arXiv:2507.16443v1 [cs.CV] 22 Jul 2025

VGGT-Long: Chunk it, Loop it, Align it – Pushing VGGT's Limits on Kilometer-scale Long RGB Sequences

Kai Deng[1], Zexin Ti[2], Jiawei Xu[1], Jian Yang[1], Jin Xie[2†]
[1]College of Computer Science, Nankai University, Tianjin, China
[2]School of Intelligence Science and Technology, Nanjing University, Suzhou, China
dengkai@mail.nankai.edu.cn, zexinti@smail.nju.edu.cn, jiaweixu@mail.nankai.edu.cn,
csjyang@nankai.edu.cn, csjxie@nju.edu.cn

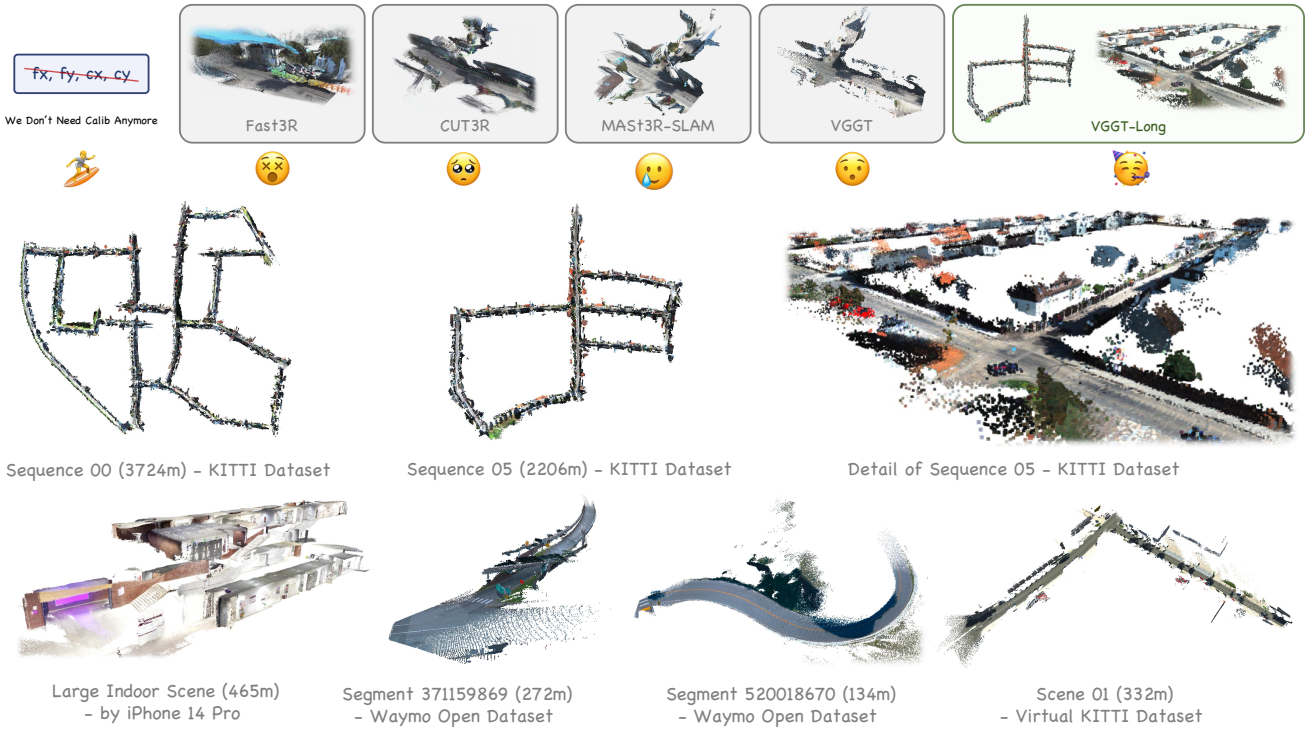


Figure 1. For large-scale outdoor scenarios, previous work suffers from: 1) severe drift (CUT3R and Fast3R); 2) unable to complete the entire long sequence (MASt3R-SLAM and VGGT). Our method **VGGT-Long** is able to complete the reconstruction of the kilometer-scale scene while maintaining the accuracy of the scene.

Abstract

*Foundation models for 3D vision have recently demonstrated remarkable capabilities in 3D perception. However, extending these models to large-scale RGB stream 3D reconstruction remains challenging due to memory limitations. In this work, we propose **VGGT-Long**, a simple yet effective system that pushes the limits of monocular 3D reconstruction to **kilometer-scale, unbounded outdoor environments**. Our approach addresses the scalability bottlenecks of existing models through a chunk-based processing strategy combined with overlapping alignment and lightweight loop closure optimization. Without requiring camera calibration, depth supervision or model retraining, VGGT-Long achieves trajectory and reconstruction performance comparable to traditional methods. We evaluate our method on KITTI, Waymo, and Virtual KITTI datasets. VGGT-Long not only runs successfully on long RGB se-*

[1†] Corresponding author.

quences where foundation models typically fail, but also produces accurate and consistent geometry across various conditions. Our results highlight the potential of leveraging foundation models for scalable monocular 3D scene in real-world settings, especially for autonomous driving scenarios. Code is available at https://github.com/DengKaiCQ/VGGT-Long.

1. Introduction

Perceiving 3D environments from monocular RGB streams is crucial for autonomous driving, yet existing methods struggle with kilometer-scale and uncalibrated sequences. Unlike the small-scale indoor 3D vision tasks, driving scenarios involve long trajectories with sparse frame correspondence, dynamic objects and challenging outdoor conditions. While some approaches [8, 45, 46] handle large-scale monocular scenes, they often depend on sophisticated multi-module pipelines or assume known camera intrinsics. Others leverage additional sensors (LiDAR [48], IMU [47] or stereo [5]), sidestepping the core challenge: scalable, calibration-free reconstruction from monocular RGB alone and it is a critical for autonomous systems.

A recent paradigm shift in 3D vision has witnessed the rise of end-to-end foundation models, largely based on the Transformer architecture [39]. A mainly of works from DUSt3R [43] and MASt3R [17] to CUT3R [42], Fast3R [44], and most recently VGGT [41], aim to replace complex, multi-component SfM and SLAM pipelines with a single and unified deep learning model. These models are trained on massive datasets to integrate camera pose estimation, intrinsic parameter regression, and 3D scene representation (typically as a point map) into one cohesive framework. A key goal is to enable backpropagation of errors through the entire system, creating a powerful and versatile foundation model for 3D reconstruction that operates on raw and uncalibrated RGB inputs. However, foundation models like CUT3R and Fast3R still struggle with severe drift in outdoor environments, even on short sequences of a few dozen frames, which limits their practical applicability. In contrast, VGGT delivers remarkably stable and accurate local reconstructions, establishing it as the state-of-the-art in terms of reconstruction quality. Its primary limitation is not performance, but its immense computational and memory footprint.

The computational and memory demands of Transformer based foundation models severely limit their scalability. Standard self-attention [39] scales quadratically with input size, and while techniques like Flash-Attention [6, 7] reduce compute complexity to linear. However, GPU memory requirement still remains prohibitive. For example, VGGT can just process 60 to 80 images on a 24 GiB RTX 4090 GPU; scaling to a KITTI Seq 00 trajectory (about 4,600 frames) would require 1,380 to 1,840 GiB, far exceeding current hardware. This bottleneck confines such models to small-scale scenes, as both memory and drift accumulation become intractable over long sequences.

Our work is inspired by recent efforts to integrate these foundation models into large-scale systems. A notable example is MASt3R-SLAM [24], which builds a sophisticatedly designed SLAM system on top of the MASt3R model. To achieve global consistency, it employs a complex backend with pose graph optimization, bundle adjustment and other heavy machinery. While powerful, such systems often entail significant engineering complexity, making them difficult to adapt to down-streaming tasks.

This raises a fundamental question: must large-scale reconstruction always equate to system-level complexity? Our philosophy diverges significantly from this trend. We advocate for a minimalist approach that unlocks the inherent potential of the foundational model itself. We posit that VGGT is already a remarkably powerful engine for large-scale 3D perception, and the primary challenge is not a lack of capability, but a lack of scalability. Instead of building another intricate system around it, we ask: can we solve the problem with the minimal overhead?

To this end, we propose VGGT-Long, a framework that extends VGGT to long sequences through a simple yet effective framework: processing the sequence in overlapping chunks, robustly aligning adjacent chunks, and correcting for drift using a high-quality loop closure module. This "chunk-and-align" paradigm avoids the need for a complex graph-based optimization backend (such as bundle adjustment [36]). It is a testament to the power of the underlying VGGT model, demonstrating that with the right strategy, its exceptional local reconstruction capabilities can be seamlessly stitched together to form a globally consistent, kilometer-scale map. Our work champions the idea that, **a sufficiently powerful base model may not necessarily require a complex backend system to assist**.

In summary, our contributions are as follows:

1. We present the first system that successfully extends monocular 3D reconstruction models to kilometer-scale, unbounded outdoor scenes, without requiring camera calibration and depth supervision.
2. We introduce a simple yet effective chunk-and-align pipeline that resolves the memory limitations of foundation models like VGGT on long video sequences, while achieving accuracy comparable to traditional methods with calibrated cameras.
3. We address the accumulated Sim(3) drift problem inherent in processing long sequences with local models, demonstrating that VGGT can serve as a robust front-end for a large-scale reconstruction system without requiring a complex backend.

2. Related Work

Structure-from-Motion (SfM). SfM methods estimate camera poses and sparse 3D structure from multi-view images. Classical SfM pipelines [1, 10, 27] typically follow an incremental strategy: detecting keypoints [9, 37], matching features [4, 19] and refining poses through bundle adjustment [36]. While robust, these pipelines rely heavily on features extraction and are limited in textureless or ambiguous scenes. Recent deep learning methods aim to enhance or replace traditional modules. Hybrid frameworks such as PixSfM [18] and DFSfM [14] combine deep features with classical optimization to refine both tracks and structure. Fully differentiable SfM pipelines [28, 32, 33, 38] further explore end-to-end learning of camera poses and depth but often suffer from scalability issues or poor generalization. VGGSfM [40] demonstrates that learned systems can surpass classical SfM on real-world datasets by integrating dense features and multi-view consistency into a unified framework.

SLAM and Visual Odometry. To overcome the scalability issue of foundation models, researchers have explored to integrate them into larger SLAM systems. Traditional SLAM systems [3, 22, 23] rely on handcrafted features and optimization, while learning-based approaches [2, 16, 34] integrate differentiable components into deep networks. However, these methods either scale poorly to long sequences or require pre-calibrated camera. MASt3R-SLAM [24] builds a sophisticated real-time SLAM framework around the MASt3R [17] without the calibration, employing complex backend machinery such as pose graph optimization and bundle adjustment to ensure global consistency.

Transformer based 3D Vision Method. A recent trend of 3D vision is the development of end-to-end foundation models, predominantly based on the Transformer architecture. These methods target dense geometry reconstruction from overlapping images, assuming unknown poses and unknown camera calibration. A pioneering line of work, including DUSt3R [43] and its successor MASt3R [17], demonstrates the feasibility of jointly estimating camera parameters and dense 3D geometry from uncalibrated image pairs. Subsequent models like CUT3R [42] and Fast3R [44] further refined this paradigm. Most recently, VGGT [41] obtained a new state-of-the-art in reconstruction quality, producing remarkably stable and accurate local 3D maps from raw RGB inputs. However, a common limitation of these models is their significant computational and memory cost, which restricts their application to short image sequences.

Our work, VGGT-Long, distinguishes itself by adopting a minimalist philosophy. Instead of building a complex system, we focus on unlocking the full potential of the powerful VGGT model itself, using a simple yet effective chunk-and-align framework to extend its capabilities to long-sequence, large-scale scenarios with the minimal overhead.

3. Method

Our proposed method, VGGT-Long, processes long monocular RGB sequences by decomposing the problem into three stages: chunking, chunk-wise alignment, loop closure and loop correction. Our method maintains the local accuracy of VGGT while ensuring global accuracy when handling the outdoor long sequences.

3.1. Sequence Chunking and Local Aligning with Confidence

Given a long image sequence $\mathcal{I} = \{\mathbf{I}_1, \ldots, \mathbf{I}_N\}$, we partition it into K overlapping chunks. Let L be the chunk size and O be the overlap size. The k-th chunk, for $k = 1, \ldots, K$, is defined as the subsequence of frames indexed from $(k-1)(L-O)$ to $(k-1)(L-O)+L$. Each chunk $\mathcal{C}_k$ is processed independently by the VGGT model [41], which outputs a set of locally consistent camera poses and a 3D point map $\mathbf{P}_k \in \mathbb{R}^{H\times W\times 3}$ with the corresponding confidence values $\mathbf{c}_k \in \mathbb{R}^{H\times W}$.

We fully leverage the confidence VGGT outputs to perform chunk alignment. In VGGT, the output confidence reflects the model's certainty about the point cloud. More specifically, in outdoor environments, objects such as the sky, oncoming vehicles, fast-moving pedestrians or raindrops during rainy conditions can affect the SIM(3) alignment calculation to some extent. VGGT's final output assigns lower confidence to these points, while assigning higher confidence to static scenes (such as buildings or stationary vehicles). Our alignment strategy is designed to achieve robust alignment based on this principle.

For each pair of adjacent chunks, $\mathcal{C}_k$ and $\mathcal{C}_{k+1}$, we identify a set of 3D point correspondences $\{(\mathbf{p}_k^i, \mathbf{p}_{k+1}^i)\}$ and $\{(\mathbf{c}_k^i, \mathbf{c}_{k+1}^i)\}$ within their overlapping region. To robustly estimate the relative Sim(3) transformation $\mathbf{S}_{k,k+1} \in \mathrm{Sim}(3)$ that aligns $\mathcal{C}_{k+1}$ to $\mathcal{C}_k$, we employ an Iteratively Reweighted Least Squares (IRLS) optimization. The objective is to minimize the following robust cost function

$$\mathbf{S}_{k,k+1}^* = \arg\min_{\mathbf{S}\in \mathrm{Sim}(3)} \sum_i \rho\left(\|\mathbf{p}_k^i - \mathbf{S}\mathbf{p}_{k+1}^i\|_2\right), \quad (1)$$

where $\rho(\cdot)$ is the Huber loss function, which down-weights the influence of outliers. The IRLS procedure solves this non-linear problem by iteratively minimizing a weighted sum of squared errors

$$\mathbf{S}^{(t+1)} = \arg\min_{\mathbf{S}\in \mathrm{Sim}(3)} \sum_i \mathbf{w}_i^{(t)} \|\mathbf{p}_k^i - \mathbf{S}\mathbf{p}_{k+1}^i\|_2^2. \quad (2)$$

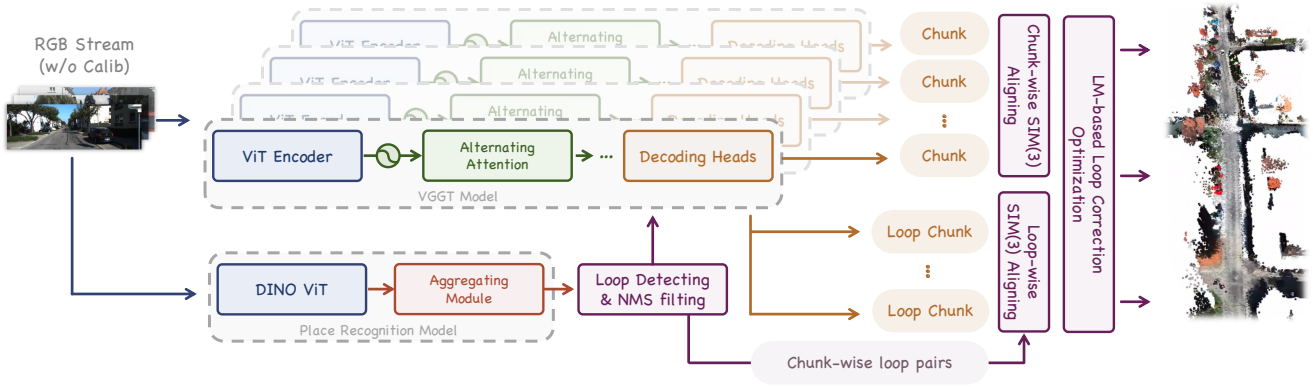
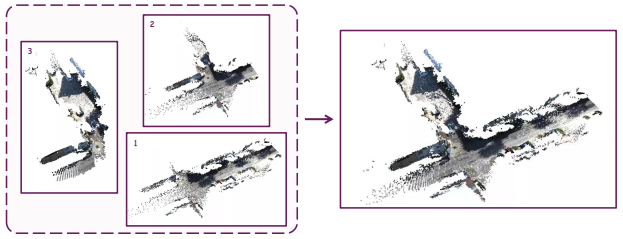
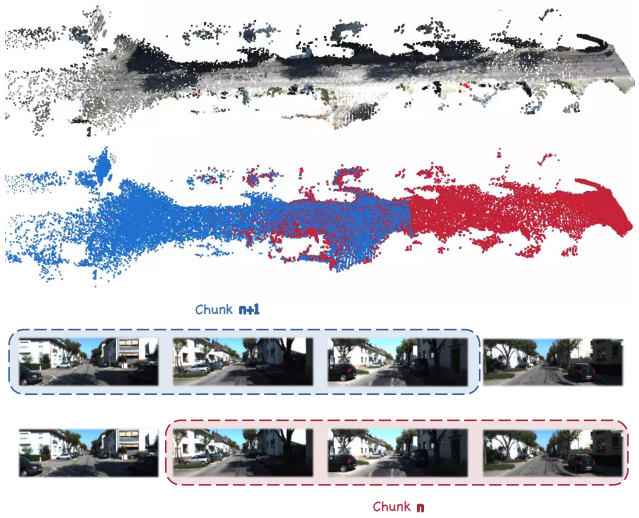


Figure 2. Overview of VGGT-Long. VGGT-Long processes long sequences by dividing them into different chunks, thereby handling the input RGB stream in a sliding window manner. We fully utilize VGGT's pointmap and confidence to perform lightweight loop closure and alignment on the output chunks, thus extending VGGT to long-sequence datasets for autonomous driving.



(a) The visualization of different chunks.



(b) The visualization of overlapping frames.

Figure 3. (a) VGGT-long divides a kilometer-scale sequence into different chunks for processing. (b) The alignments are derived from the consistency of overlapping frames in 3D space.

At each iteration t, the weight $\mathbf{w}_i^{(t)}$ for the i-th correspondence is a product of the model's confidence $\mathbf{c}_i$ and a robustness term derived from the Huber loss

$$\mathbf{w}_i^{(t)} = \mathbf{c}_i \cdot \frac{\rho'(r_i^{(t)})}{r_i^{(t)}}, \tag{3}$$

where $r_i^{(t)} = \|\mathbf{p}_k^i - \mathbf{S}^{(t)}\mathbf{p}_{k+1}^i\|_2$ is the residual from the previous iteration. Each weighted least-squares problem is solved efficiently in closed form using a weighted version of the Umeyama algorithm. Through this approach, we exclude low-confidence points (e.g., rapidly moving objects and sky regions) as outliers (see Fig. 4). For medium-confidence objects (e.g., slowly moving vehicles), we assign reduced alignment weights, while concentrating higher weights on high-confidence structures (e.g., buildings). In our implementation, we directly discard points with confidence values below $0.1\times$ the median confidence of the entire chunk. The remaining low-confidence points that survive this filtering contribute minimally to the alignment process due to their attenuated weights.

3.2. Loop Detection and Loop-wise SIM(3) Aligning

To correct the accumulated drift inherent in sequential estimation, we perform loop closure detection across the entire sequence. This process involves identifying non-adjacent chunks as the same scene and robustly estimating the Sim(3) transformation between them.

First, we use a pre-trained Visual Place Recognition (VPR) model [15], which leverages a DINOv2 backbone [25], to extract a compact and descriptive global feature vector $\mathbf{d}_i$ for each image $\mathbf{I}_i$ in the sequence. These descriptors capture the high-level semantic and geometric content of the images.

With global descriptors for all images, we identify potential loop closure candidates. For each descriptor, we perform an efficient nearest neighbor search to find other images with high cosine similarity. A pair of images $(\mathbf{I}_i, \mathbf{I}_j)$ is considered as a potential loop closure if their similarity

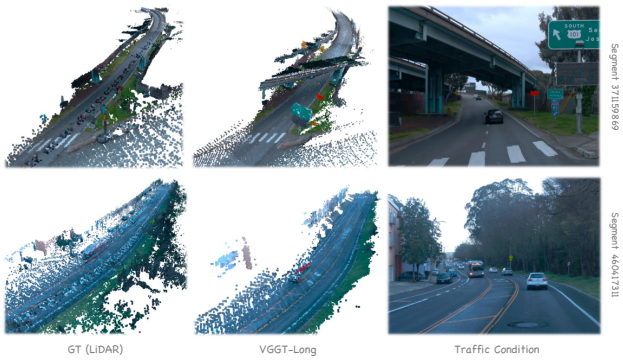


Figure 4. Confidence-aware alignment suppress the influence of high-speed dynamic objects (such as vehicles) on alignment and reconstruction. It could be observed that higher-density vehicles cannot be effectively filtered out by the LiDAR, but VGGT-Long has the ability to handle this situation.

score exceeds a threshold τ_s and their frame indices are sufficiently separated, i.e., $|i - j| > \Delta t_{min}$. To ensure that the detected loops are distinct and to avoid redundant matches in temporally close frames, we apply Non-Maximum Suppression (NMS). This filtering step selects the strongest match within a local time window, yielding a set of high-confidence image-level loop pairs.

For each validated loop pair $(\mathbf{I}_i, \mathbf{I}_j)$, we adopt a specialized strategy to generate a high-quality local reconstruction of the looped scene. We form a new, temporary image batch by concatenating sub-sequences of frames centered around the indices i and j. This batch, containing temporally disjoint views of the same location, is then processed by the VGGT model. Unlike the sequential, sliding-window processing described in Sec. 3.1, this approach provides VGGT with a more diverse, time-dispersed perspective, enabling a more robust reconstruction of the scene's geometry by leveraging a wider baseline.

The resulting 3D point map, which we can call it the "loop-centric" chunk, is then aligned with the original point maps of the corresponding chunks $\mathcal{C}_i$ and $\mathcal{C}_j$ (which were generated from temporally continuous frames). To compute the final loop-closing transformation $\mathbf{S}_{ij}$, we chain the alignments through this new chunk. Specifically, we compute the transformations from the original chunks to the loop-centric chunk and then compose them

$$\mathbf{S}_{ji} = \mathbf{S}_{j,\text{loop}} \circ \mathbf{S}_{i,\text{loop}}^{-1}, \tag{4}$$

where $\mathbf{S}_{i,\text{loop}}$ and $\mathbf{S}_{j,\text{loop}}$ are the transformations that align the loop-centric chunk to the coordinate frames of chunks $\mathcal{C}_i$ and $\mathcal{C}_j$ respectively (described in Sec. 3.1). The composition $\mathbf{S}_{j,\text{loop}} \circ \mathbf{S}_{i,\text{loop}}^{-1}$ effectively chains the transformations to compute the direct alignment from chunk $\mathcal{C}_i$ to $\mathcal{C}_j$ (i.e., $i \rightarrow \text{loop} \rightarrow j$). This provides a robust geometric

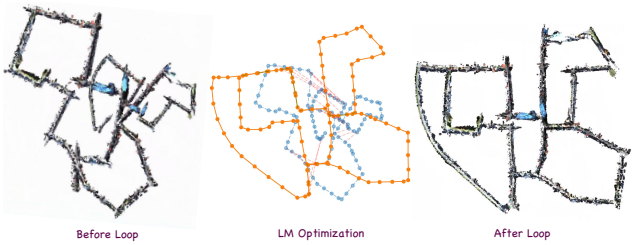


Figure 5. Without loop constraints, errors will be accumulated continuously at the kilometer scale. The use of Global LM Optimization can alleviate this accumulated error.

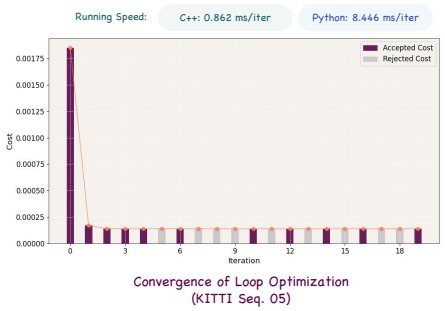


Figure 6. The loop optimization can be converged in just 3 iterations and can achieve millisecond-level performance.

constraint for the global optimization by bridging the two distant chunks through a shared, high-quality local reconstruction.

3.3. Global SIM(3) LM-based Optimization

To achieve global consistency, We follow [8, 20, 29], performing a global optimization over the Sim(3) transformations of all chunks. Instead of constructing a complex factor graph, we directly minimizes a non-linear least-squares objective function composed of two types of geometric constraints: sequential constraints from adjacent chunks (Sec. 3.1) and loop closure constraints from non-adjacent chunks (Sec. 3.2). The goal is to jointly optimize the transformations $\{\mathbf{S}_k\}_{k=1}^{K}$ for all chunks to be maximally consistent with these relative measurements. The optimization problem is formulated as

$$\begin{aligned}\{\mathbf{S}_k^*\} = \arg\min_{\{\mathbf{S}_k\}} &\sum_{k=1}^{K-1} \| \log_{\text{Sim}(3)}(\mathbf{S}_{k,k+1}^{-1} \mathbf{S}_k^{-1} \mathbf{S}_{k+1}) \|_2^2 \\ &+ \sum_{(i,j)\in\mathcal{L}} \| \log_{\text{Sim}(3)}(\mathbf{S}_{ij}^{-1} \mathbf{S}_i^{-1} \mathbf{S}_j) \|_2^2,\end{aligned} \tag{5}$$

where $\mathbf{S}_{k,k+1}$ is the relative transformation between adjacent chunks, $\mathbf{S}_{ij}$ is the loop closure transformation between chunks i and j, and $\mathcal{L}$ is the set of all loop closures. The $\log_{\text{Sim}(3)}(\cdot)$ map converts a Sim(3) transformation into its

7-dimensional tangent space representation in the Lie algebra $\mathfrak{sim}(3)$, allowing for unconstrained optimization. We solve this non-linear least-squares problem efficiently using the Levenberg-Marquardt (LM) algorithm. The global optimization operates chunk-wise, maintaining a small set of SIM(3) variables (typically dozens even for KITTI, see Fig. 5). Convergence is achieved within few iterations, with processing times in milliseconds (see Fig. 6).

4. Experiments

We evaluate VGGT-Long on three datasets — KITTI Dataset Odometry Track [13], Waymo Open Dataset (v1.4.1) [31], and Virtual KITTI Dataset (v1.3.1) [11] to assess its performance in large-scale monocular 3D reconstruction. Our experiments are conducted on the computer with Ubuntu 22.04, and equipped with 12 Intel Xeon Gold 6128 3.40 GHz CPUs, 67GiB of RAM, and an NVIDIA RTX 4090 GPU with 24GiB of VRAM.

4.1. Metrics

We report four different metrics. For tracking performance, following the recent literature, we use the Absolute Trajectory Error (ATE) metric [30] to evaluate the model's long-term consistency over extended sequences. For reconstruction metrics, we adopt the settings from VGGT [41] and employ: 1) Accuracy: Euclidean distance from each predicted point to its nearest Ground Truth (GT) point; 2) Completeness: Euclidean distance from each GT point to its nearest predicted point; 3) Chamfer Distance: The average of the above two metrics. Due to the scale ambiguity inherent in monocular 3D methods, we first perform coarse alignment between the predicted poses and GT poses before calculating the reconstruction metrics. Subsequently, we refine the alignment using the Point-to-Point Iterative Closest Point (ICP) method [26] and then compute these metrics.

4.2. Experiments Settings

For the calculation of reconstruction performance metrics, to avoid interference from outlier point clouds, we uniformly retain points with confidence values greater than $0.75\times$ the average confidence for Transformer-based 3D methods. As for models like DROID-SLAM, which do not output confidence values, we retain points with depth values larger than $0.75\times$ the average inverse depth as the filtered point cloud.

Following the VGGT's configuration, we downsample input images to 518-pixel width while the hight maintaining aspect ratio during inference. Due to the more "elongated" image aspect ratios in KITTI and Virtual KITTI datasets, we set the chunk size to 75 frames, while adopting 60 frames for Waymo (nearly saturating the whole capacity of RTX 4090 24GiB VRAM). Our inference pipeline first processes all images through VPR, then releases the VPR model's memory occupancy to allocate sufficient GPU memory resources for VGGT.

4.3. CPU Memory Management Strategy

To handle large-scale scenarios where CPU memory cannot accommodate all VGGT outputs, we implement a memory-efficient strategy: after processing each chunk, we store the results on disk. During subsequent alignment phases, we selectively load only relevant chunk pairs into CPU memory for SIM(3) computation, and immediately freeing the memory after calculation. This design offloads CPU memory pressure to disk storage, and will effectively prevents the operating system crashes, freezes, or other catastrophic failures caused by excessive CPU memory consumption. Upon completing computations, the model purges all intermediate results to minimize the storage overhead. The final outputs (colored point clouds and camera poses) employ stream writing techniques to bypass CPU memory constraints during large-scale reconstruction, effectively eliminating RAM pressure from handling massive point cloud datasets.

4.4. KITTI, Waymo & Virtual KITTI

We begin with the KITTI odometry dataset, a classic benchmark for SLAM evaluation. Due to the long sequence lengths and outdoor settings, KITTI presents significant challenges for monocular reconstruction. Table 1 reports the ATE across 11 sequences.

VGGT-Long outperforms the learning-based methods such as DROID-SLAM and DPVO. Unlike ORB-SLAM2 or LDSO, our method does not rely on calibrated camera intrinsics and still maintains competitive accuracy. Notably, VGGT-Long runs successfully on all sequences, while foundation models like Fast3R, CUT3R, and VGGT fail due to memory overflow. This verifies that our chunk-and-align framework effectively extends VGGT to long sequences without sacrificing scalability.

To validate generalization, we test VGGT-Long on the Waymo Open Dataset, which features urban driving with high variability in scene appearance and traffic conditions. VGGT-Long achieves an average ATE of 1.996m across ten 200-frame segments. Compared to methods like CUT3R and MASt3R-SLAM, VGGT-Long yields significantly lower errors, as shown in Table 2. On segments with strong viewpoint diversity, VGGT-Long consistently produces more accurate and complete 3D reconstructions.

We further assess robustness under synthetic domain shifts using the Virtual KITTI dataset, which offers multiple weather and lighting conditions. As Table 4 shows, VGGT-Long maintains stable ATE across all tested conditions, including fog, rain, and sunset. Unlike DROID-SLAM, which occasionally fails to track, and CUT3R, which exhibits large drifts, our method remains robust without requiring retrain-

	Methods	LC	Calibration	Recon.	Avg.	Avg.*	00	01	02	03	04	05	06	07	08	09	10
	seq. frames	-	-	-	*2109*	*2210*	*4542*	*1101*	*4661*	*801*	*271*	*2761*	*1101*	*1101*	*4071*	*1591*	*1201*
	seq. length (m)	-	-	-	*2012.243*	*1968.147*	*3724.19*	*2453.20*	*5067.23*	*560.89*	*393.65*	*2205.58*	*1232.88*	*649.70*	*3222.80*	*1705.05*	*919.52*
	seq. speed (m / frame)	-	-	-	*0.95*	*0.89*	*0.82*	*2.23*	*1.09*	*0.70*	*1.45*	*0.80*	*1.12*	*0.59*	*0.79*	*1.07*	*0.77*
	contains loop	-	-	-	-	-	✓	✗	✓	✗	✗	✓	✓	✓	✗	✓	✗
Classic	ORB-SLAM2 (w/o LC) [22]	✗	*Required*	*Sparse*	69.727	26.480	40.65	502.20	47.82	[0.94]	1.30	29.95	40.82	16.04	(43.09)	(38.77)	[5.42]
Classic	ORB-SLAM2 (w/ LC) [22]	✓	*Required*	*Sparse*	54.816	[9.464]	[6.03]	508.34	[14.76]	(1.02)	1.57	[4.04]	(11.16)	(2.19)	[38.85]	[8.39]	(6.63)
Classic	LDSO [12]	✓	*Required*	*Sparse*	[22.425]	23.500	9.32	(11.68)	(31.98)	2.85	1.22	(5.10)	13.55	2.96	129.02	21.64	17.36
Learning Based	DROID-VO [34]	✗	*Required*	*Dense*	54.188	51.187	98.43	84.20	108.80	2.58	0.93	59.27	64.40	24.20	64.55	71.80	16.91
Learning Based	DPVO [35]	✗	*Required*	*Sparse*	53.609	57.701	113.21	12.69	123.40	2.09	[0.68]	58.96	54.78	19.26	115.90	75.10	13.63
Learning Based	DROID-SLAM [34]	-	*Required*	*Dense*	100.278	75.846	92.10	344.60	107.61	2.38	1.00	118.50	62.47	21.78	161.60	72.32	118.70
Learning Based	DPV-SLAM [20]	✓	*Required*	*Sparse*	53.034	57.187	112.80	[11.50]	123.53	2.50	0.81	57.80	54.86	18.77	110.49	76.66	13.65
Learning Based	DPV-SLAM++ [20]	✓	*Required*	*Sparse*	(25.749)	27.138	(8.30)	11.86	39.64	2.50	(0.78)	5.74	11.60	[1.52]	110.90	76.70	13.70
Learning Based	MASt3R-SLAM [24]	✓	*No Need*	*Dense*	/	/	*TL*	*TL*	*TL*	*TL*	*TL*	*TL*	*TL*	*TL*	*TL*	*TL*	*TL*
Learning Based	CUT3R [42]	✗	*No Need*	*Dense*	/	/	*OOM*	*OOM*	*OOM*	148.07	22.31	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*
Learning Based	Fast3R [44]	✗	*No Need*	*Dense*	/	/	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*
Learning Based	VGGT [41]	✗	*No Need*	*Dense*	/	/	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*
	VGGT-Long (Ours)	✓	*No Need*	*Dense*	27.637	(18.284)	8.67	121.17	32.08	6.12	4.23	8.31	[5.34]	4.63	53.10	41.99	18.37

Table 1. Camera Tracking Results (ATE RMSE [m] ↓) on the KITTI Dataset. *[LC]* denotes *loop closure*. Since the Seq. 01 is a high-speed sequence, its movement pattern is significantly different from those of other sequences. *[Avg.*]* shows the mean ATE excluding Seq 01. *VGGT-Long* achieves relatively good tracking accuracy without the input of camera intrinsic matrix. *[OOM]* is short for *CUDA Out-Of-Memory* on a single RTX 4090. *[TL]* is short for *Tracking Lost*. See Fig. 7 for trajectory visual results. Color defination: [first], (second), third.

Segment ID	Calib.	Avg.	163453191	183829460	315615587	346181117	371159869	405841035	460417311	520018670	610454533
Frame num.	-	*198*	*198*	*199*	*199*	*199*	*196*	*199*	*198*	*199*	*198*
Segment length	-	*172.533*	*159.963*	*42.301*	*165.149*	*351.213*	*272.661*	*85.743*	*265.906*	*134.552*	*62.739*
Segment speed	-	*0.871*	*0.808*	*0.213*	*0.830*	*1.765*	*1.391*	*0.431*	*1.343*	*0.676*	*0.317*
Traffic	-	-	*Low*	*High*	*Low*	*Low*	*Medium*	*Low*	*Medium*	*Low*	*High*
DROID SLAM [34]	*Required*	(4.396)	(3.705)	[0.301]	[0.447]	(8.653)	9.320	7.621	(4.170)	*TL*	[0.264]
MASt3R-SLAM [24]	*No Need*	5.560	4.500	(0.556)	1.833	12.544	(8.601)	[1.412]	5.428	(7.910)	1.195
CUT3R [42]	*No Need*	9.872	8.781	3.810	5.790	24.015	13.070	7.261	13.206	8.597	3.229
Fast3R [44]	*No Need*	/	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*
VGGT [41]	*No Need*	/	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*
VGGT-Long (Ours)	*No Need*	[1.996]	[1.753]	2.629	(0.559)	[3.452]	[3.343]	(1.444)	[1.541]	[2.547]	(0.455)

Table 2. Camera Tracking Results (ATE RMSE [m] ↓) on the Waymo Open Dataset. Color defination: [first], (second).

ing or domain adaptation.

In the experiments, the previous methods achieve lower tracking accuracies. The reasons for DROID-SLAM's poor performance on long outdoor sequences have been thoroughly discussed in [8, 20]. MASt3R-SLAM works in short sequences but often fails in longer ones, a limitation tied to its reliance on MASt3R [17] for feature matching. In autonomous driving scenarios with less scene variation than that in indoor scenarios, like on a long straight road, the system may not generate new keyframes for extended periods. When it finally does, the significant distance from the last keyframe causes feature matching to fail, leading to tracking loss. As for CUT3R, it employs continuous state tokens to encode the entire 3D scene, which is analogous to NeRF [21] in novel view synthesis. Consequently, CUT3R faces similar challenges as NeRF in large-scale outdoor scenes. That is, a compact representation struggles to capture the vast geometric details of expansive large-scale outdoor environments.

4.5. Loop Optimization Analysis & Ablation Study

Table 5 reports the runtime of VGGT-Long on several KITTI sequences. All measurements exclude disk I/O time, as it can vary depending on OS background programs, disk bandwidth, and memory throughput. On our computer, each chunk takes approximately 25ms to load and 95ms to write, which is negligible given that each KITTI sequence contains only a few dozen chunks. Therefore, disk latency does not significantly impact overall runtime.

Our system achieves efficient chunk-wise processing (about 2.6–2.8s per chunk) and fast Sim(3) alignment (about 0.2s). We evaluate the effectiveness and efficiency of our loop optimization module in Fig. 6. The proposed Sim(3)-based Levenberg-Marquardt solver converges within **3 iterations** on average and takes less than **15ms per step** (taking 0.4-1.3ms/iter in C++ or 3.5–13ms/iter in Python). These results demonstrate the practicality of our system in real-time or near real-time scenarios even on kilometer-scale sequences.

Table 6 presents an ablation study on KITTI sequences. "LC" indicates whether loop closure is enabled. "IRLS" denotes the use of iterative reweighted least squares in chunk alignment; disabling it results in a single-pass confidence-weighted alignment. "Weight" refers to whether confidence-based weighting is applied during alignment. To disable it, we normalize the VGGT confidence map to a uniform value across all points. Removing either the loop closure module or the IRLS weighting results in noticeable accuracy degradation. Specifically, removing loop closure increases the ATE to 58.69m on Seq. 00, while disabling IRLS leads to a 13% drop in performance. The best performance is achieved when all components, loop closure,

Segment ID	Metric	Calib.	Avg.	163453191	183829460	315615587	346181117	371159869	405841035	460417311	520018670	610454533
Frame num.	-	-	*198*	*198*	*199*	*199*	*199*	*196*	*199*	*198*	*199*	*198*
Segment length	-	-	*172.533*	*159.963*	*42.301*	*165.149*	*351.213*	*272.661*	*85.743*	*265.906*	*134.552*	*62.739*
Segment speed	-	-	*0.871*	*0.808*	*0.213*	*0.830*	*1.765*	*1.391*	*0.431*	*1.343*	*0.676*	*0.317*
Traffic	-	-	-	*Low*	*High*	*Low*	*Low*	*Medium*	*Low*	*Medium*	*Low*	*High*
DROID-SLAM [34]	Accuracy ↓	*Required*	*1.201*	[0.781]	1.136	2.247	2.393	[1.090]	[0.539]	[0.740]	*TL*	[0.677]
	Completeness ↓		*8.540*	4.610	10.245	5.540	8.669	8.592	11.144	5.320	*TL*	14.201
	Chamfer ↓		*4.870*	2.696	5.691	3.893	5.531	4.841	5.842	3.030	*TL*	7.439
MASt3R-SLAM [24]	Accuracy ↓	*No Need*	3.772	3.189	2.988	3.787	4.689	4.436	1.166	4.637	6.417	2.637
	Completeness ↓		3.177	[1.715]	[3.284]	2.047	[2.981]	[2.679]	[2.895]	2.002	4.429	6.560
	Chamfer ↓		3.474	2.452	3.136	2.917	3.835	3.558	2.031	3.319	5.423	4.599
CUT3R [42]	Accuracy ↓	*No Need*	3.884	3.580	1.144	2.418	3.712	3.679	4.346	2.012	12.320	1.744
	Completeness ↓		6.801	8.251	9.352	8.748	8.537	5.467	3.393	6.164	[2.302]	8.999
	Chamfer ↓		5.343	5.916	5.248	5.583	6.125	4.573	3.869	4.088	7.311	5.371
VGGT-Long (Ours)	Accuracy ↓	*No Need*	[1.182]	1.002	[0.395]	[0.925]	[1.668]	2.580	0.679	0.784	[1.358]	1.246
	Completeness ↓		[2.860]	2.762	3.417	[1.738]	3.261	2.791	3.216	[1.840]	4.694	[2.022]
	Chamfer ↓		[2.021]	[1.882]	[1.906]	[1.331]	[2.465]	[2.685]	[1.948]	[1.312]	[3.026]	[1.634]

Table 3. Point Map Estimation Results on the Waymo Open Dataset. The GT point cloud is collected by LiDAR, which on vehicles is typically scanning at a height below human eye level, the GT point cloud's coverage is actually narrower than the observation range of RGB cameras. As can be seen in *Segment 371159869*, the LiDAR-captured GT point cloud fails to perceive the 3D structure of the overpass, whereas almost all visual-based methods can, to varying degrees, detect and reconstruct this structure. Therefore, the numerical values in this table should be taken as reference only, and readers are encouraged to refer to Fig. 9 to Fig. 12 for a more intuitive visual comparison between methods. However, whether by visual comparison or the reference numerical values, it is evident that VGGT-Long achieves high reconstruction performance than other methods. Color defination: [first].

	Calib.	**Scene 01** *Speed: 0.743, Num.: 447, Length 332.487 m*						**Scene 02** *Speed: 0.509, Num.: 223, Length 113.603 m*					
Condition	-	*Clone*	*Fog*	*Morning*	*Overcast*	*Rain*	*Sunset*	*Clone*	*Fog*	*Morning*	*Overcast*	*Rain*	*Sunset*
DROID-SLAM [34]	*Required*	(1.0267)	(1.8679)	(0.9886)	(1.0146)	[0.7763]	[1.1453]	[0.0981]	[0.0401]	[0.0493]	[0.0478]	[0.0363]	[0.1127]
MASt3R-SLAM [24]	*No Need*	*TL*	*TL*	*TL*	*TL*	*TL*	*TL*	*TL*	*TL*	*TL*	*TL*	*TL*	*TL*
CUT3R [42]	*No Need*	43.3039	62.1908	50.6081	38.7345	51.5482	43.7854	23.7711	9.9484	28.4150	24.6440	7.9632	25.9732
Fast3R [44]	*No Need*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*
VGGT [41]	*No Need*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*
VGGT-Long (Ours)	*No Need*	[0.7632]	[0.8739]	[0.9279]	[0.6704]	(1.7991)	(1.2593)	(0.7229)	(0.7086)	(0.7214)	(0.6809)	(0.6930)	(0.6890)
	Calib.	**Scene 06** *Speed: 0.192, Num.: 270, Length 51.899 m*						**Scene 18** *Speed: 0.751, Num.: 339, Length 254.443 m*					
Condition	-	*Clone*	*Fog*	*Morning*	*Overcast*	*Rain*	*Sunset*	*Clone*	*Fog*	*Morning*	*Overcast*	*Rain*	*Sunset*
DROID-SLAM [34]	*Required*	[0.0633]	[0.0243]	[0.0302]	[0.0509]	*TL*	[0.0197]	[2.4775]	(2.0316)	(1.8939)	(2.3317)	(2.5497)	(1.9434)
MASt3R-SLAM [24]	*No Need*	*TL*	*TL*	*TL*	*TL*	*TL*	*TL*	*TL*	*TL*	*TL*	*TL*	*TL*	*TL*
CUT3R [42]	*No Need*	0.8358	(0.4078)	0.5990	0.7204	(1.0590)	1.0127	19.4402	8.6279	*6.7201*	20.2124	16.7768	31.1191
Fast3R [44]	*No Need*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*
VGGT [41]	*No Need*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*
VGGT-Long (Ours)	*No Need*	(0.3649)	0.5425	(0.3761)	(0.4024)	[0.5585]	(0.3819)	(1.6509)	[0.7969]	[1.2879]	[1.2556]	[1.6480]	[1.7402]
	Calib.	**Scene 20** *Speed: 0.849, Num: 837, Length 711.229 m*						**Average**					
Condition	-	*Clone*	*Fog*	*Morning*	*Overcast*	*Rain*	*Sunset*	**01 Avg.**	**02 Avg.**	**06 Avg.**	**18 Avg.**	**20 Avg.**	**All Avg.**
DROID-SLAM [34]	*Required*	[3.5915]	[5.0789]	[3.7332]	[3.8521]	[3.7800]	(4.9074)	(1.1366)	[0.0640]	[0.0377]	(2.2046)	[4.1572]	[1.5200]
MASt3R-SLAM [24]	*No Need*	*TL*	*TL*	*TL*	*TL*	*TL*	*TL*	/	/	/	/	/	/
CUT3R [42]	*No Need*	129.4984	76.9624	117.9483	114.5124	66.7003	116.5289	48.3618	20.1191	0.7724	17.1494	103.6918	38.0189
Fast3R [44]	*No Need*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	/	/	/	/	/	/
VGGT [41]	*No Need*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	/	/	/	/	/	/
VGGT-Long (Ours)	*No Need*	(9.6553)	(8.1851)	(6.3447)	(4.5642)	(6.4990)	[4.8500]	[1.0490]	(0.7026)	(0.4377)	[1.3966]	(6.6830)	(2.0538)

Table 4. Camera Tracking Results (ATE RMSE [m] ↓) on the Virtual KITTI Dataset. Color defination: [first], (second).

Chunk Size = 75		**VPR Model**	**Chunk Process**	**Chunk Align**	**LM Opt. (C++)**	**LM Opt. (Python)**
Seq.	**Seq. Frames**	*Time / Frame*	*Time / Chunk*	*Time / Iter*	*Time / Iter*	*Time / Iter*
00	*4542*	21.264 ms	2.811 s	0.284 s	1.249 ms	13.394 ms
05	*2761*	17.023 ms	2.614 s	0.273 s	0.862 ms	8.446 ms
06	*1101*	18.523 ms	2.728 s	0.278 s	0.436 ms	3.592 ms

Table 5. Runtime Analysis of each components of VGGT-Long. All the components are able to operate in almost real-time.

LC	IRLS	Weight	00	05
✗	✓	✓	58.69	36.01
✓	✗	✓	12.29	10.98
✓	✓	✗	11.28	10.13
✓	✓	✓	8.67	8.31

Table 6. Ablation study (ATE RMSE [m] ↓). Experiments demonstrate that loop closure significantly reduces cumulative errors, and the combination with all components yields optimal performance. IRLS and confidence-weighted alignment are enabled.

5. Conclusion

In this work, we presented VGGT-Long, a simple yet effective framework that extends monocular RGB-only 3D reconstruction to long, unbounded video sequences using foundation models. Our method overcomes the GPU memory limitations of existing 3D vision models without requiring camera calibration. Through extensive experiments on KITTI, Waymo, and Virtual KITTI, we demonstrated that VGGT-Long achieves accurate and scalable 3D reconstruction across diverse real-world and synthetic environments. In the future, we will continue researching ways to improve the accuracy and consistency of 3D foundation models for long outdoor sequences.

References

[1] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M Seitz, and Richard Szeliski. Building rome in a day. *Communications of the ACM*, 54 (10):105–112, 2011. 3

[2] Michael Bloesch, Jan Czarnowski, Ronald Clark, Stefan Leutenegger, and Andrew J Davison. Codeslam—learning a compact, optimisable representation for dense visual slam. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2560–2568, 2018. 3

[3] Carlos Campos, Richard Elvira, Juan J. Gomez Rodriguez, Jose M. M. Montiel, and Juan D. Tardos. ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM. *IEEE Transactions on Robotics*, 37(6): 1874–1890, 2021. 3

[4] Hongkai Chen, Zixin Luo, Jiahui Zhang, Lei Zhou, Xuyang Bai, Zeyu Hu, Chiew-Lan Tai, and Long Quan. Learning to match features with seeded graph matching network. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6301–6310, 2021. 3

[5] Igor Cvišić, Ivan Marković, and Ivan Petrović. Soft2: Stereo visual odometry for road vehicles based on a point-to-epipolar-line metric. *IEEE Transactions on Robotics*, 39 (1):273–288, 2022. 2

[6] Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024. 2

[7] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 2

[8] Kai Deng, Yigong Zhang, Jian Yang, and Jin Xie. Gigaslam: Large-scale monocular slam with hierarchical gaussian splats. *arXiv preprint arXiv:2503.08071*, 2025. 2, 5, 7

[9] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 224–236, 2018. 3

[10] Jan-Michael Frahm, Pierre Fite-Georgel, David Gallup, Tim Johnson, Rahul Raguram, Changchang Wu, Yi-Hung Jen, Enrique Dunn, Brian Clipp, Svetlana Lazebnik, et al. Building rome on a cloudless day. In *European conference on computer vision*, pages 368–381. Springer, 2010. 3

[11] A Gaidon, Q Wang, Y Cabon, and E Vig. Virtual worlds as proxy for multi-object tracking analysis. In *CVPR*, 2016. 6

[12] Xiang Gao, Rui Wang, Nikolaus Demmel, and Daniel Cremers. Ldso: Direct sparse odometry with loop closure. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2198–2204. IEEE, 2018. 7

[13] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012. 6

[14] Xingyi He, Jiaming Sun, Yifan Wang, Sida Peng, Qixing Huang, Hujun Bao, and Xiaowei Zhou. Detector-free structure from motion. *CVPR*, 2024. 3

[15] Sergio Izquierdo and Javier Civera. Optimal transport aggregation for visual place recognition. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, pages 17658–17668, 2024. 4

[16] Krishna Murthy Jatavallabhula, Soroush Saryazdi, Ganesh Iyer, and Liam Paull. gradslam: Automagically differentiable slam. *arXiv preprint arXiv:1910.10672*, 2019. 3

[17] Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Grounding image matching in 3d with mast3r. In *European Conference on Computer Vision*, pages 71–91. Springer, 2024. 2, 3, 7

[18] Philipp Lindenberger, Paul-Edouard Sarlin, Viktor Larsson, and Marc Pollefeys. Pixel-Perfect Structure-from-Motion with Featuremetric Refinement. In *ICCV*, 2021. 3

[19] Philipp Lindenberger, Paul-Edouard Sarlin, and Marc Pollefeys. Lightglue: Local feature matching at light speed. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17627–17638, 2023. 3

[20] Lahav Lipson, Zachary Teed, and Jia Deng. Deep patch visual slam. In *European Conference on Computer Vision*, pages 424–440. Springer, 2025. 5, 7

[21] 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):99–106, 2021. 7

[22] Raul Mur-Artal and Juan D. Tardos. ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017. 3, 7

[23] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardos. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015. 3

[24] Riku Murai, Eric Dexheimer, and Andrew J Davison. Mast3r-slam: Real-time dense slam with 3d reconstruction priors. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 16695–16705, 2025. 2, 3, 7, 8

[25] Maxime Oquab, Timothée Darcet, Theo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Russell Howes, Po-Yao Huang, Hu Xu, Vasu Sharma, Shang-Wen Li, Wojciech Galuba, Mike Rabbat, Mido Assran, Nicolas Ballas, Gabriel Synnaeve, Ishan Misra, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2023. 4

[26] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In *Proceedings third international conference on 3-D digital imaging and modeling*, pages 145–152. IEEE, 2001. 6

[27] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016. 3

[28] 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. 3

[29] Hauke Strasdat, J Montiel, and Andrew J Davison. Scale drift-aware large scale monocular slam. *Robotics: science and Systems VI*, 2(3):7, 2010. 5

[30] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 573–580. IEEE, 2012. 6

[31] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2446–2454, 2020. 6

[32] Chengzhou Tang and Ping Tan. Ba-net: Dense bundle adjustment network. *arXiv preprint arXiv:1806.04807*, 2018. 3

[33] Zachary Teed and Jia Deng. Deepv2d: Video to depth with differentiable structure from motion. *arXiv preprint arXiv:1812.04605*, 2018. 3

[34] 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. 3, 7, 8

[35] Zachary Teed, Lahav Lipson, and Jia Deng. Deep patch visual odometry. *Advances in Neural Information Processing Systems*, 36, 2024. 7

[36] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustment—a modern synthesis. In *International workshop on vision algorithms*, pages 298–372. Springer, 1999. 2, 3

[37] Michał Tyszkiewicz, Pascal Fua, and Eduard Trulls. Disk: Learning local features with policy gradient. *Advances in Neural Information Processing Systems*, 33:14254–14265, 2020. 3

[38] Benjamin Ummenhofer, Huizhong Zhou, Jonas Uhrig, Nikolaus Mayer, Eddy Ilg, Alexey Dosovitskiy, and Thomas Brox. Demon: Depth and motion network for learning monocular stereo. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5038–5047, 2017. 3

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 2

[40] 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*, pages 21686–21697, 2024. 3

[41] 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. 2, 3, 6, 7, 8

[42] Qianqian Wang, Yifei Zhang, Aleksander Holynski, Alexei A Efros, and Angjoo Kanazawa. Continuous 3d perception model with persistent state. *arXiv preprint arXiv:2501.12387*, 2025. 2, 3, 7, 8

[43] 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*, pages 20697–20709, 2024. 2, 3

[44] 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. 2, 3, 7, 8

[45] Nan Yang, Rui Wang, Jorg Stuckler, and Daniel Cremers. Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry. In *Proceedings of the European conference on computer vision (ECCV)*, pages 817–833, 2018. 2

[46] Huangying Zhan, Chamara Saroj Weerasekera, Jia-Wang Bian, Ravi Garg, and Ian Reid. Df-vo: What should be learnt for visual odometry? *arXiv preprint arXiv:2103.00933*, 2021. 2

[47] Ji Zhang and Sanjiv Singh. Visual-lidar odometry and mapping: Low-drift, robust, and fast. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 2174–2181. IEEE, 2015. 2

[48] Ji Zhang, Sanjiv Singh, et al. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and systems*, pages 1–9. Berkeley, CA, 2014. 2

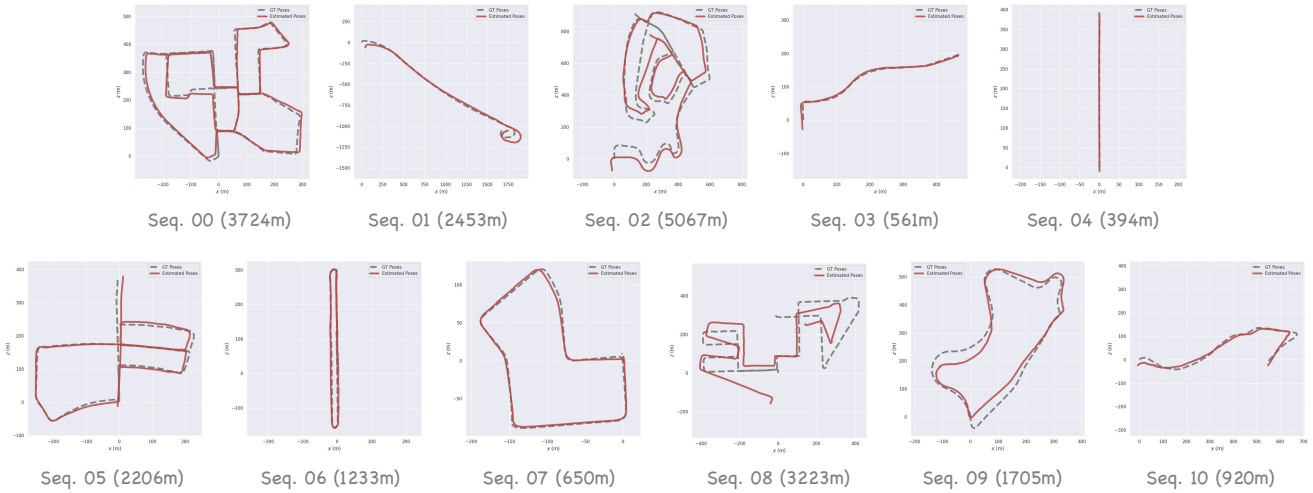


Figure 7. Trajectory visual results of KITTI Dataset. Although there is overlap in 3D space in Sequence 08, it often corresponds to different driving directions within the same lane. In such cases, the 2D RGB loop detection module fails to identify potential loops. Nevertheless, even under such circumstances, according to Table 1, VGGT-Long still outperforms many existing methods in terms of numerical accuracy.

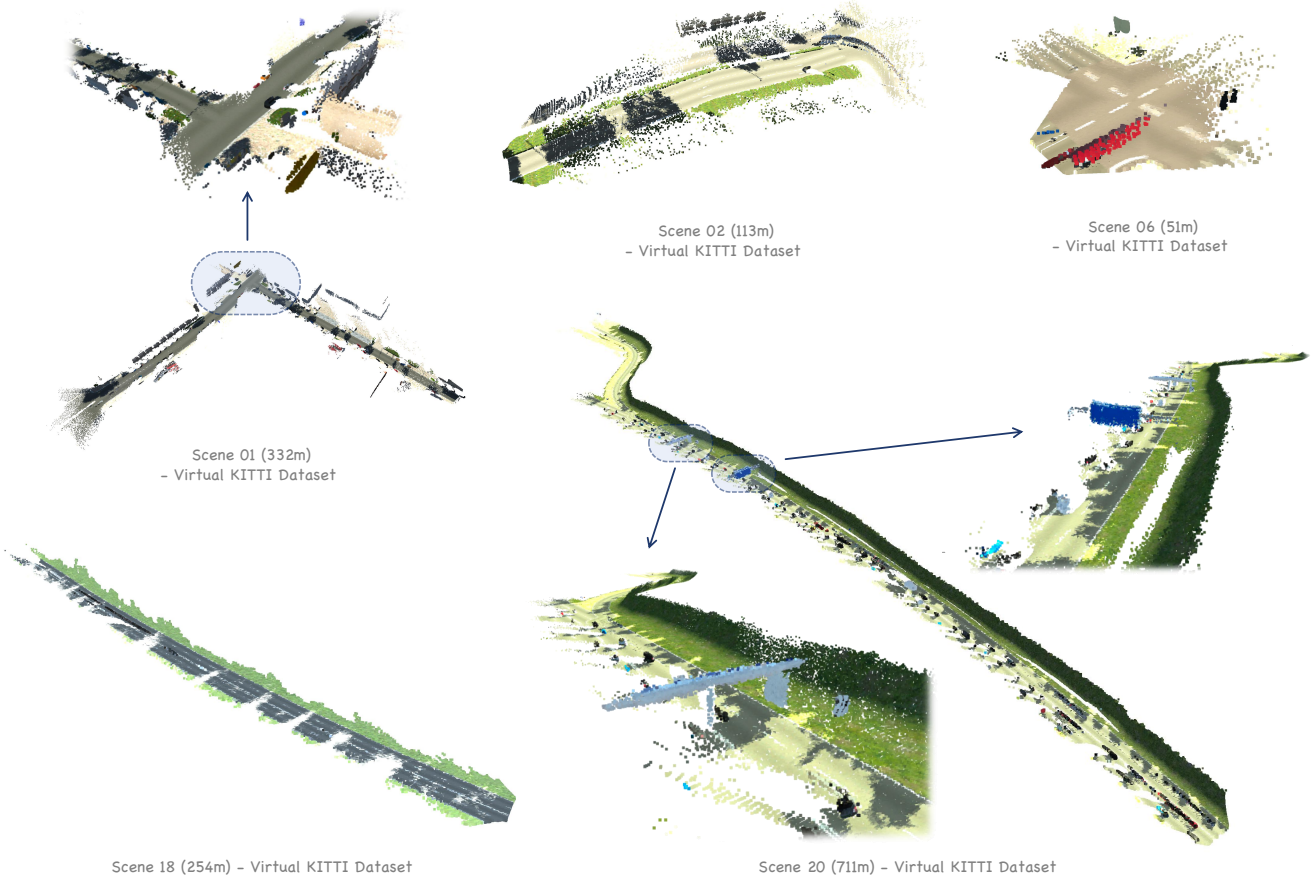


Figure 8. Visual results of Virtual KITTI Dataset.

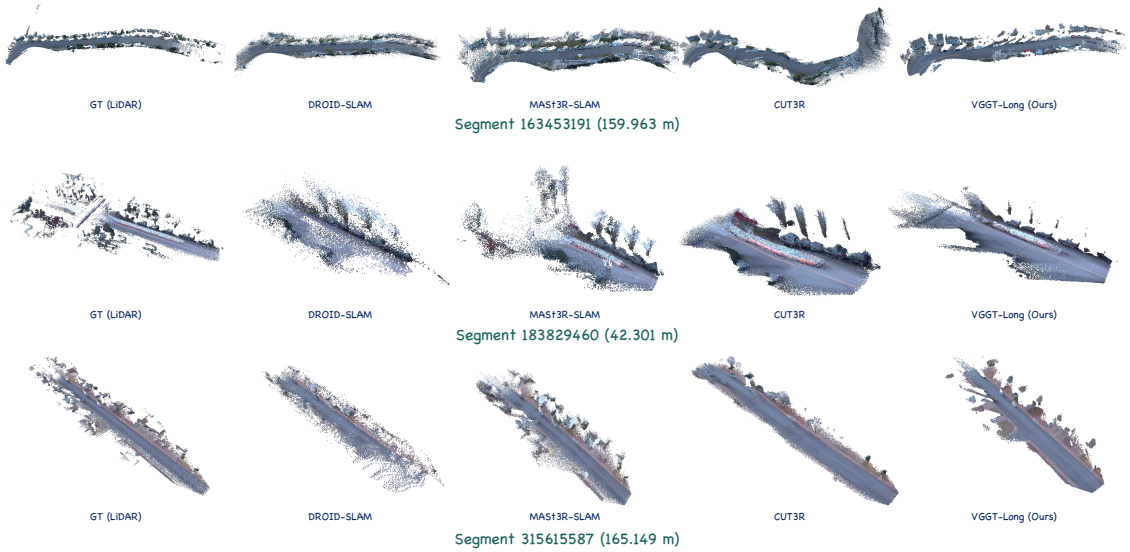


Figure 9. Visual comparison of Waymo Open Dataset. Part 1.



Figure 10. Visual comparison of Waymo Open Dataset. Part 2.

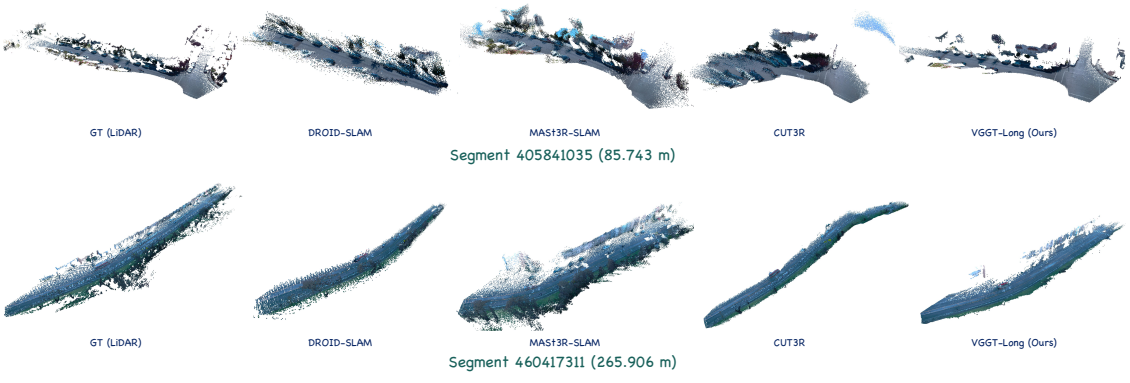


Figure 11. Visual comparison of Waymo Open Dataset. Part 3.

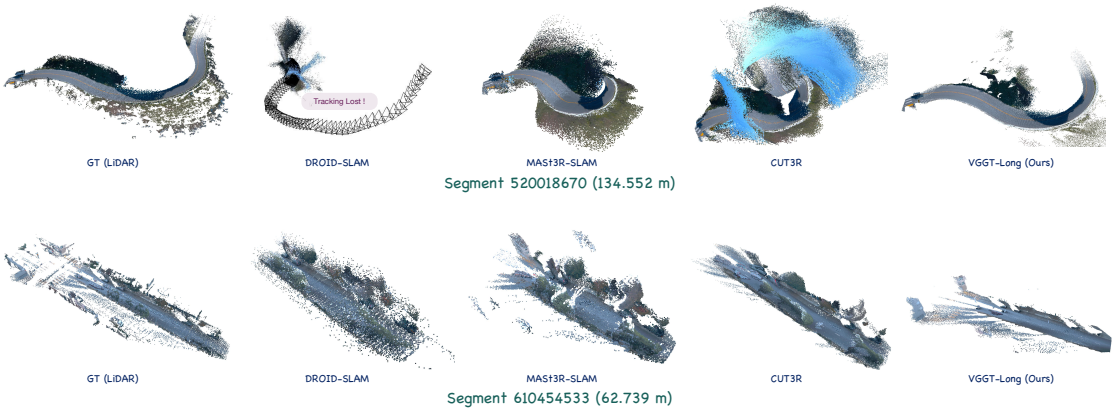


Figure 12. Visual comparison of Waymo Open Dataset. Part 4.