, and Katerina Fragkiadaki. Particle video revisited: Tracking through occlusions using point trajectories. In *ECCV*, 2022. 1, 2, 4

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2

[10] Nianjuan Jiang, Zhaopeng Cui, and Ping Tan. A global linear method for camera pose registration. In *Proceedings of the IEEE international conference on computer vision*, pages 481–488, 2013. 3

[11] Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. CoTracker: It is better to track together. 2023. 1, 2

[12] Philipp Lindenberger, Paul-Edouard Sarlin, and Marc Pollefeys. Lightglue: Local feature matching at light speed. *arXiv preprint arXiv:2306.13643*, 2023. 1

[13] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 1

[14] 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. *Proc. ECCV*, 2020. 2

[15] Pierre Moulon, Pascal Monasse, and Renaud Marlet. Global fusion of relative motions for robust, accurate and scalable structure from motion. In *Proceedings of the IEEE international conference on computer vision*, pages 3248–3255, 2013. 3

[16] Onur Ozyesil and Amit Singer. Robust camera location estimation by convex programming. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2674–2683, 2015.

[17] Rother. Linear multiview reconstruction of points, lines, planes and cameras using a reference plane. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 1210–1217. IEEE, 2003. 3

[18] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4938–4947, 2020. 1

[19] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2, 3

[20] Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial intelligence and machine learning for multi-domain operations applications*, pages 369–386. SPIE, 2019. 1

[21] Noah Snavely, Steven M Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. In *ACM siggraph 2006 papers*, pages 835–846. 2006. 3

[22] Chris Sweeney, Torsten Sattler, Tobias Hollerer, Matthew Turk, and Marc Pollefeys. Optimizing the viewing graph for structure-from-motion. In *Proceedings of the IEEE international conference on computer vision*, pages 801–809, 2015. 3

[23] Jianyuan Wang, Christian Rupprecht, and David Novotny. Posediffusion: Solving pose estimation via diffusion-aided bundle adjustment. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9773–9783, 2023. 1, 2, 3

[24] Xingkui Wei, Yinda Zhang, Zhuwen Li, Yanwei Fu, and Xiangyang Xue. Deepsfm: Structure from motion via deep bundle adjustment. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pages 230–247. Springer, 2020. 3