
GeoVideo: Introducing Geometric Regularization into Video Generation Model

–Supplementary Materials–

Yunpeng Bai¹ Shaoheng Fang¹ Chaohui Yu^{2,3} Fan Wang² Qixing Huang¹

¹The University of Texas at Austin, ²DAMO Academy, Alibaba Group, ³Hupan Lab

1 More Visualization Results

For more visualization results, please visit this website:

<https://geovideo.github.io/GeoVideo/>

2 Implementation Details

To evaluate the geometric reliability of generated videos, we adopt two complementary metrics: **Multi-View Consistency Score (MVCS)** and **Reprojection Error**. Both are computed from the predicted depth maps and estimated camera poses from VGGT [3], and capture different aspects of multi-view structural consistency.

Multi-View Consistency Score (MVCS)

MVCS measures the temporal consistency of the predicted depth maps in adjacent frames. MVCS is computed for adjacent frames by default.

Given predicted depth maps D_i and camera poses T_i for each frame i , we define MVCS as follows:

1. For each pixel $\mathbf{u} = (x, y)$ in frame i , back-project to a 3D point using intrinsics K :

$$\mathbf{p}_i = T_i^{-1} \cdot D_i(\mathbf{u}) \cdot K^{-1} \begin{bmatrix} \mathbf{u} \\ 1 \end{bmatrix} \quad (1)$$

2. Transform \mathbf{p}_i into the frame j ' coordinate system and project to image space:

$$\hat{\mathbf{u}}_{i \rightarrow j} = \pi(KT_j\mathbf{p}_i) \quad (2)$$

3. Sample the predicted depth at the projected location and compute the warped depth:

$$\hat{D}_{i \rightarrow j}(\hat{\mathbf{u}}) = \text{depth of } \mathbf{p}_i \text{ in frame } j \quad (3)$$

4. Compute consistency at valid pixels via a truncated normalized error:

$$\text{CS} = 1 - \frac{1}{N} \sum_{\mathbf{u}} \min \left(1, \frac{|\hat{D}_{i \rightarrow j}(\hat{\mathbf{u}}) - D_j(\hat{\mathbf{u}})|}{\delta} \right) \quad (4)$$

Here, δ is a threshold (0.05), and N is the number of valid pixels. Higher CS indicates better geometric consistency between frames.

To evaluate consistency over a sequence, we compute the MVCS between multiple frame pairs and take the average:

$$\text{MVCS} = \frac{1}{|\mathcal{S}|} \sum_{\substack{i,j \\ (i,j) \in \mathcal{S}}} \text{CS}(i \rightarrow j) \quad (5)$$

Here, \mathcal{S} denotes the set of selected frame pairs (e.g., adjacent or temporally close). The average MVCS captures overall multi-view geometric consistency across the video sequence.

Reprojection Error

Reprojection Error measures how well the reconstructed 3D scene aligns with original image observations.

1. Back-project each depth map D_i to a set of 3D points \mathcal{P}_i , storing the source pixel \mathbf{u}_i and the source frame index.
2. Fuse all \mathcal{P}_i into a global point cloud $\mathcal{X}_{\text{global}}$, applying voxel grid simplification to fuse points from different views, while retaining per-point metadata during the fusion process.
3. For each point $\mathbf{p} \in \mathcal{X}_{\text{global}}$, project it back to its original view:

$$\hat{\mathbf{u}}_i = \pi(K_i T_i \mathbf{p}) \quad (6)$$

4. Compute the 2D reprojection error as the pixel-wise Euclidean distance:

$$\text{ReprojError} = \frac{1}{|\mathcal{X}_{\text{global}}|} \sum_{\mathbf{p}} \|\hat{\mathbf{u}}_i - \mathbf{u}_i\|_2 \quad (7)$$

A lower Reprojection Error indicates more accurate alignment between the global geometry and the source image observations.

Voxel Grid Simplification and Denoising

To improve the quality and usability of the reconstructed global point cloud $\mathcal{X}_{\text{global}}$, we apply two commonly used point cloud processing techniques: *voxel grid downsampling* and *statistical outlier removal*. Voxel grid downsampling is applied during both training and evaluation to reduce redundancy and improve computational efficiency. In contrast, statistical outlier removal is only used during training to suppress noisy depth predictions and is omitted during evaluation to preserve the raw geometry for accurate metric computation.

Voxel Grid Simplification. To reduce redundancy in dense regions of the fused point cloud and improve computational efficiency, we apply voxel grid simplification while preserving the overall 3D structure. Specifically, the 3D space is divided into uniform voxels of size τ , and all points falling within the same voxel are aggregated into a single representative point, calculated as the centroid of the points in that voxel.

Formally, each point \mathbf{p}_i is assigned to a voxel indexed by:

$$\mathbf{v}_i = \left\lfloor \frac{\mathbf{p}_i}{\tau} \right\rfloor, \quad (8)$$

and the representative point for voxel \mathbf{v} is:

$$\bar{\mathbf{p}}_{\mathbf{v}} = \frac{1}{|\mathcal{V}|} \sum_{\mathbf{p}_i \in \mathcal{V}} \mathbf{p}_i, \quad (9)$$

where \mathcal{V} denotes the set of all points falling into voxel \mathbf{v} . The final simplified point cloud consists of all such representative points.

To ensure that the size of the voxel is adaptable to the spatial density of the reconstructed point cloud, we estimate a representative nearest-neighbor distance d at all points. Specifically, for each point \mathbf{p}_i , we compute the distance to its nearest neighbor:

$$d_i = \min_{j \neq i} \|\mathbf{p}_i - \mathbf{p}_j\|_2, \quad (10)$$

and define d as the q -th quantile of the distance distribution:

$$d = \text{quantile}_q(\{d_i\}_{i=1}^N), \quad (11)$$

where q is typically set to 0.05 to reduce the influence of outliers. We then set the voxel size as:

$$\tau = \alpha \cdot d. \quad (12)$$

In our setting, we fuse the $F = 81$ frames into a single global point cloud. To ensure that each voxel aggregates roughly one contribution per frame, we set the expected number of points per voxel to $n = 81$. Assuming a locally uniform distribution, the density of the point cloud is $\rho = 1/d^3$, and the expected number of points per voxel is:

$$n_{\text{voxel}} = \rho \cdot \tau^3 = \frac{\tau^3}{d^3} = \alpha^3. \quad (13)$$

Solving for α under the constraint $\alpha^3 = 81$ yields:

$$\alpha = \sqrt[3]{81} \approx 4.33, \quad (14)$$

and thus, the voxel size is set as:

$$\tau = 4.33 \cdot d. \quad (15)$$

This adaptive strategy ensures consistent geometric fusion across frames, preserving spatial resolution while eliminating redundancy.

Statistical Outlier Removal. To further denoise the point cloud and remove outliers caused by inconsistent depth estimates, we apply statistical outlier removal. For each point, we compute the average distance to its k nearest neighbors. Points whose mean distances deviate significantly (i.e., beyond a certain threshold μ) from the global distribution are classified as outliers and removed.

Given a point \mathbf{p}_i and its k neighbors $\{\mathbf{p}_{i_1}, \dots, \mathbf{p}_{i_k}\}$, we calculate the average neighbor distance:

$$d_i = \frac{1}{k} \sum_{j=1}^k \|\mathbf{p}_i - \mathbf{p}_{i_j}\|_2 \quad (16)$$

Then, the global mean \bar{d} and standard deviation σ_d of all d_i values are calculated. A point is retained if:

$$d_i \leq \bar{d} + \mu \cdot \sigma_d \quad (17)$$

We typically use $k = 20$ and $\mu = 1.0$ in our experiments. This process helps to eliminate sparse outliers while preserving meaningful geometric structures.

Camera Pose Head Output. Our model includes a camera pose prediction head, which outputs encoded camera parameters for each frame in the generated video. The camera pose head is adapted from the architecture used in VGGT [3], allowing efficient regression of the translation, rotation, and field of view parameters in a unified encoding. The output is a tensor of shape $B \times S \times 9$, where B is the batch size and S is the sequence length (number of frames). Each 9-dimensional vector represents the camera pose in the absT_quaR_FoV format, consisting of:

- **Translation** $\mathbf{T} \in \mathbb{R}^3$: the absolute 3D position of the camera.
- **Rotation (Quaternion)** $\mathbf{q} \in \mathbb{R}^4$: a unit quaternion representing the camera orientation.
- **Field of View (FoV)** $\mathbf{f} \in \mathbb{R}^2$: horizontal and vertical field-of-view parameters.

Formally, the camera pose vector for frame t is:

$$\text{pose}_t = [\mathbf{T}_t; \mathbf{q}_t; \mathbf{f}_t] \in \mathbb{R}^9$$

The motion probability head and depth head are adapted from MegaSaM [2] and Video Depth Anything [1], respectively, with the modification that they take as input the features of the generated video of all frames. Since the MegaSaM motion probability map has a lower resolution than the input, during training we down-sample the predicted motion features accordingly to match the resolution of the MegaSaM output before computing the loss.

References

- [1] Sili Chen, Hengkai Guo, Shengnan Zhu, Feihu Zhang, Zilong Huang, Jiashi Feng, and Bingyi Kang. Video depth anything: Consistent depth estimation for super-long videos. *arXiv preprint arXiv:2501.12375*, 2025.
- [2] Zhengqi Li, Richard Tucker, Forrester Cole, Qianqian Wang, Linyi Jin, Vickie Ye, Angjoo Kanazawa, Aleksander Holynski, and Noah Snavely. Megasam: Accurate, fast, and robust structure and motion from casual dynamic videos. *arXiv preprint arXiv:2412.04463*, 2024.
- [3] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.